



INSTITUTO FEDERAL  
Sul de Minas Gerais

2018

MEDIOTEC - REDE - E TEC

# APOSTILA DE **Informática** Programação para Web III

Prof. : Elton Adriano Ribeiro da Silva



## Palavra do professor-autor

Prezado Aluno!

O objetivo desta apostila é auxiliá-lo da melhor maneira a compreender e a utilizar os principais recursos práticos e teóricos da disciplina Programação para WEB III. Por meio deste estudo, você poderá absorver e aumentar seu conhecimento através de pesquisas, leituras, práticas e discussões.

## Apresentação da disciplina

No andamento desta disciplina, teremos várias atividades relacionadas a cada aula estudada através de nosso Ambiente Virtual de Aprendizagem. Aliar a teoria com a prática é a melhor maneira de atingir o conhecimento. Busque participar ativamente e sempre envie suas dúvidas ao Professor Formador e ao Professor Mediador.

Discutiremos ao longo da disciplina:

- Funcionamento dos computadores pessoais.
- Principais dispositivos e componentes de um computador.
- Montagem de computadores.
- Possíveis problemas de drivers e dispositivos.
- Identificação e verificação de tensões de alimentação de um computador.
- Testes de funcionalidades de dispositivos.
- Configuração de sistemas operacionais.
- Backup.
- Segurança de dados.
- Periféricos.
- Conexão física entre dispositivos.

Entre outros assuntos pertinentes a nossa disciplina.

Minha função será orientá-lo, apoiá-lo e acompanhá-lo em seu processo de aprendizagem, ajudando-o em suas necessidades e dúvidas.

Sucesso!

# Sumário

	<b>1</b>
<b>Palavra do professor-autor</b>	<b>2</b>
<b>Apresentação da disciplina</b>	<b>2</b>
<b>Aula 1 – World Wide Web (WWW)</b>	<b>5</b>
1.1. O que é	5
1.2. O que é Hipertexto	5
<b>Aula 2 – O Funcionamento dos Sistemas Web – HTML</b>	<b>7</b>
2.1. – O HTML	7
2.2. – Estrutura básica de uma página HTML	7
<b>Aula 3 – Programação Cliente x Programação Servidor</b>	<b>11</b>
3.1. – Programação Cliente	11
3.2. – Programação Servidor	11
<b>Aula 4 – Arquitetura Web</b>	<b>13</b>
4.1. – Arquitetura da Informação	13
4.2. - Três Tipos de Arquitetura de Aplicação Web	13
4.2.1. - Servidor HTML	14
4.2.2. - “Widgets” de geração JS (AJAX)	14
4.2.3. - Página única orientada para os serviços de aplicações Web (Web 2.0 e aplicações HTML5)	14
4.3. - Programação para Web	14
<b>Aula 5 – A Linguagem de Programação PHP</b>	<b>17</b>
5.1. – PHP - definição	17
5.2. - Servidor de Teste	17
5.3. – IDE	20
5.4. - Testando o Ambiente	21
<b>Aula 6 – Sintaxe Básica e Estudo das Variáveis</b>	<b>23</b>
6.1 - Página PHP	23
6.2. - Sintaxe Básica	24
6.3. – Variáveis	25
6.3.1 - Nome das Variáveis	25
6.3.2 – Declaração de Variáveis	25
6.3.3 – Escopo de uma variável	25
6.4. – Strings	27
6.4.1 – Operação de Concatenação	27
6.4.2 – Função strlen ( )	27
6.4.3 - Função strpos( )	27
6.5. – Operadores	28

6.5.1 – Operadores Aritméticos	28
6.5.2 – Operadores de Atribuição	28
6.5.3 – Operadores de Incremento e Decremento	28
6.5.4 - Operadores de Comparação	28
6.5.5 - Operadores Lógicos	29
<b>6.6. – If...Else</b>	29
<b>6.7. – Switch</b>	30
<b>Aula 7 – Estruturas de Repetição</b>	<b>32</b>
7.1. – Estruturas de Repetição	32
7.1.1 – While	32
7.1.2 - Do...While	32
7.1.3 – For	33
<b>Aula 8 – Funções em PHP</b>	<b>34</b>
8.1 – Funções	34
8.2 – Adicionando Parâmetros	34
8.3 – Retornando Valores	35
8.4 – Formulários	35
8.4.1 – Método GET	35
8.4.2 – Método POST	36
<b>Aula 9 - Introdução à Programação em Linguagem Orientada a Objetos</b>	<b>37</b>
9.1. - Programação Orientada a Objetos (POO)	37
<b>Referências</b>	<b>39</b>

## Aula 1 – World Wide Web (WWW)

Objetivo: Conceituar World Wide Web

### 1.1. O que é

Segundo Bert & Sierra (2009), World Wide Web ou WWW é um sistema de documentos dispostos na Internet que permitem o acesso às informações apresentadas no formato de hipertexto. Para ter acesso a tais informações, pode-se usar um programa de computador chamado navegador. Os navegadores mais famosos são: [Internet Explorer](#), [Mozilla Firefox](#), [Google Chrome](#) e [Safari](#).

De acordo com o site Tech Tudo, a ideia de World Wide Web surgiu em 1980, na Suíça. O precursor foi o britânico Tim Berners-Lee. Um computador NeXTcube foi usado por Berners-Lee como primeiro servidor web e também para escrever o primeiro navegador, o WorldWideWeb, em 1990.

Em 6 de agosto de 1991, Tim Berners-Lee postou um resumo sobre todas as suas ideias e projetos no grupo de notícias de nome alt.hypertext. Essa data marca a estreia oficial da Web como um serviço publicado na Internet.

A partir de então, com o avanço das tecnologias de internet, a web também pôde se desenvolver grandiosamente, aumentando significativamente a quantidade de conteúdo disponível e também a interação entre usuários e páginas. A chamada Web 2.0, desenvolvida a partir do início dos anos 2000, é a fase na qual o conteúdo da web é oferecido também pelos seus usuários em blogs, vlogs, fotologs e redes sociais.

Atualmente, podemos dizer que vivemos na Web 3.0 (também conhecida como web semântica), que é a fase em que os sistemas de busca estão cada vez mais inteligentes e o conteúdo disponível na teia mundial torna-se cada vez mais personalizado.

### 1.2. O que é Hipertexto

Marcuschi diz que o hipertexto é um conceito associado às tecnologias da informação e que faz referência à escrita eletrônica. Desde sua origem, o hipertexto vem mudando a noção tradicional de autoria, uma vez que ele contempla diversos textos. Trata-se, portanto, de uma espécie de obra coletiva, ou seja, apresenta textos dentro de outros, formando, assim, uma grande rede de informações interativas. Nesse sentido, sua maior diferença é justamente a forma de escrita e leitura. Assim, num texto tradicional, a leitura segue uma linearidade, enquanto no hipertexto ela é não linear. Conforme demonstrado na figura abaixo:

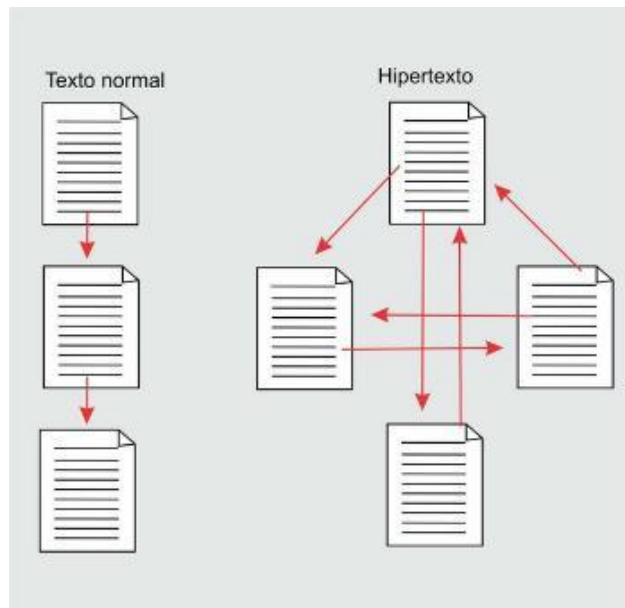


Figura 1 - A diferença entre um texto normal e o hipertexto. Fonte: TechTudo

Um forte exemplo de hipertexto são os artigos na internet. No corpo do texto eles apresentam diversos links ("ligação", em inglês) ou hiperlinks nas palavras ou nos assuntos que estejam relacionados. Isso permite que o próprio leitor tenha uma posição mais ativa, escolhendo as informações que prefere acessar. Além dos artigos da internet, um livro de contos, dicionários e enciclopédias são considerados exemplos de hipertextos. A informação contida neles proporciona um caráter não linear, em que o leitor também pode selecionar as informações e os caminhos de leitura que preferir. Sendo assim, a leitura de hipertexto é realizada por associações.



## **Aula 2 – O Funcionamento dos Sistemas Web – HTML**

Objetivo: Entender o funcionamento dos sistemas web

### **2.1. – O HTML**

De acordo com o site Info Escola, O HTML é uma linguagem de marcação utilizada para desenvolvimento de sites.

Essas linguagens são constituídas de códigos que delimitam conteúdos específicos, segundo uma sintaxe própria. O HTML tem códigos para criar páginas na web e são esses códigos que definem o tipo de letra, qual o tamanho, cor, espaçamento, e vários outros aspectos do site.

