Profº Elton Rodrigo

Esta apostila foi criada e editada com base em informações contidas nos sites **imasters.com.br**, **caelum.com.br**, **linhadecodigo.com.br**, **infowester.com, aprenderphp.com.br**, **oficinadanet.com.br, revistaphp.com.br, php.net, diegomacedo.com.br** e **devmedia.com.br.**

As imagens nela contidas foram capturadas com *PrintScreen* de computadores associados.

Sumário

1. Conceitos Básicos de PHP

1.1. O Que é PHP?

PHP (um acrónimo recursivo para "PHP: Hypertext Preprocessor") é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na Web. Apesar de ser uma linguagem de fácil aprendizado e de uso para pequenos scripts dinâmicos simples, o PHP é uma linguagem poderosa orientada a objetos.

1.2. Histórico

A linguagem surgiu por volta de 1994, como um subconjunto de scripts Perl criados por Rasmus Lerdof. Com as adições de Zeev Suraski e Andi Gutmans, dois programadores israelitas pertencentes ao Technion, o Instituto Israelita de Tecnologia, que reescreveram o parser, era lançada em 1997 a PHP 3, primeira versão estável e parecida com a linguagem atual. Ao reescrever o parser, foi criado o Zend Engine, que é mantido oficialmente pela empresa Zend em conjunto com a comunidade PHP. Em Maio de 2000 veio a público a versão 4, e em Julho de 2004, a versão 5, onde a principal mudança foi uma nova API para orientação a objetos provida pelo Zend Engine 2.

Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. Diversos módulos são criados no repositório de extensões PECL (PHP Extension Community Library) e alguns destes módulos são introduzidos como padrão em novas versões da linguagem. É muito parecida, em tipos de dados, sintaxe e mesmo funções, com a linguagem C e com a C++. Pode ser, dependendo da configuração do servidor, embutida no código HTML. Além disso, destaca-se a extrema facilidade com que PHP lida com servidores de base de dados, como MySQL, PostgreSQL, Microsoft SQL Server, Oracle entre outros.

Existem versões do PHP disponíveis para os seguintes sistemas operacionais: Windows, Linux, Mac OS, OS/2, AS/400, Novell Netware, RISC OS, IRIX e Solaris. A Wikipédia funciona sobre um software inteiramente escrito em PHP, usando bases de dados MySQL: o MediaWiki.

Uma das grandes vantagens do PHP é que ele é distribuído gratuitamente através do site http://www.php.net. No site oficial do PHP você encontrará sempre as versões mais recentes para downloads. Além de gratuito, o seu código-fonte é aberto, e por fim você encontra toda a documentação do software também no site.

O PHP é baseado no servidor, ou seja, quando você executa uma página PHP no seu navegador, todo o código PHP é executado no servidor de origem da página, ou seja, o navegador apenas exibe a página processada.

2. Instalando o PHP no Windows

Com o avanço da tecnologia não dá para ficar parado no HTML básico. Sites dinâmicos, que acessam banco de dados (como fóruns, cadastros, etc), precisam de uma solução diferente do mero HTML. Entra aí o **PHP**, normalmente rodando no servidor Apache.

A hospedagem PHP é quase sempre mais barata (do que ASP, ColdFusion, entre outros concorrentes do PHP) e, pela linguagem ser aberta, existem muitos módulos para PHP, permitindo uma diversidade muito maior do que com o ASP ou ASP.Net, da Microsoft. Mais barata mas de qualidade, muitas vezes superior. A quantidade de scripts prontos em PHP também é enorme.

Dá pra fazer praticamente de tudo com PHP, como lojas virtuais, sites de relacionamentos, interfaces para webmail, chats, fóruns em geral, gerenciadores de conteúdos para sites para agilizar a publicação, etc. Para fazer funcionar um site em PHP, normalmente usa-se a combinação LAMP: Linux + Apache + MySQL + PHP. Os servidores que oferecem suporte ao PHP quase sempre rodam Linux, mas o webmaster não precisa conhecer o sistema. Basta saber programar e carregar as páginas, usando uma interface web (gerenciador de arquivos) ou, mais profissionalmente, FTP.

Pois bem, mas como testar os sites, localmente? Não basta dar um duplo clique nos *arquivos.php*, como se faz com os .html. É necessário ter um servidor web configurado para isso. Veremos então como rodar PHP no Windows (criando o "WAMP", Windows + Apache + MySQL + PHP ;).