No início era muito complicado aprender HTML, pois eram muitos comandos para fazer algo simples. A cada nova versão, o HTML fica mais fácil de utilizar e adquire mais funções. Atualmente, qualquer pessoa pode acessar a internet e aprender a construir um site básico em questão de horas, seguindo os passos de tutoriais e aprendendo as funções de cada código.

O HTML foi à primeira linguagem de nível mundial, porém, não é a única. Existem muitas outras linguagens destinadas à criação de páginas da web, embora o HTML ainda prevaleça. Atualmente já é possível integrar várias linguagens na mesma página da web, possibilitando usar duas ou mais linguagens no mesmo site.

Para criar e editar códigos em HTML é necessário qualquer editor de texto comum, como bloco de notas. Para testar os códigos, basta salvar o arquivo em formato HTML e executar. Para o teste, é necessário ter um navegador configurado como padrão. Não é necessária internet, pois o arquivo com os códigos está na máquina onde está sendo executado.

### **2.2. – Estrutura básica de uma página HTML**

Ao acessar uma página web através de um navegador, ele é capaz de interpretar o código HTML e renderizá-lo de forma compreensível para o usuário final, exibindo textos, botões, etc., com as configurações definidas por meio das diversas tags que essa linguagem dispõe.

Atualmente a HTML encontra-se na versão 5 e é padronizada pelo W3C (World Wide Web Consortium), uma organização internacional responsável por estabelecer padrões para a internet.

A estrutura básica de uma página HTML pode ser vista na imagem a seguir, na qual podemos ver as principais tags que são necessárias para que o documento seja interpretado corretamente pelos browsers.

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04     <meta charset="UTF-8"/>
05     <title>Document</title>
06 </head>
07 <body>
08     <!-- Conteúdo -->
09 </body>
10 </html>
```

Figura 2 – Estrutura básica de uma página HTML. Fonte: DevMedia.

**Linha 1:** a instrução DOCTYPE deve ser sempre a primeira a aparecer em uma página HTML e indica para o browser qual versão da linguagem está sendo usada. Nesse caso, estamos trabalhando com a HTML5, versão na qual a declaração do DOCTYPE é bastante simples, como podemos ver na listagem.

**Linhas 2 e 10:** abertura e fechamento da tag html, que delimita o documento. Sendo assim, as demais tags da página devem estar nesse espaço.

**Linhas 3 e 6:** abertura e fechamento da tag head, que define o cabeçalho do documento. O conteúdo nesse espaço não é visível no browser, mas contém instruções sobre seu conteúdo e comportamento. Dentro dessa tag, por exemplo, podem ser inseridas folhas de estilo e scripts.

**Linha 4:** a tag meta, nesse caso, especifica qual conjunto de caracteres (character set ou charset) será usado para renderizar o texto da página. O UTF-8 contém todos os caracteres dos padrões Unicode e ASCII, sendo, portanto, o mais utilizado em páginas web. A mesma tag meta, porém com outros atributos, pode ser utilizada para outros fins, como na SEO (Search Engine Optimization).

**Linha 5:** a tag title define o título da página, aquele que aparece na janela/aba do navegador.

**Linhas 7 e 9:** abertura e fechamento da tag body, marcando o espaço no qual deve estar contido o conteúdo visual da página. As demais tags que representam texto, botões, etc., devem ser adicionadas nesse intervalo.

**Linha 8:** nessa linha podemos observar a sintaxe para adição de comentários em HTML. Esse trecho não é processado pelo browser.

Para desenvolver páginas com HTML, precisamos basicamente de um editor de texto, como o Bloco de Notas, do Windows, Nano e Emacs, no Linux, entre outros. Há, ainda, editores com opções avançadas, como recursos de syntax highlight e autocomplete, como Sublime Text, Atom, Brackets e Visual Studio Code, que podem ser usados para editar documentos HTML. Independente do editor



utilizado, podemos simplesmente copiar o código da Figura 2 para um deles e salvar o arquivo com extensão html. Em seguida, podemos abrir esse arquivo em um browser. Outro aspecto importante da linguagem é que ela é case insensitive, ou seja, não leva em consideração a diferença entre letras maiúsculas e minúsculas. No entanto, o uso apenas de letras minúsculas tem sido utilizado como padrão pelos desenvolvedores.



## Aula 3 – Programação Cliente x Programação Servidor

Objetivo: Diferenciar Programação Cliente x Servidor.

### 3.1. – Programação Cliente

No lado do cliente, ou seja, do usuário, a aplicação roda diretamente em seu computador. O Lado do cliente dá a resposta em tempo real para qualquer interação que é feita no website.

Por exemplo: A maioria dos websites tem um formulário para ser preenchido e dentro desse formulário é preciso ter validações para verificar se o campo não ficou em branco ou se foram digitados caracteres inválidos.

A maioria dessas validações é feita em Java script (client-side), ou seja, assim que o usuário interage com o formulário, ele já responde de imediato, seja por um botão ou por um comportamento, como mudança de foco numa caixa de texto ou até mesmo enquanto o usuário está digitando algo. O navegador não precisa buscar o código no servidor, pois este já foi baixado para computador do usuário.

Linguagens Client-Side:

- HTML
- CSS
- Java script

A imagem abaixo retrata o fluxo da programação cliente:

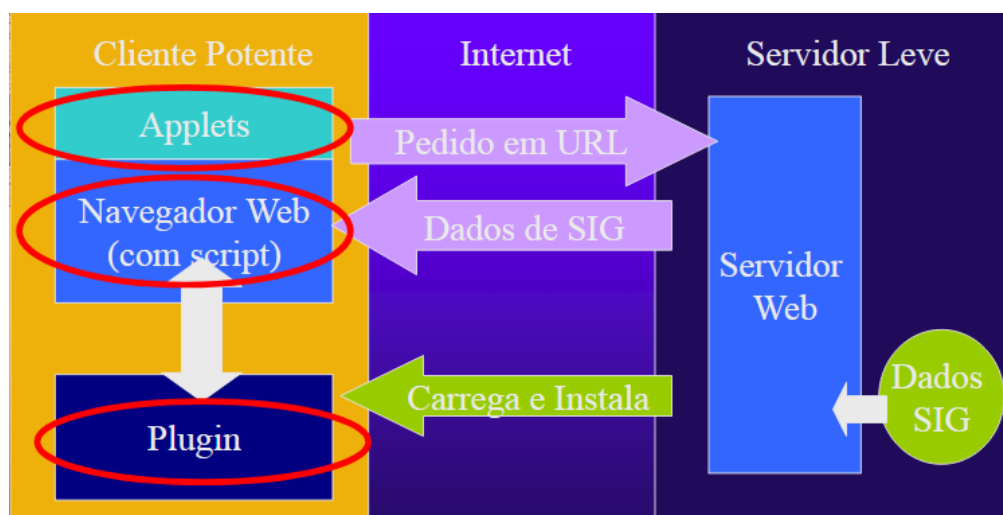


Figura 3 – Programação Cliente. Fonte: SlidPlayer

### 3.2. – Programação Servidor

No lado do servidor, rodam as aplicações necessárias para um website funcionar, como banco de dados e a linguagem que o programador esteja usando para a criação do software, como, por

exemplo, o PHP. Toda vez que o usuário abrir uma página que tenha um código PHP, o navegador vai executar o comando direto do servidor.

Como exemplo pode-se citar o acesso a um banco de dados qualquer, não é possível usar uma linguagem client-side, como o Java script. Teremos de usar, automaticamente, uma linguagem server-side, como PHP.

Linguagens server-side:

- Banco de dados como SQL, MySQL, Oracle...
- PHP
- ASP
- ASP.NET

A imagem abaixo retrata o fluxo da programação servidor:

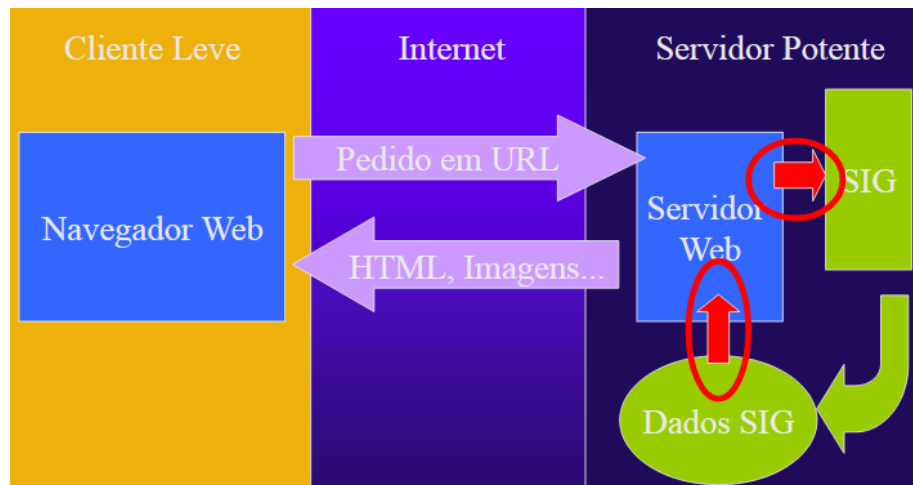


Figura 4 – Programação Servidor. Fonte: SlidPlayer

## **Aula 4 – Arquitetura Web**

Objetivo: Conceituar Arquitetura web

### **4.1. – Arquitetura da Informação**

A arquitetura web tem por objetivo criar as estruturas de organização da informação de um website para que o usuário consiga compreendê-lo com facilidade. Na web, a arquitetura de informação cuida de projetar a estrutura, o esqueleto de um website sobre o qual todas as demais partes irão se apoiar.

West (2001) cita que “arquitetura de informação é a prática de projetar a infraestrutura de um website, especialmente a sua navegação”, e Shiple (2001) afirma que “arquitetura de informação é a fundação para um ótimo web design. Ela é o esquema do website sobre o qual todos os outros aspectos são construídos – forma, função, metáfora, navegação e interface, interação e design visual”. Esse esquema, essa infraestrutura, tem um propósito: atender as necessidades de informação dos usuários do website.

Dijck (2003) cita que “o principal trabalho de um arquiteto de informação é organizar a informação de um website para que seus usuários possam encontrar coisas e alcançar seus objetivos”. Toub (2000) reforça isso citando a minha definição favorita: “Arquitetura de informação é a arte e a ciência de estruturar e organizar ambientes de informação para ajudar as pessoas a satisfazerem suas necessidades de informação de forma efetiva”.

Conhecer os usuários, suas necessidades, hábitos, comportamentos e experiências é fundamental para elaborar a arquitetura de informação de um website, mas não é suficiente. É necessário também entender as características do conteúdo que será apresentado (volume, formato, estrutura, governança, dinamismo, etc.) e as especificidades do contexto de uso (objetivo do website, cultura e política da empresa, ambiente de uso, restrições tecnológicas, etc.). Essa trinca, usuário–conteúdo–contexto e suas interdependências são únicos para cada website e o papel do arquiteto é justamente conseguir equilibrá-las para que a informação certa seja acessada pela pessoa certa no momento certo (ROSENFELD; MORVILLE, 2002).

### **4.2. - Três Tipos de Arquitetura de Aplicação Web**

Expressões como “aplicação web”, “arquitetura front-end”, “Web 2.0” e “aplicações para HTML5” tornaram-se bastante correntes. Acontece, porém, serem frequentemente utilizadas de uma forma errônea, alienada da especificidade da implementação e do uso da arquitetura de aplicação web. Apresento aqui três tipos de aplicações:

#### **4.2.1. - Servidor HTML**

A arquitetura mais conhecida. O servidor gera conteúdo HTML que envia para o cliente como uma página completa de HTML. Por vezes, esse tipo de arquitetura é chamado “Web 1.0”, por ter sido o primeiro a aparecer e a dominar a rede.

#### **4.2.2. - “Widgets” de geração JS (AJAX)**

Arquitetura mais desenvolvida que o primeiro tipo, com a diferença que a página exibida pelo navegador consiste em “widgets” (unidades funcionalmente independentes). Os conteúdos são enviados através da consulta do AJAX ao servidor, ora como página de HTML já construída, ora como JSON, que se converte no conteúdo da página através da ligação Java Script.

A sua principal vantagem é que as atualizações do servidor chegam à parte da página solicitada pelo cliente. Também é positivo que os “widgets” estejam separados de um modo funcional, pois, se um “widget” específico trabalhar com uma parte da página, as mudanças introduzidas não afetam as restantes.

#### **4.2.3. - Página única orientada para os serviços de aplicações Web (Web 2.0 e aplicações HTML5)**

A expressão “Web 2.0” não é a mais adequada, por implicar que os utilizadores introduzam conteúdo. No fundo, refere-se a projetos e serviços ativamente desenvolvidos e sujeitos a melhorias pelos próprios utilizadores, como acontece nos blogs, wikis e nas redes sociais.

Neste caso, uma página HTML contendo código Java Script é descarregada pelo servidor. O código remete para um serviço web específico e recolhe apenas conteúdos corporativos. Estes são utilizados pela aplicação Java Script, que gera a página com conteúdo HTML.

Este tipo de arquitetura representa uma evolução face à anterior, que não permite exibir uma grande quantidade de funções estruturadas e inter-relacionadas.

Note-se ainda que atualmente são raras as aplicações que funcionem autonomamente “off-line”. Esta abordagem permite uma fácil conversão inversa, que consiste na publicação de uma aplicação já existente na web.

### **4.3. - Programação para Web**

O computador ainda não entende exatamente o que o ser humano fala. O que o computador entende são zeros e uns. Graças à velocidade que possui, ele pode ler uma gigantesca quantidade de zeros e uns e fazer coisas incríveis, mas, para os seres humanos é impossível instruir o computador dessa forma. Por isso, foram criados códigos que dão ordens básicas. A organização dessas ordens é

uma gramática e seu conjunto de palavras-chave é sua ortografia, e assim temos uma linguagem, uma forma de comunicação com o computador que diz a ele o que é para ele fazer e quando.

Dessa forma podemos escrever mais facilmente e pedir para uma aplicação traduzir esta linguagem para a que o computador entenda, ou seja, dizemos que uma linguagem mais próxima dos seres humanos é de alto nível, já a linguagem de zeros e uns é de baixo nível. Essa tradução pode possuir duas abordagens: “compilativa” e “interpretativa”.