Uma maneira simples de instalar esses softwares é através do **Easy PHP** que em apenas em poucos passos instala o servidor Apache, o módulo para programação em PHP e o banco de dados MySQL.

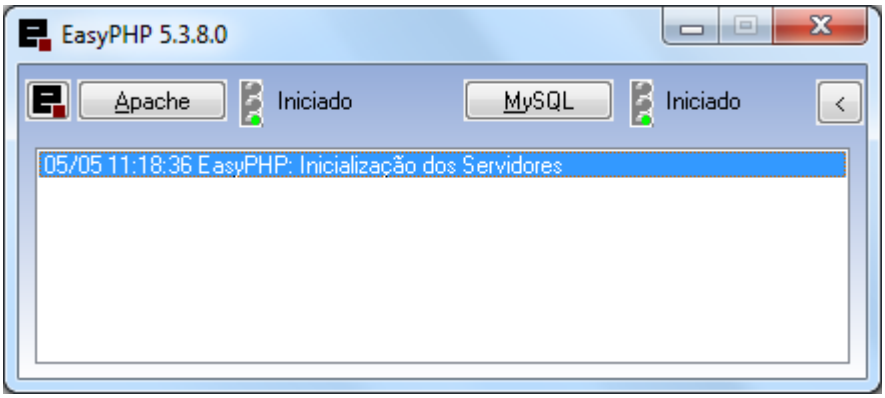
Local para download na Web: http://www.easyphp.org/

Este é um programa muito interessante para as pessoas que querem, no menor tempo possível, trabalhar com PHP sobre Windows, sem ter que passar pelas dificuldades de instalar e configurar todos os servidores e módulos necessários para utilizar esta linguagem de criação de páginas do lado do servidor.

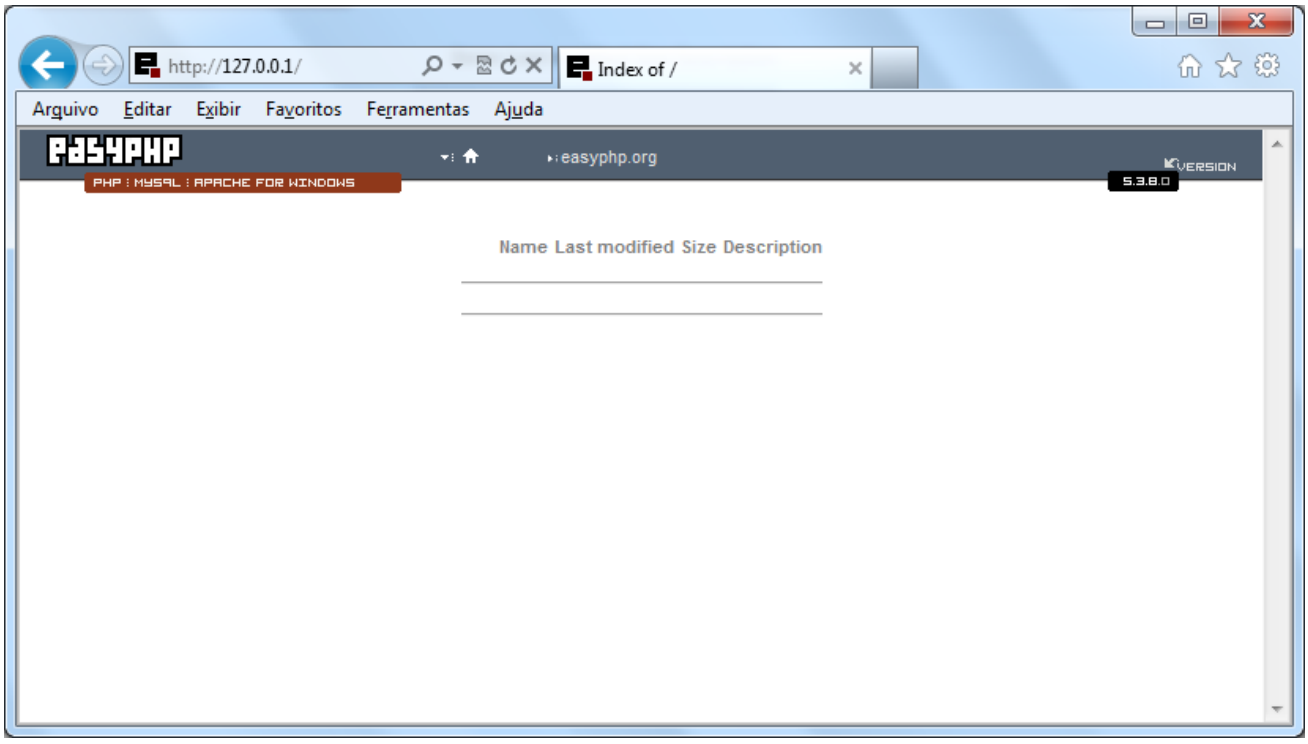
- O Easy PHP não é só um programa, são três em um:
- Apache, o servidor mais popular de páginas web.
- MySQL, o banco dados mais difundido de código livre.
- PHP, a linguagem ou tecnologia mais difundida para realizar páginas com programação no servidor, acesso ao banco de dados, etc.