Na compilação a tradução é feita de uma vez. Todo o programa é traduzido para a linguagem de baixo nível e pode ser executado indefinidamente, sem precisar ser compilado novamente. É como se um tradutor de inglês pegasse um livro e traduzisse ele para o português. O livro traduzido pode ser impresso várias vezes sem precisar ser traduzido novamente. Como exemplos de linguagem “compilativa” temos Pascal e C.

Na interpretação a tradução é realizada enquanto as linhas são lidas. Toda vez que a linha é lida, é traduzida e executada. É como se um tradutor estivesse em uma palestra e, enquanto o palestrante fala, ele vai traduzindo para o público. Se o palestrante repetir tudo de novo, em um momento que ele precise explicar novamente para alguém, o tradutor terá que traduzir novamente. Então, se uma linha for lida mil vezes em um código, ela será interpretada e executada mil vezes. Como exemplo de linguagens interpretativas, temos o PHP, Python e Java Script.

A forma como a internet é construída privilegia as linguagens interpretativas. Observe a Figura 5, a seguir. No passo 1, o usuário solicita uma página para um servidor web. O servidor percebe que a página solicitada é uma aplicação em PHP e solicita sua interpretação no passo 2. Como resultado, o interpretador pode criar uma página web com o resultado da computação e envia como resposta para o usuário, no passo 3.

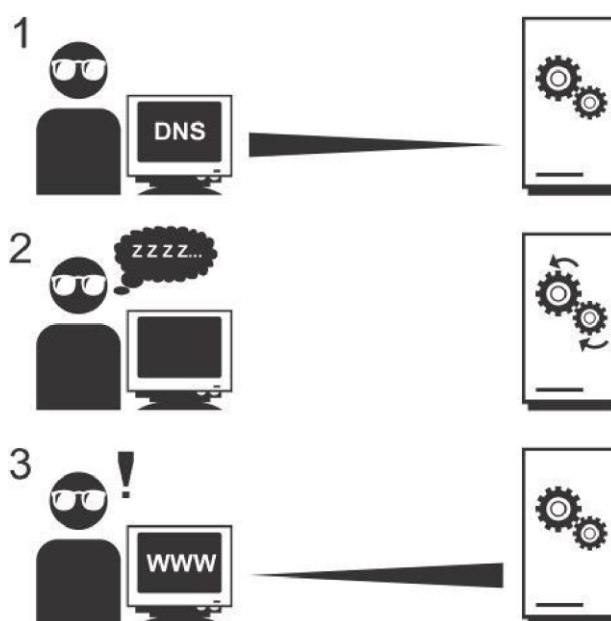


Figura 5 - Ilustração que mostra os passos simplificados da solicitação de uma página web. Fonte: Próprio autor.



Não importa quantas vezes o usuário faça isso ou quantos usuários diferentes solicitem a mesma página, toda vez o aplicativo (programa) vai ter que interpretar e executar. A velocidade de tradução depende de vários fatores, mas, geralmente a interpretação é mais lenta que a compilação.

É dessa forma que vamos pensar de agora em diante. O usuário solicita algo ao programa, ele faz a computação e devolve uma página como resposta.

## Aula 5 – A Linguagem de Programação PHP

Objetivo: Conhecer a linguagem de programação PHP.

### 5.1. – PHP - definição

PHP é uma linguagem de script executada do lado do servidor e é uma poderosa ferramenta para a criação de páginas web dinâmicas e interativas. PHP significa “**P**HP: **H**ypertext **P**reprocessor”. É aberto para quem se interessar em melhorá-lo e gratuito para uso.

A versão atual do PHP é a 7 e pode ser baixada no site [www.php.net](http://www.php.net). Lá, você também pode encontrar novidades sobre o desenvolvimento, documentação, links para a comunidade de desenvolvedores e o código-fonte do PHP. Por ser uma linguagem de programação que é executada do lado do servidor, ela deve ser instalada junto a um servidor de páginas web. Se você for contratar um serviço de hospedagem de websites, deve verificar se o PHP está instalado no servidor e qual a versão.

### 5.2. - Servidor de Teste

Um servidor é um programa de computador que serve algo. Podemos ter um servidor de e-mail, que serve e-mails; um servidor de vídeos, que é quem envia o vídeo, como no YouTube, e também temos um servidor que serve as páginas web. A instalação do PHP deve ser feita em um servidor web já instalado.

Então, para testarmos nossas páginas PHP precisamos de uma empresa que forneça isso? Não. Podemos instalar um servidor em nosso computador junto com o PHP e testarmos nosso código PHP localmente. A isso damos o nome de servidor local. Atente para o fato de que nosso servidor é local, ou seja, as páginas apenas serão vistas em nosso computador.

Quando nossa programação estiver pronta, podemos enviá-la para o servidor que contratarmos para que seja disponibilizada mundialmente.

Parece muito complicado? Existe um pacote de software que facilita esse trabalho, seu nome é XAMPP. Ele é todo em português e existe uma versão *portable*, ou seja, leve e própria para um pendrive, por exemplo. A figura 6 reproduz a tela para download e logo abaixo dela está disponível o link para download, tanto da versão *portable* quanto da versão para seu sistema operacional.

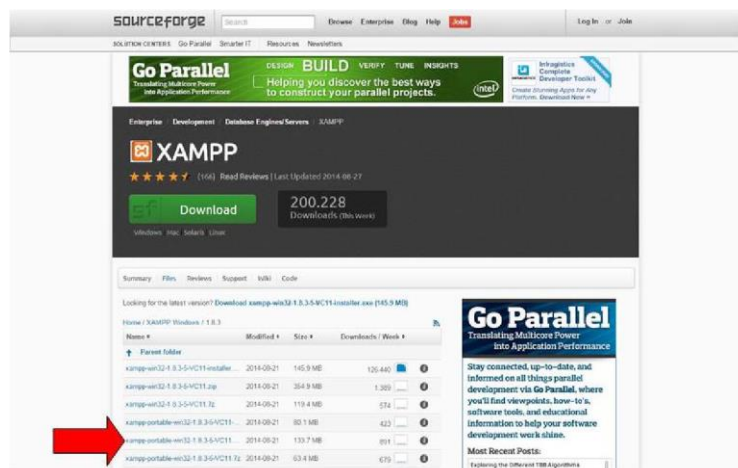


Figura 6 - Tela para download XAMPP. Fonte: Source Forge.

Link para baixar a versão para seu sistema operacional:

[https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html).

Link para baixar a versão *portable*:

<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.8.3/xampp-portable-win32-1.8.3-5-VC11.zip/download>

Descompacte o arquivo em um bom lugar, preferencialmente na raiz de seu sistema operacional, e memorize sua localização. Dentro há vários programas, entre eles o Apache, que é um servidor de páginas web com o PHP já instalado e o banco de dados web MySQL. Além disso, há ainda o phpMyAdmin, um aplicativo web feito com PHP para gerenciar os bancos do MySQL. Com ele você pode criar tabelas, campos, visualizar e gerenciar tudo no MySQL.

Antes de podermos testar o servidor, temos que configurar sua localização. Não se preocupe que tudo será muito fácil. Dentro da pasta que você descompactou, procure o arquivo **setup\_xampp.bat** e execute-o. O que ele faz é descobrir onde estão os servidores. Isso só é preciso porque você pode levar esta pasta no pen drive para outros lugares, assim, a localização pode mudar. Sempre que você mudar a pasta de lugar, vai ter que executar este arquivo para configurar a localização, mas, se não mudar de lugar não precisará repetir o processo.

Aparecerá a janela conforme ilustrado na Figura 7. Aguarde até aparecer uma mensagem dizendo que a atualização foi realizada. Pressione, então, qualquer tecla para que a janela desapareça.

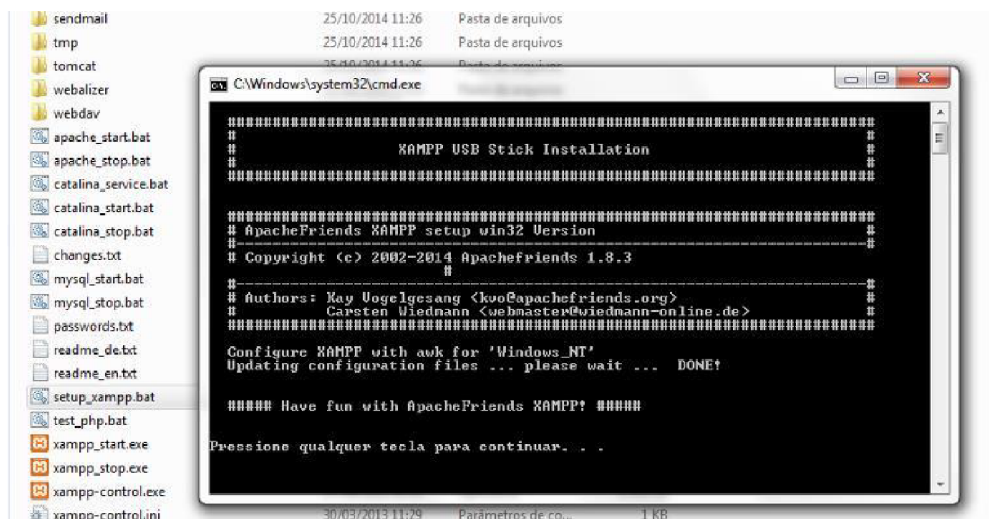


Figura 7 - Menu do setup\_xampp.bat. Fonte: Próprio Autor.

Uma vez configurado, vamos executar o arquivo **xampp-control.exe**. Ele é um programa que gerencia os servidores. Na primeira execução, ele pergunta qual o idioma que será utilizado, escolha a bandeira americana e pressione “Save”.

Com a tela do “Control Panel” aberta, pressione os botões “Start” para ligar o Apache e o MySQL. Aparecerá a tela de permissão do Windows, permita o acesso. Só é necessário fazer isso na primeira vez.

Espere algum tempo para os servidores começarem e você verá a tela da Figura 8, com os nomes dos aplicativos destacados em verde.

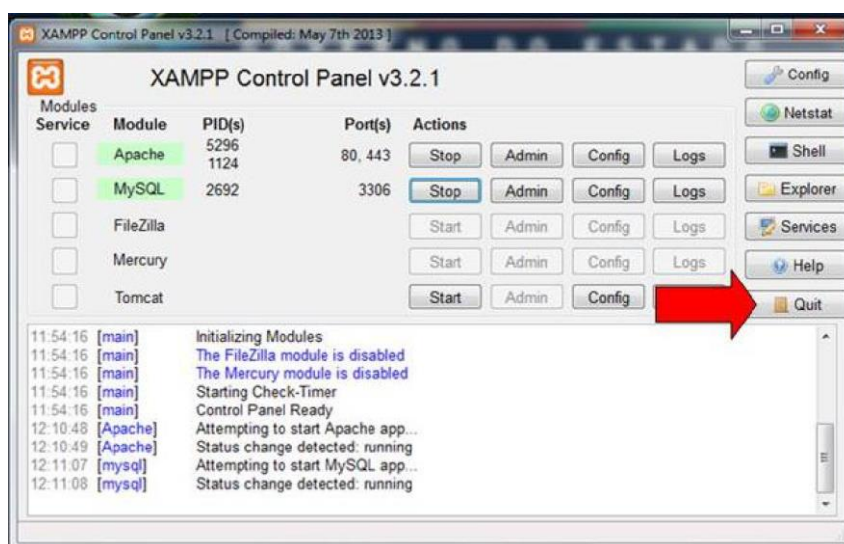


Figura 8 – Tudo OK. Fonte: Próprio Autor.

Agora que tudo está instalado, configurado e funcionando, vamos testar nosso ambiente de teste. Abra seu navegador preferido e digite o endereço **http://localhost/xampp/**. Deve abrir uma tela

como a da Figura 9. Selecione o idioma Português (Brasil) e aparecerá uma página similar à imagem da Figura 10. Esta página não está na internet, foi servida pelo seu servidor local.



Figura 9 – Página Inicial do XAMPP. Fonte: Próprio Autor



Figura 10 – XAMPP para Windows em Português. Fonte: Próprio Autor

### 5.3. – IDE

Uma IDE é um ambiente integrado de desenvolvimento, ou seja, um ambiente que reúne vários aplicativos que são necessários e úteis para o desenvolvimento. Existem diversas IDEs para desenvolvimento em PHP, algumas pagas, outras gratuitas. Entre elas está o **Net Beans**.

Antes de instalá-la, precisaremos instalar a máquina virtual Java, porque o Net Beans foi desenvolvido em Java. Clique no link a seguir e escolha a opção JRE:

<http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>

Aceite os termos de uso e clique na versão referente ao seu sistema operacional, conforme a Figura 11.

Java SE Runtime Environment 8u171		
You must accept the <a href="#">Oracle Binary Code License Agreement for Java SE</a> to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux x86	64.47 MB	<a href="#">jre-8u171-linux-i586.rpm</a>
Linux x86	80.38 MB	<a href="#">jre-8u171-linux-i586.tar.gz</a>
Linux x64	61.43 MB	<a href="#">jre-8u171-linux-x64.rpm</a>
Linux x64	77.41 MB	<a href="#">jre-8u171-linux-x64.tar.gz</a>
Mac OS X x64	74.58 MB	<a href="#">jre-8u171-macosx-x64.dmg</a>
Mac OS X x64	66.21 MB	<a href="#">jre-8u171-macosx-x64.tar.gz</a>
Solaris SPARC 64-bit	52.13 MB	<a href="#">jre-8u171-solaris-sparcv9.tar.gz</a>
Solaris x64	50.03 MB	<a href="#">jre-8u171-solaris-x64.tar.gz</a>
Windows x86 Online	1.79 MB	<a href="#">jre-8u171-windows-i586-iftw.exe</a>
Windows x86 Offline	61.66 MB	<a href="#">jre-8u171-windows-i586.exe</a>
Windows x86	64.84 MB	<a href="#">jre-8u171-windows-i586.tar.gz</a>
Windows x64	68.5 MB	<a href="#">jre-8u171-windows-x64.exe</a>
Windows x64	68.92 MB	<a href="#">jre-8u171-windows-x64.tar.gz</a>

Dê um duplo clique no arquivo que você baixou e o processo de instalação terá início. Aceite os próximos dois passos clicando em “Next”.

Logo em seguida, faça o download do Net Beans. A instalação é bem simples.  
[https://netbeans.org/downloads/start.html?platform=windows&lang=pt\\_BR&option=php&bits=x64](https://netbeans.org/downloads/start.html?platform=windows&lang=pt_BR&option=php&bits=x64)

## 5.4. - Testando o Ambiente

Agora que temos tudo pronto, vamos testar o nosso ambiente de desenvolvimento. Para isso, vamos fazer uma página simples em PHP, o nosso “Hello world!”.

No Net Beans, cada site é um novo projeto, por isso abra o menu Arquivo => Novo Projeto, será aberta a tela a seguir, onde você escolherá a opção PHP.

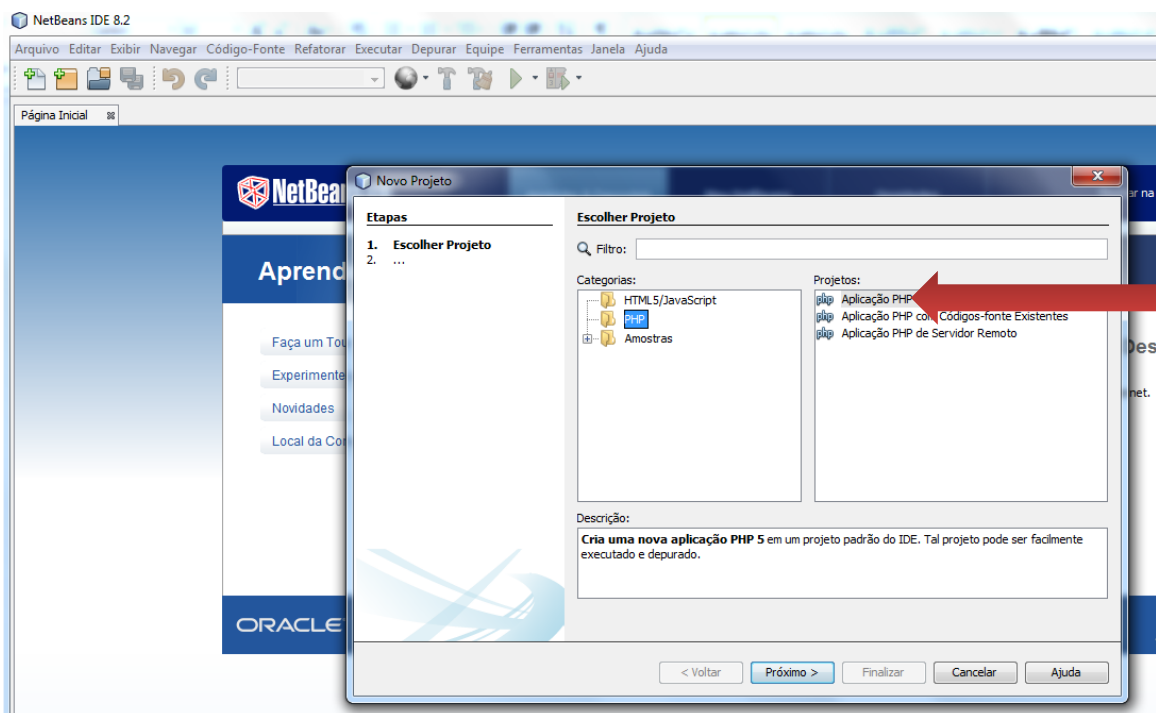


Figura 12 – Tela do Net Beans, escolhendo a categoria PHP. Fonte: Próprio Autor.

Clique no botão “Próximo”. Na tela a seguir, dê um nome ao projeto: “HelloMundo”. Em seguida, você define o seu Local Host, ou seja, o local onde estará sua página, sua URL local, seu endereço local. As duas próximas telas poderão ser desconsideradas neste momento, portanto, clique no botão “Finalizar”.

A tela do Net Beans será como a que segue:

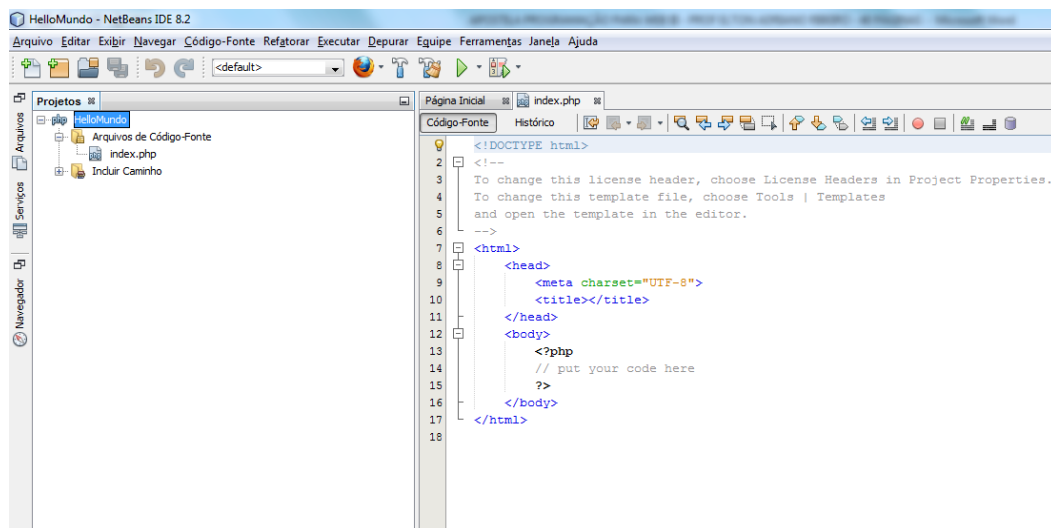


Figura 13 – Projeto HelloMundo, no Net Beans. Fonte: Próprio Autor.

Para criar nossa primeira página, digite o seguinte código:

```
<html>
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <?php
    echo "<p>Olá Mundo</p>"
  ?>
</body>
</html>
```

Figura 14 – Primeiro código PHP. Fonte: Próprio Autor.

Salve o arquivo.

Abra seu navegador e digite o seguinte endereço: <https://127.0.0.1/HelloMundo/>. Se tudo foi configurado corretamente, será exibida sua primeira página PHP. Conforme mostra a figura 15:

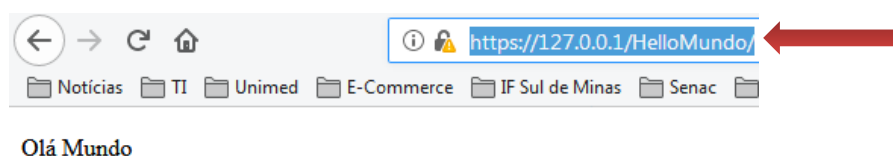


Figura 15 – Primeira página Web. Fonte: Próprio Autor.



## Aula 6 – Sintaxe Básica e Estudo das Variáveis

Objetivo: Conceituar Sintaxe Básica e conhecer as variáveis do PHP.

### 6.1 - Página PHP

O PHP é uma linguagem de programação que é executada do lado do servidor. Ela é diferente, por exemplo, do JavaScript, que também é uma linguagem interpretada, mas executada do lado do cliente, ou seja, no navegador do usuário e não no servidor.

Vamos ver o que acontece quando se usam páginas PHP em um site. Acompanhe o fluxo na Figura 16.

Primeiramente o usuário faz um pedido de uma página pelo endereço DNS, ou clicando em um link que possui um endereço DNS (1). Quando a solicitação chega ao servidor, ele verifica se o que foi pedido é uma página PHP e entrega para o interpretador (2). O interpretador lê a página e quando encontra um código PHP o executa (3). Quando termina de ler, o interpretador constrói uma página HTML como resultado da computação realizada e entrega ao servidor (4). Finalmente, o servidor envia a página HTML resultante para o usuário. Todas as vezes que são feitas requisições de página PHP, esse processo é repetido.

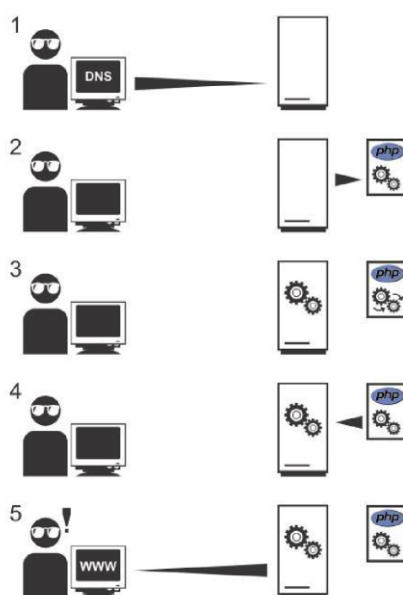


Figura 16 – Fluxo de execução páginas PHP. Fonte: Loop Infinito

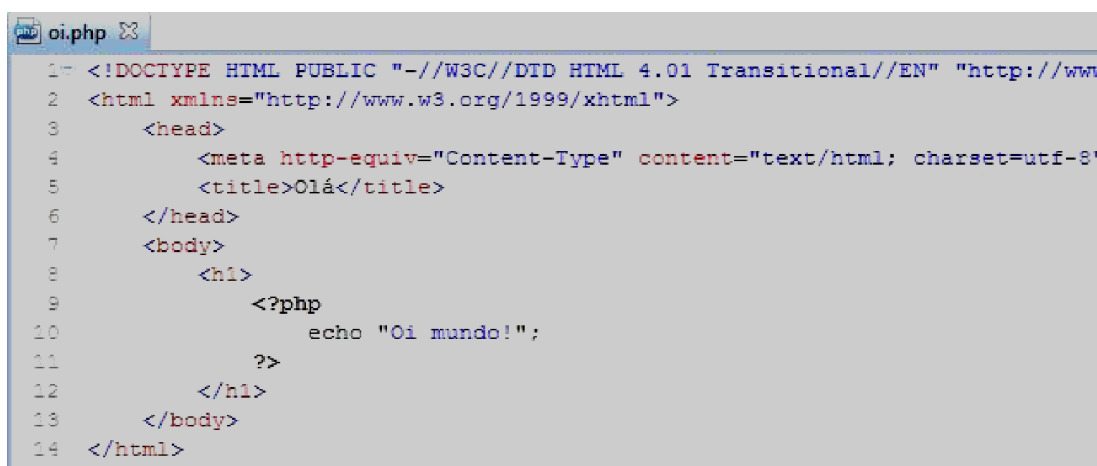
Mais adiante, observe que as páginas PHP são páginas HTML com código PHP misturado. Para que o servidor saiba que são páginas PHP, ao invés de colocarmos a extensão .html, colocamos .php. Podemos enviar para o servidor uma página PHP contendo apenas código HTML. O servidor vai ter todo o trabalho descrito na Figura 16 e irá retornar à mesma página, sem realizar qualquer computação. Quando formos fazer nosso sistema de locadora teremos casos como este.

## 6.2. - Sintaxe Básica

Para que uma página seja interpretada pelo PHP, ela deve ter a extensão **.php**, mas não só isso, todo código PHP deve estar entre dois sinais que indicam o início e o fim de um pedaço de computação. Assim, o interpretador vai saber que o que está dentro é comando PHP.

Então, para marcar o início utilizamos **<?php** e para marcar o final utilizamos **?>**.

O comando “echo” escreve o que está logo após, entre aspas duplas. Vamos escrever “Oi mundo!”, utilizando PHP, conforme demonstrado na Figura 17.

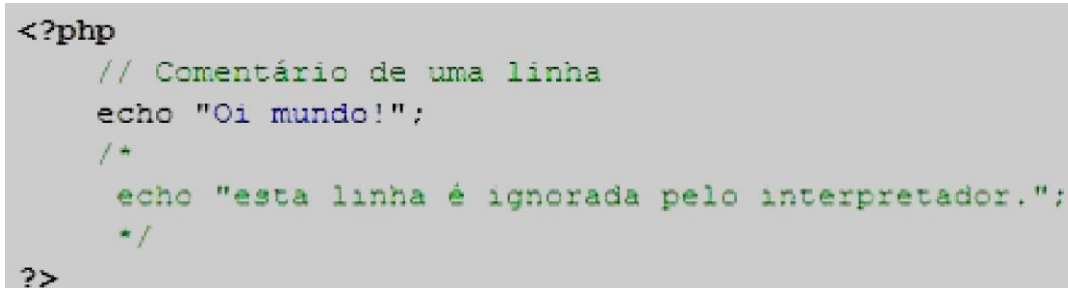