2.1 - Instalação do EasyPHP

Uma vez baixado, a instalação é imediata. Para iniciar, abriremos Easy PHP, se já não o tivermos aberto. O Resultado será a abertura de uma janela como a figura abaixo, desde que, os servidores PHP e MySQL tenham sido iniciados corretamente.

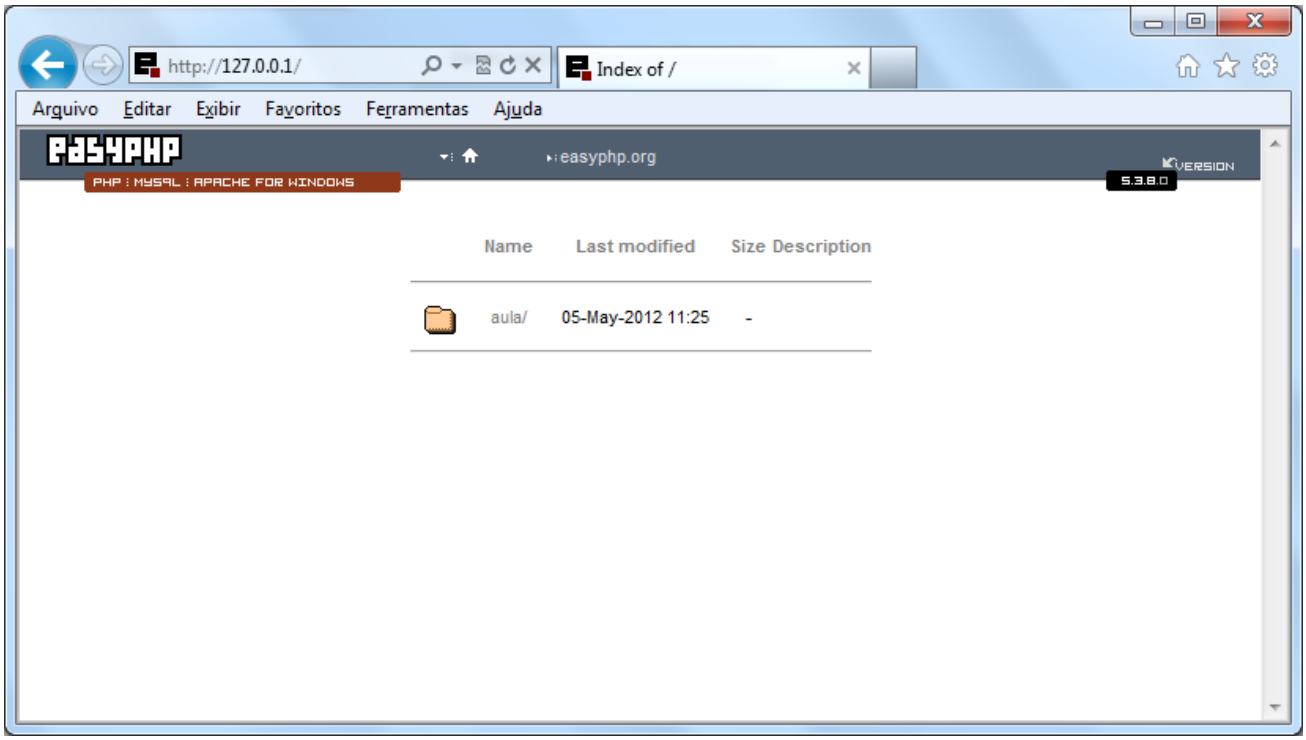


Quando o **Apache** e o **MySql** estiverem iniciados clique, com o botão direito do mouse, no ícone do **EasyPHP** na barra de tarefas e escolha a opção **Localhost**, ou digite diretamente no navegador o endereço **http://localhost/**, nesse momento deverá aparecer a página de início do EasyPHP como mostrado na figura abaixo.



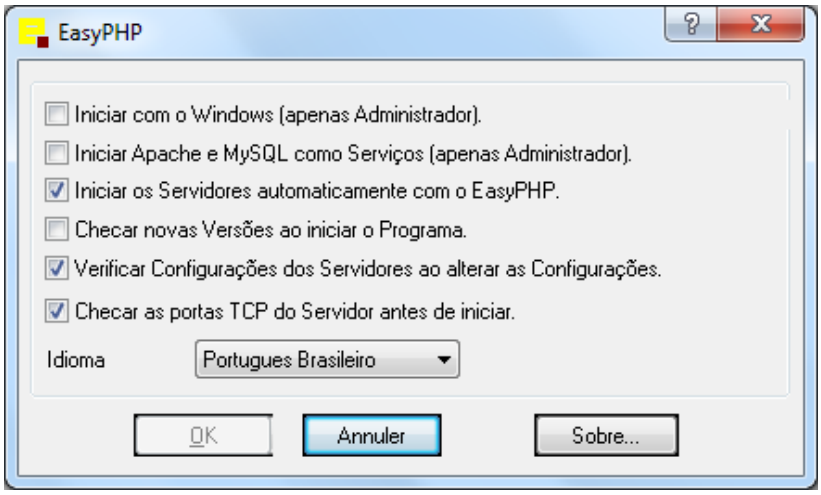
Os documentos web serão armazenados na pasta de publicação de documentos do EasyPHP "**C:\Arquivos de programa\EasyPHP-5.3.8.0\www**". Crie, então, uma pasta chamada **aula** dentro

da qual serão gravados nossos primeiros scripts em PHP. Se acessarmos a página local, agora, constataremos a presença da pasta **aula** como mostrado na figura abaixo.



2.2 - Como configurar o EasyPHP

Na **Barra de Tarefas** clique no ícone do **EasyPHP**, e aponte para configuração – **EasyPHP**. Na janela que surge, mostrada na figura abaixo, poderemos alterar qualquer das opções disponibilizadas:



- **Iniciar com o Windows (apenas administrador)**: como diz a própria opção, estando selecionada (ticada), o EasyPHP será iniciado juntamente com o Windows.

- **Iniciar o Apache e o MySQL como serviços (apenas administrador)**: com esta opção selecionada os servidores **Apache** e **MySQL** são iniciados como serviços mesmo que esteja logado na máquina, outro usuário, que não o administrador.
- **Iniciar o servidor automaticamente com o EasyPHP:** esta opção indica que quando o **EasyPHP** for ativado os servidores serão iniciados. Esta opção é padrão.
- **Checar novas Versões ao iniciar o Programa:** esta opção selecionada indica que toda a vez que o **EasyPHP** for iniciado será feita uma busca no site do projeto do EasyPHP, e surgirá uma mensagem informando se há nova versão disponível.
- **Verificar Configurações dos Servidores ao alterar as Configurações:** Indica se as configurações dos servidores sofreram alterações, os mesmos serão reiniciados.
- **Checar as portas TCP do Servidor antes de iniciar**: opção marcada por padrão.

O programa não termina por aqui, ainda podemos instalar alguns complementos ideais para começar a trabalhar com **PHP** e banco de dados **MySQL**, como **PhpMyAdmin**, um administrador de banco de dados bastante conhecido. Podemos encontrá-lo no endereço http://localhost/mysql/

O acesso a **PhpMyAdmin** fica bloqueado a outros computadores da rede local, visto que, ele é instalado como um diretório virtual do Apache e com acesso restringido à rede local. Contudo, consultando as FAQs (perguntas mais frequentes) poderemos aprender a permitir que outras máquinas da rede o venham acessar.

3. Testando o PHP

Para criar e editar scripts, em PHP, podemos utilizar qualquer editor de Html, ou até mesmo o bloco de notas do Windows. Vamos inicialmente utilizar o bloco de notas, depois optaremos por outro editor PHP.

3.1. Entendendo o código

Um código php pode conter, ou não, **tags Html**, essas tags não são processadas pelo servidor, são simplesmente passadas ao browser solicitante. Normalmente utiliza-se **Html** para fazer a **parte estática** da página, sua estrutura e o **php** para a **parte lógica**, que exige processamento. "Deve-se salvar os códigos em PHP com extensão "**.php**".