```
1- <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
2 <html xmlns="http://www.w3.org/1999/xhtml">
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
5     <title>Olá</title>
6   </head>
7   <body>
8     <h1>
9       <?php
10         echo "Oi mundo!";
11       ?>
12     </h1>
13   </body>
14 </html>
```

Figura 17 – Sintaxe básica em PHP. Fonte: Próprio Autor

Anteriormente, vimos como testar o ambiente, tenha certeza de que o XAMPP esteja ligado e faça o mesmo com o código acima.

Quando escrevemos sistemas, podemos chegar a uma quantidade enorme de códigos e esquecermos porque criamos um ou outro, para isso utilizamos os “comentários” em uma linha, colocando o // no final dela ou para um intervalo utilizamos /\* no começo e \*/ no final. Como pode ser visto na Figura 18.



```
<?php
    // Comentario de uma linha
    echo "Oi mundo!";
    /*
        echo "esta linha é ignorada pelo interpretador.";
    */
?>
```

Figura 18 – Comentários no código. Fonte: Próprio Autor

## 6.3. – Variáveis

De acordo com Miles (2008), variáveis são como “caixas” na memória do computador, onde armazenamos algo. Para trabalharmos com dados, precisamos armazenar seus valores em um lugar que seja fácil de encontrar quando necessitarmos. Essas “caixas” chamam-se variáveis. O conceito é o mesmo das variáveis matemáticas, onde utilizamos letras para representar um valor, por exemplo, Z ou Y.

Da mesma forma que na matemática, as variáveis em PHP podem ser utilizadas para guardar valores ( $X=2$ ) e expressões ( $Z=X+Y$ ).

### 6.3.1 - Nome das Variáveis

Ao invés de utilizarmos letras, podemos dar nomes às variáveis, por exemplo: sexo, cor, modelo, idade, etc. As variáveis devem obedecer às seguintes regras:

- Uma variável deve começar com o sinal de cifrão \$, seguido pelo nome da variável;
- O nome de uma variável deve começar com uma letra ou com os caracteres de sublinhado;
- O nome de uma variável só pode conter caracteres alfanuméricos e sublinhados (Az, 0-9 e \_);
- O nome de uma variável não deve conter espaços;
- PHP distingue maiúsculas de minúsculas, dessa forma \$nome, \$Nome, \$NOME e \$nOmE são variáveis diferentes.

### 6.3.2 – Declaração de Variáveis

Toda variável deve ter um valor atribuído a ela, para isso utilizamos o sinal de igual ( = ), por exemplo:

```
$nome = "IF Sul de Minas";  
$idade = 10;
```

Quando executarmos o programa, o computador guardará na memória o nome IF Sul de Minas e o número 10, para que usemos a qualquer momento.

### 6.3.3 – Escopo de uma variável

Não podemos acessar nossas variáveis em qualquer lugar do código. Elas pertencem a um determinado local, a que chamamos de escopo. Temos quatro tipos de escopo: Local, Global, Estático e Parâmetro.

Uma variável declarada dentro de uma função é **local** e só pode ser acessada dentro dessa função. Uma função é um pequeno programa, basicamente.

```
1 <?php
2 function teste() {
3     $x = 2 + 2; // A variável $x só existe localmente.
4 } // Quando a função termina a variável $x é apagada da memória.
5
6 echo $x; // Quando esta linha for lida a variável $x não existe.
7 ?>
```

Figura 19 – Exemplo de variável local. Fonte: Próprio Autor

Uma variável que é declarada fora de uma função tem um escopo mais abrangente, sendo classificada como **global**. Uma variável global pode ser acessada em qualquer lugar do programa, exceto dentro de uma função.

```
<?php
$x=2; // Escopo global
$y=5; // Escopo global

function teste() {
    global $x, $y; // Aqui chamamos as variáveis que estão fora da função.
    $y = $x + $y; // Aqui somamos o conteúdo de $x e $y e colocamos dentro de $y
}

teste();
echo $y; // Escreve 7
?>
```

Figura 20 – Escopo global. Fonte: Próprio Autor.

Quando o escopo de uma variável local ou global termina, ela é excluída, mas você pode querer que ela continue a existir por mais tempo. Sendo assim, utilize a palavra **static** quando declarar a variável.

```
<?php

function teste() {
    static $x=0; // Escopo estático
    echo $x; // Escreve seu valor
    $x++; // Soma mais um ao seu valor
}

teste(); // Aqui, quando chegasse no final, a variável $x era para ser excluída.
teste(); // Mas como ela foi declarada com static, não é apagada.
teste(); // E chamada novamente mostrando o valor acrescido.

?>
```

Figura 21 – Escopo estático. Fonte: Próprio Autor.

## 6.4. – Strings

O PHP disponibiliza várias funções para a manipulação de strings. As strings são os conteúdos existentes nas variáveis. Vamos abordar algumas das mais corriqueiras.

### 6.4.1 – Operação de Concatenação

Concatenar é unir, juntar. O operador de concatenação realiza uma operação de unir duas strings. Seu símbolo é o ponto e podemos unir duas ou mais strings na mesma expressão.

```
<?php
$txt1 = "Oi!";
$txt2 = "Fulando de Tal.";

echo $txt1 . " " . $txt2;
?>
```

Figura 22 – Exemplo de concatenação. Fonte: Próprio Autor.

Será escrito “Oi! Fulano de Tal”. Neste caso, concatenamos um espaço em branco no meio de duas variáveis.

### 6.4.2 – Função strlen ( )

A função strlen conta a quantidade de caracteres. Será escrito como saída, 9.

```
<?php
$txt1 = "Oi!";
$txt2 = "Fulando de Tal.";

echo $txt1 . " " . $txt2;
?>
```

Figura 23 – Exemplo da função strlen( ). Fonte: Próprio Autor

### 6.4.3 - Função strpos( )

Usada para procurar uma string dentro de uma string. Caso ache a função, retorna o número da posição do primeiro caractere da string procurada, caso não ache, retorna **falso**. O exemplo abaixo retornará o valor 10.