Há quatro conjuntos de tags que podem ser usadas para marcar blocos de código PHP. Delas, somente duas (`<?php. . .?>` e `<script language="php">. . .</script>`) são sempre disponíveis. As outras podem ser ativadas ou desativadas a partir do arquivo de configuração **php.ini**.

O interpretador identifica quando um código é PHP pelas seguintes tags:

```
<?php
 comandos
?>

<script language="php">
 comandos
</script>

<?
 comandos
?>

<%
 comandos
%>
```

Para cada fim de linha de código tem que haver um ponto e vírgula, indicando ao sistema fim de instrução.

Exemplo.

```
<?php
      echo 'com ponto e vírgula' ;
?>
```

Linhas de comando, de controle, não precisam de ponto e vírgula.

Exemplo.:

```
<?php
 if ($x == $x){ //aqui não precisa de ponto e vírgula
       echo 'com ponto e vírgula' ; //aqui precisa de ponto e vírgula
}
?>
```

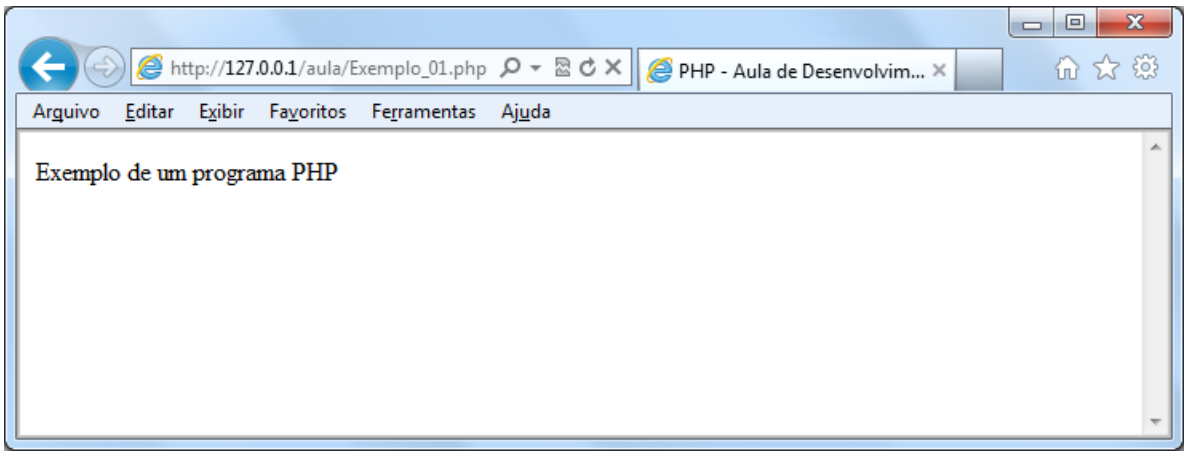
3.2 - Primeiro Exemplo em PHP

1) Abra o Bloco de Notas, do Windows, e digite o código como mostrado na figura abaixo.

```
<html>
<head>

<title> PHP - Aula de Desenvolvimento de Software I </title>

</head>

<body>
        <?php
             echo "Exemplo de um programa PHP";
        ?>
</body>

</html>
```

2) Salve o arquivo como **Exemplo_01.php** na pasta **"C:\Arquivos de programa\EasyPHP-5.3.8.0\www\aula".**

3) Para executarmos o arquivo criado devemos abrir o navegador de Internet, instalado na máquina, e digitar o endereço: http://localhost/aula/Exemplo_01.php. O resultado deve ser parecido com o mostrado na figura abaixo.



O comando `echo` retorna, o que está entre aspas, na forma de uma **string**, e disponibiliza, no formato **html**. Esta string pode ser passada diretamente (como no exemplo) para o corpo do navegador, ou para uma variável que poderá ser utilizada mais tarde no processamento das informações.

O processo de interpretação, de um arquivo, pelo PHP inicia-se quando uma das **tags especiais** é encontrada no conteúdo (conjunto de códigos) do arquivo. O interpretador passa, então, a executar os comandos e só para quando encontra uma **tag de fechamento do PHP**. Este mecanismo permite, portanto, que o **código PHP** seja inserido em um documento **escrito em HTML**. Logo, tudo o que estiver fora das **tags PHP** é exibido, no navegador, exatamente como foi digitado.

3.3 - Separador de Instruções e Variáveis

As instruções, no PHP, são separadas da mesma forma que muitas linguagens, ou seja: cada instrução termina com um ponto e vírgula (;). A tag (?>) implica no fechamento de uma instrução, porém ela não vem seguida de (;).

Exemplo:

```
echo "Exemplo de um programa em PHP";
echo "desenvolvido para iniciantes";
```

3.3.1 - Variáveis

As variáveis, em PHP, não precisam ser declaradas, mas devem sempre ter "$" na frente do nome. Por sua vez os **nomes** devem sempre começar com **letras** ou com o caractere "_".

Exemplo: `$nome` ou `$_CPF`

Observação: O PHP é *case sensitive,* portanto `$nome` é diferente de `$Nome` ou `$NOME`**.**

Em uma variável podemos armazenar os mais diversos tipos de dados.

String:

Uma string é um conjunto de caracteres de qualquer tipo. É o "vale tudo" da programação. Qualquer coisa entra ali. Pode colocar letra, número, símbolo, enfim, aceita tudo. Por exemplo, "Aprendendo PHP".

Inteiro:

Um inteiro é essencialmente um número, pode ser positivo ou negativo. Inteiro é representado pelo tipo "integer" em php (e na grande maioria das linguagens). Um integer tem 8 bits, portanto tem um limite mínimo e um máximo. Mas, para as coisas mais triviais integer dá conta do recado.

Float ou dooble

Este é um outro tipo de variável que só aceita número. Mas, diferentemente do tipo inteiro, aqui podemos colocar números com casas decimais. Como em integer, este tipo também tem limites, mas vai ser bem complicado de você chegar neles. Ele pode suportar até 14 casas decimais. Um integer suporta 5 dígitos no máximo. Num site comum dificilmente usa-se float. Você só vai usar isso em coisas mais elaboradas como sistemas de controle de estoque por exemplo.

Booleano

Um valor booleano é a síntese do sistema binário, onde baseamos a informática. O sistema binário consiste em representar tudo em apenas duas formas: 0=desligado e 1=ligado. O booleano é precisamente essa representação. Ele serve para determinar se algo é verdadeiro ou falso. 0 para falso e 1 para verdadeiro. Diversas funções do php retornam booleanos. Eles normalmente são usados nos verificadores de condição if/else.

Array

Array (vetor) é um tipo de variável largamente usado. Consiste basicamente em conjuntos de variáveis com um indexador e um valor. São pares, chaves ou indexadores e valor. Funciona como um índice de um livro: para cada página listada no índice temos um capítulo. Existem dois tipos de arrays: array unidimensional e array multidimensional.

Um array multidimensional é um array que contem outros arrays (matriz). Agora é como se você estivesse numa biblioteca. Imagine cada estante como sendo um array, daí cada prateleira como sendo um outro array dentro do array estante, e em cada prateleira os livros são arrays do array prateleira. Os capítulos do livro são os elementos do array livro. Para denominar as chaves ou indexadores de um array podemos usar basicamente *strings*. Números também são usados. Várias funções retornam arrays então é bom saber trabalhar com eles. Eles facilitam nossa vida consideravelmente em várias aplicações.

Objeto

Objeto é um tipo de variável exclusivo de programação orientada a objeto. É algo parecido com um array. Só que um objeto é composto de métodos e propriedades. Esses métodos e propriedades são determinados em classes. Um objeto é a instância de uma classe, ou seja, é a classe pronta para ser usada. Várias funções de php retornam objetos contendo somente propriedades.

Resource

Resource é um tipo especial de variável. Ele é gerado para controlarmos arquivos, conexões de internet e conexão com banco de dados. Quando fazemos uma consulta no banco de dados normalmente retorna-se um resource que deve ser trabalhado por outras funções.

3.4 - Segundo Exemplo em PHP

Vamos ver, agora, como realizar a declaração de variáveis no PHP.