```
<?php
echo strpos("Fulano de Tal","Tal");
?>
```

Figura 24 – Exemplo de uso da função strpos( ) . Fonte: Próprio Autor.

## 6.5. – Operadores

Os operadores servem para executar uma operação em uma expressão, como as operações de soma, subtração, multiplicação e divisão. Em PHP temos os seguintes operadores: Operadores Aritméticos, Operadores de Atribuição, Operadores de Incremento e Decremento, Operadores de Comparação e Operadores Lógicos.

### 6.5.1 – Operadores Aritméticos

OPERADOR	NOME	DESCRIÇÃO	EXEMPLO	RESULTADO
$x + y$	Adição	Soma de x e y	$2 + 2$	4
$x - y$	Subtração	Subtração de x e y	$7 - 2$	5
$x * y$	Multiplicação	Produto de x por y	$3 * 2$	6
$x / y$	Divisão	Quociente de x por y	$4 / 2$	2
$x \% y$	Módulo	Resto da divisão de x por y	$5 \% 2$	1
$-x$	Negativo	Inverte o sinal de x	-2	2

Figura 24 – Operadores Aritméticos. Fonte: Próprio Autor

### 6.5.2 – Operadores de Atribuição

OPERADOR	COMO SE FOSSE...	DESCRIÇÃO
$x = y$	$x = y$	Coloca no operando da esquerda o resultado ou valor da direita
$x += y$	$x = x + y$	Adiciona y a x
$x -= y$	$x = x - y$	Subtrai y de x
$x *= y$	$x = x * y$	Multiplica x por y
$x /= y$	$x = x / y$	Divide x por y
$x \% = y$	$x = x \% y$	Coloca em x o resto da divisão de x por y
$x . = y$	$x = x . y$	Concatena duas strings

Figura 25 – Operadores de Atribuição. Fonte: Próprio Autor.

### 6.5.3 – Operadores de Incremento e Decremento

OPERADOR	NOME	DESCRIÇÃO
$++x$	pré-incremento	Soma um ao valor de x e depois retoma x
$x++$	pós-incremento	Retoma x e depois soma um ao valor de x
$--x$	pré-decremento	Subtrai um do valor de x e depois retoma x
$x--$	pós-decremento	Retoma x e depois subtrai um do valor de x

Figura 26 – Operadores de Incremento e Decremento. Fonte: Próprio Autor.

### 6.5.4 - Operadores de Comparação

OP.	NOME	DESCRIÇÃO	EX.	RES.
$x == y$	Igual	Retorna verdadeiro se for igual.	$2 == 3$	false
$x != y$	Não igual	Retorna verdadeiro se for diferente.	$2 != 3$	true
$x <> y$	Diferente	Retorna verdadeiro se for diferente.	$2 <> 3$	true
$x > y$	Maior que	Retorna verdadeiro se x for maior que y	$3 > 2$	true
$x < y$	Menor que	Retorna verdadeiro se x for menor que y	$2 < 3$	true
$x >= y$	Maior que ou igual	Retorna verdadeiro se x for maior que ou igual a y	$3 >= 3$	true
$x <= y$	Menor que ou igual	Retorna verdadeiro se x for menor que ou igual a y	$2 <= 3$	true

Figura 27 – Operadores de Comparação. Fonte: Próprio Autor.

### 6.5.5 - Operadores Lógicos

OPERADOR	NOME	DESCRIÇÃO	EXEMPLO	RESULTADO
$x \text{ and } y$	e	Retorna verdadeiro se x e y forem verdadeiro	$x=1; y=9$ $(x < 3 \text{ and } y > 5)$	true
$x \text{ or } y$	ou	Retorna verdadeiro se houver um verdadeiro	$x=1; y=9$ $(x == 1 \text{ or } y == 2)$	true
$x \&\& y$	e	Retorna verdadeiro se x e y forem verdadeiro	$x=1; y=9$ $(x < 3 \&\& y > 5)$	true
$x \parallel y$	ou	Retorna verdadeiro se houver um verdadeiro	$x=1; y=9$ $(x == 1 \parallel y == 2)$	true
$!x$	não	Retorna verdadeiro se x for falso	$x = 1, y = 9$ $!(x == y)$	true

Figura 28 – Operadores Lógicos. Fonte: Próprio Autor.

### 6.6. – If...Else

Linguagens de programação têm duas estruturas em comum: estruturas de repetição e estruturas de condicionais. O “if...else” é uma estrutura condicional. Ela verifica uma condição, se for verdadeira faz algo, senão faz outra coisa.

If (condição) comando;

If (condição) {

// Bloco de código. Aqui pode ter várias linhas.

}

Nos casos acima, o “if” verifica a condição, se for verdade ela executa um comando. Caso queira mais de um comando, você deve criar um bloco de comando utilizando { e }. Caso a condição seja falsa, o comando ou bloco será ignorado. O “else” significa “senão”. Utilize-o para colocar um comando ou bloco de comandos, caso a condição seja falsa. Conforme mostra a figura 29.



```

<?php
$x = 10;
$y = 100;
if ($x < $y) echo "<p>x é menor do que y<p>";

if ($y > $x) {
    echo "<p>y é maior do que x, ";
    echo "o valor de y é " . $y . "</p>";
} else {
    echo "x é maior que y";
}
?>

```

Figura 29 – Estrutura Condicional. Fonte: Próprio Autor

O exemplo acima escreve “**x é menor do que y**”, passa à próxima linha e escreve “**y é maior do que x, o valor de y é 100**”.

## 6.7. – Switch

É uma estrutura de condição, assim como o if...else, que verifica uma condição com várias possibilidades de resposta.

switch (n) {

case x:

código a ser executado se n igual a x;

break;

case y:

código a ser executado se n igual a y;

break;

...

default:

código a ser executado se n não for nenhuma das opções anteriores.

}

```

<?php
$musica = "rock";
switch ($musica) {
    case "sertanejo":
        echo "Adoro sertanejo!";
        break;
    case "rock":
        echo "Rock'n Roll, baby!";
        break;
    case "samba":
        echo "Samba é o que tem de melhor!";
        break;
    default:
        echo "Não gosto nem de sertanejo, nem de rock, nem de samba, prefiro funk!";
}
?>

```

Figura 30 – Exemplo de estrutura switch. Fonte: Próprio Autor.

O exemplo acima retorna “**Rock’n Roll, baby!**”. Você pode mudar o valor da variável **\$musica** para ver os outros resultados. Também pode colocar um valor que não exista nas opções, para exibir a opção **default**.

## Aula 7 – Estruturas de Repetição

Objetivo: Conceituar Estruturas de Repetição

### 7.1. – Estruturas de Repetição

Quando programamos em PHP precisamos repetir o mesmo código para que ele seja executado várias vezes. As estruturas de repetição foram criadas para fazer este serviço para nós, sem que precisemos repetir o código. Em PHP temos quatro estruturas de repetição:

- **while** - repete o bloco de código enquanto uma condição especificada for verdadeira;
- **do...while** - repete o bloco de código uma vez, depois repete novamente enquanto a condição especificada for verdadeira;
- **for** - repete o bloco de código um determinado número de vezes;
- **foreach** - repete o bloco de código para cada elemento de um array.

#### 7.1.1 – While

Repete o bloco de código enquanto uma condição especificada for verdadeira. O exemplo da Figura 31 define a variável \$i inicialmente com o valor de 1 (\$i = 1). Em seguida, o loop **while** continuará a ser executado enquanto i for menor ou igual a 5 (\$i <= 5). A variável i vai aumentar em 1 a cada vez que o loop é executado (\$i++;)

```
<html>
<body>

<?php
$i=1; // $i começa com o valor de 1
while($i<=5) { // repete enquanto for menor que ou igual a 5
    echo "The number is " . $i . "<br>";
    $i++; // soma 1 ao valor de $i
}
?>

</body>
</html>
```

Figura 31 – Loop while. Fonte: Próprio Autor.

#### 7.1.2 - Do...While

Esta estrutura repete o bloco de código uma vez e, depois, repete novamente enquanto a condição especificada for verdadeira. O exemplo da Figura 32 define uma variável \$i inicialmente com o valor de 1 (\$i = 1;). Então, ele começa o do...while. O ciclo irá somar a variável i com 1 e, em seguida,

escrever alguma saída. Em seguida, a condição é verificada (i é menor ou igual a 5), se for, o bloco de código é repetido. Se não for, a repetição é interrompida e a próxima linha depois do do...while é lida.

```
<html>
<body>
|
<?php
$i=1; // $i inicia com o valor de 1
do { // executa o bloco
    $i++;
    echo "The number is " . $i . "<br>";
} while ($i<=5); // Se $i for menor ou igual a 5, repete novamente
?>

</body>
</html>
```

Figura 32 – Estrutura do...while. Fonte: Próprio Autor

### 7.1.3 – For

Usada quando se sabe com antecedência o número de repetições. O exemplo da Figura 33 define um ciclo que começa com \$i = 1. A repetição continuará a funcionar enquanto a variável \$i for menor que ou igual a 5. A variável \$i vai aumentar em 1 a cada vez que o loop é executado.

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++) {
    echo "O número é " . $i . "<br>";
}
?>

</body>
</html>
```

Figura 33 – Estrutura de repetição for. Fonte: Próprio Autor

## Aula 8 – Funções em PHP

Objetivo: Conceituar e conhecer as funções referentes ao PHP.