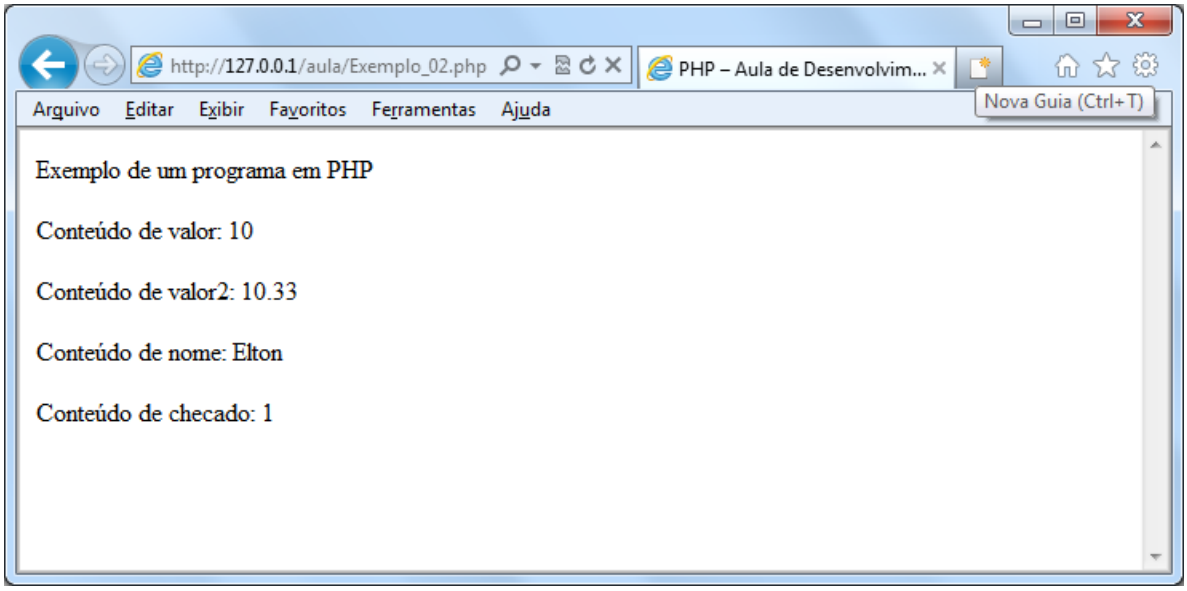
1) Abra o **Bloco de Notas** e digite o código abaixo:

```
<html>
<head>
      <title>PHP – Aula de Desenvolvimento de Software</title>
      <head>
      <body>
            <?php
               // Declarando variáveis
               $valor = 10;    // inteiro
               $valor2 = 10.33;    // ponto flutuante
               $nome = "Elton";  // variável texto – string
               $checado = true;  // variável booleana

               echo " Exemplo de um programa em PHP";
               echo "<P>";

               echo "Conteúdo de valor:  ";
               echo $valor;
               echo "<P>";

               echo "Conteúdo de valor2:  ";
               echo $valor2;
               echo "<P>";

               echo "Conteúdo de nome:  ";
               echo $nome;
               echo "<P>";

               echo "Conteúdo de checado:  ";
               echo $checado;
               echo "<P>";
            ?>
</body>
</html>
```

***Observação**:* você pode acrescentar comentários para as linhas de código de várias formas, umas delas é a utilização da string "//" antes do comentário. Quando o interpretador PHP encontra essa sequência ele ignora o restante da linha.

2) Salve com o nome Exemplo_02.php na pasta "C:\Arquivos de programa\EasyPHP-5.3.8.0\www\aula"

3) Para executar, abra o navegador, e digite o endereço: http://localhost/aula/Exemplo_02.php, você terá como resultado a seguinte tela:



3.4.2. Linha de Comentários no PHP

O PHP suporta comentários do '**C**', '**C++**' e **Unix shell**.

Exemplo:

```
<?php
    echo "Isto é um teste"; //Comentário de uma linha no C++
    /* Isto é um comentário de mais de uma linha
       e aqui temos outra linha */
    echo "Isto é um outro teste";
    echo "O último teste"; #Comentário no estilo Unix shell
?>
```

Os comentários de uma linha só terão efeito até o fim da linha, ou fim do bloco de código PHP atual, o que ocorrer primeiro.

```
<h1>Isto é um <?php # echo " simples";?> exemplo.</h1>
<p>No título acima você lerá 'Isto é um exemplo'.
```

No exemplo acima notamos que o comentário está dentro das **tags de abertura e fechamento do PHP**, logo tudo o que estiver fora destas tags serão visíveis no navegador.

3.5 - Terceiro Exemplo:

Utilização de duas variáveis que receberão, respectivamente, a data e hora atuais.