### 8.1 – Funções

Funções são pequenos programas que ajudam sua aplicação em uma tarefa. Vamos supor que você tenha uma aplicação que calcula uma nota fiscal, mas, para dar o total da nota você tem que fazer outro cálculo, o do frete. Esse cálculo do frete não faz parte do cálculo da nota fiscal, porque ele pode existir ou não. Além disso, o usuário pode tentar outros lugares de entrega para baratear o frete. Podemos criar uma função que faça só esse cálculo. Usando funções, o código fica mais simples e de fácil manutenção. Podemos até utilizar funções de outras pessoas, sem nem mesmo saber como foram programadas.

Uma função precisa de um nome e de um bloco de código que será executado quando chamamos a função pelo seu nome. Coloque um nome que reflita o que função irá fazer. A sintaxe da função consiste em:

```
function nomeDaFuncao() {  
    bloco de código;  
}
```

### 8.2 – Adicionando Parâmetros

Parâmetros são como variáveis que recebem um valor quando a função é executada. Normalmente esses parâmetros são necessários para a realização da tarefa. Por exemplo, uma função que soma dois números. Para poder somar, ela vai precisar dos dois números que são os parâmetros.

```
<html>  
<body>  
  
<?php  
// para trabalho a função precisa receber dois valores  
// o primeiro vai para $nome, o segundo para $sobrenome  
function apresentacao($nome, $sobrenome) {  
    echo "Oi, " . $nome . " " . $sobrenome;  
}  
  
apresentacao("Fulano", "de Tal"); // na chamada da função passamos os valores.  
apresentacao("Beltrano", "de Tal"); // observe a ordem, é importante.  
?  
  
</body>  
</html>
```

Figura 34 - Passagem de parâmetros para funções. Fonte: Próprio Autor

## 8.3 – Retornando Valores

Podemos retornar um valor computado pela função utilizando a palavra-chave **return**. Este comando retorna o que estiver à direita e encerra a função. A Figura 35 mostra um exemplo de retorno de função. A função calcula a soma de dois números, \$x e \$y. Ela deve escrever uma página com “2+2=4”.

```
<html>
<body>

<?php
function soma($x, $y) {
    $total=$x+$y;
    return $total; // aqui o valor de $total é retornado
}

echo "2 + 2 = " . soma(2, 2); // soma(2, 2) é substituído pelo retorno da função
?>

</body>
</html>
```

Figura 35 – Retorno de uma função. Fonte: Próprio Autor

## 8.4 – Formulários

Para poder receber informações do lado do cliente, ou seja, do usuário, utilizamos hiperlinks ou formulários. Recebemos informação do usuário quando ele clica em um hiperlink, chamado de método GET, ou quando nos envia um formulário, chamado de método POST.

### 8.4.1 – Método GET

Este método foi uma das primeiras formas de comunicação cliente/servidor via web, ele utiliza a URL do site para enviar dados. Não há necessidade de um formulário HTML, no entanto, é visível para todos. Digitar senhas utilizando esse método é inviável.

Como é realizado o envio de dados por GET:

[www.meusite.com.br/index.php?nome=fulano&idade=30](http://www.meusite.com.br/index.php?nome=fulano&idade=30). Depois da interrogação são enviadas duas variáveis: nome=fulano, idade=30. O & separa uma variável da outra. Na Figura 36, a página PHP recebe os dados e escreve uma página de resposta.

```

<html>
<body>

<?php
$usuario = $_GET["nome"]; //o parâmetro deve ser igual ao enviado.

echo "<h1>Oi! " . $usuario . "</h1>";
?>

</body>
</html>

```

Figura 36 – Página recebe os dados enviados e os processa. Fonte: Próprio Autor.

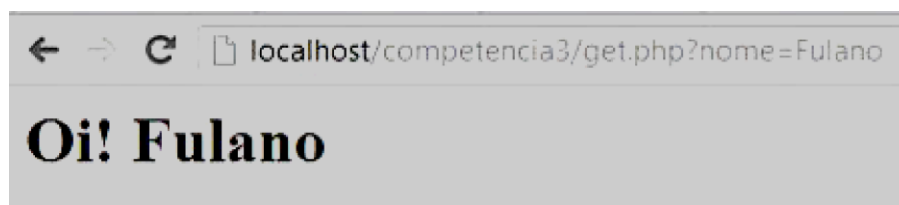


Figura 37 – O endereço da página contém o nome. Fonte: Próprio Autor.

#### 8.4.2 – Método POST

O método POST utiliza um formulário HTML para enviar a informação. Veja no exemplo da Figura 38.

```

<html>
<body>

<?php
$usuario = $_POST["nome"]; //o parâmetro deve ser igual ao enviado.
$tempo = $_POST["idade"];

echo "<p>Oi! " . $usuario . "<br />";
echo "Voce tem " . $tempo . " anos.</p>";
?>

</body>
</html>

```

Figura 38 – Página PHP recebe dados de um formulário HTML. Fonte: Próprio autor.

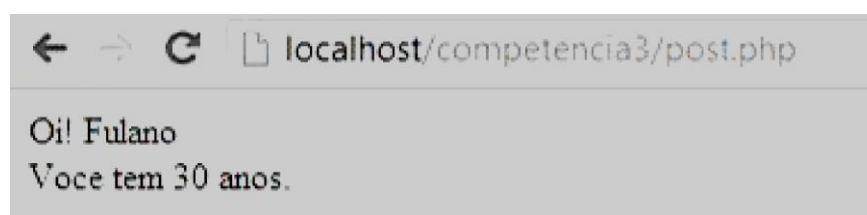


Figura 39 – Página de Saída. A URL não contém dados. Fonte: Próprio Autor.



## Aula 9 - Introdução à Programação em Linguagem Orientada a Objetos

Objetivo: Conceituar Linguagem Orientada a Objetos.

### 9.1. - Programação Orientada a Objetos (POO)

O desenvolvedor novato percebe que muitas das ferramentas e linguagens atuais (PHP, Java, Ruby, Python, entre outras) são linguagens que utilizam o paradigma orientado a objetos. A princípio isso pode parecer ruim - “mais uma coisa para aprender”, mas, na verdade é o que vai permitir ao desenvolvedor se aprimorar.

Sabendo POO, o desenvolvedor poderá resolver os mesmos problemas utilizando não somente o passo-a-passo tradicional, mas também uma modelagem do problema de uma forma mais natural/real.

Orientação de objetos (OO), em uma definição formal, é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas “objetos”. Ou seja, é um modelo utilizado no desenvolvimento de software em que trabalhamos com unidades chamadas *objetos*.

De acordo com Santos (2003), a Programação Orientada a Objetos (POO) foi criada para tentar aproximar o mundo real e o mundo virtual, a ideia fundamental é tentar simular o mundo real dentro do computador. Para isso, nada mais natural do que utilizar objetos, afinal, nosso mundo é composto de objetos, certo?

Na Programação Orientada a Objetos, o programador (você) é responsável por moldar o mundo dos objetos e definir como os objetos devem interagir entre si.

Os objetos “conversam” uns com os outros através do envio de mensagens e o papel principal do programador é definir quais serão as mensagens que cada objeto pode receber e também qual a ação que o objeto deve realizar ao receber cada mensagem.

Isso possibilita a criação de códigos com baixo acoplamento e que podem ser facilmente reutilizados, o que são alguns dos principais motivos para se programar orientado a objetos.

Vejamos o exemplo para trocar uma lâmpada, agora usando a Programação Orientada a Objetos (POO). A primeira coisa a fazer é pensarmos em classes. Em outras palavras, coisas envolvidas no estudo de caso (trocar uma lâmpada):

- Pessoa, Lâmpada, Escada, Bocal da Lâmpada, Interruptor.

Definido isso, podemos combinar suas características e ações. Por exemplo: “uma Pessoa pega uma Lâmpada boa”.

Apenas esse fato de dizer que a “Pessoa” “pega” “Lâmpada” “boa” já nos diz muita coisa, pois tudo isso pode ser modelado usando o paradigma de Orientação a Objetos. Em outras palavras, tudo não passa de modelar objetos que possuem ações e características.

Em nosso caso, “Pessoa” seria uma classe/objeto, “pega” seria uma ação da “Pessoa”, “Lâmpada” seria outra classe/objeto e “boa” seria o estado/característica/atributo da “Lâmpada”.

E por que estudar POO? Irei te lançar um desafio...

- Acesse o [ranking da TIOBE](#), um dos rankings de linguagens de programação mais conceituados e confiáveis;
- Confira as linguagens que ocupam as primeiras posições do ranking;
- Agora acesse o [Wikipédia](#) e procure características sobre essa linguagem.

Eis o que em 90% dos casos você irá encontrar: “Essa linguagem de programação é baseada no paradigma.

## Referências

BERT; SIERRA, Kathy. **Use a cabeça!**. Rio de Janeiro: Alta Books, 2009.

BOAVIDA, F.; BERNARDES, M. TCP/IP: **Teoria e Prática**. Lisboa: FCA. 2012. ISBN 978-97-2722-745-7.

MILES, Pilone; MILES, Russ. **Use a cabeça! Desenvolvimento de software**. Rio de Janeiro: Alta Books, 2008.

SANTOS, Rafael. **Introdução à programação orientada a objetos**. Rio de Janeiro: Elsevier, 2003.

SEGARAN, T. **Programming Collective Intelligence: Building Smart Web 2.0 Applications**. Sebastopol, USA: O'Reilly Media. 2007.



·rede  
**e-Tec**  
Brasil

**MEDI**  **tec**