1) Abra o Bloco de Notas e digite o Código abaixo:

```
<HTML>
<BODY>
<?php

      $data = date("d/m/y",time());
      $hora = date("h:i", time());

?>

      Hoje é dia

      <?php
            echo $data;
      ?>

      e agora são

      <?php
            echo $hora;
      ?>

</BODY>
</HTML>
```

2) Salve com o nome Exemplo_03.php na pasta "C:\Arquivos de programa\EasyPHP-5.3.8.0\www\aula"

3) Para executar, abra o navegador e digite o endereço: http://localhost/aula/Exemplo_03.php.

Obs.: no exemplo acima, as variáveis `$data` e `$hora` receberam o valor da data e hora do sistema através da função `date( )`.

3.6 - Conhecendo melhor os tipos Básicos:

3.6.1 - Booleanos

Este é o tipo mais fácil. Um booleano expressa um valor de verdade. Ele pode ser `TRUE` ou `FALSE`. `True` é representado pelo número inteiro "1".

Sintaxe

Para especificar um literal booleano, use as palavras chave TRUE ou FALSE.

```
<?php
$valor = True; // assimila o valor TRUE para $valor
?>
```

3.6.2 - Inteiros

Um inteiro é um número do conjunto `Z = {..., -2, -1, 0, 1, 2, ...}`.

Sintaxe

O tamanho de um inteiro é dependente de plataforma, sendo um número aproximado a 2 bilhões o valor mais comum (número de 32 bits com sinal). O PHP não suporta inteiros sem sinal. Vide exemplos abaixo.

```
<?php
    $a = 1234; # número decimal
    $a = -123; # um número negativo
?>
```

3.6.3 - Overflow de inteiros

Se você especifica um número além dos limites do tipo inteiro, ele será interpretado como um ponto flutuante. Assim, se você realizar uma operação que resulte em um número além dos limites do tipo inteiro, um ponto flutuante será retornado também.

Nota: Não há operador de divisão inteira no PHP. `1/2` retorna o ponto flutuante `0.5`. Você pode moldar (casting) o valor para inteiro para sempre cortar o número, ou você pode usar a função round(). Veja o exemplo:

```
<?php
    var_dump(25/7);          // float(3.5714285714286)
    var_dump((int) (25/7)); // int(3)
    var_dump(round(25/7));  // float(4)
?>
```

3.6.4 - Números de ponto flutuante

Números de ponto flutuante (também conhecidos como "floats", "doubles" ou "números reais") podem ser especificados utilizando qualquer uma das seguintes sintaxes:

```
<?php
    $a = 1.234;
    $pi = 3.14159265;
?>
```

3.6.5 - Strings

Uma string é uma série de caracteres. No PHP, um caractere é o mesmo que um byte, ou seja, há exatamente 256 caracteres diferentes possíveis.

Nota: Não há nenhum problema nas strings se tornarem muito grande. Não há nenhum limite para o tamanho de *strings* imposta pelo PHP, então não há razão para se preocupar com *strings* longas.

Sintaxe

Uma *string* literal pode ser especificada de três formas diferentes.

- Apóstrofo.
- Aspas.

3.6.6 - Apóstrofos

A maneira mais simples para especificar uma *string* é delimitá-la entre apóstrofos o caracter ➔ (').

Para especificar um apóstrofo você precisará "escapá-la" com uma barra invertida (\), como em muitas outras linguagens. Se uma barra invertida precisa ocorrer antes de um apóstrofo ou no final da *string*, você precisa duplicá-la.

Note que se você tentar escapar qualquer outro caracter, a barra invertida também será impressa. Então geralmente não é necessário escapar a própria barra invertida.

Exemplo:

```
$texto = "Mamãe disse "oi papai" na hora."
```

Vai dar erro, pois existem **aspas duplas dentro de aspas duplas.**

Agora,

```
$texto= "Mamãe disse \"oi papai\" na hora" a barra invertida é o escape.
```

3.6.7 - Aspas

Se a *string* é delimitada entre aspas ("), o PHP entende mais sequências de escape para caracteres especiais:

Tabela - Sequências de escape

Sequência	Significado
\n	**fim de linha** (linefeed ou LF)
\r	**retorno de carro** (carriage return ou CR)
\t	**TAB horizontal**
\\	**contra barra ou barra invertida**
\$	**sinal de cifrão**
\"	**aspas**

Novamente, se você tentar escapar qualquer outro caractere, a barra invertida será impressa também!

3.7 - Interpretação de variáveis

Quando uma *string* é especificada dentro de aspas ou heredoc, variáveis são interpretadas dentro delas.

Há dois tipos de sintaxe: uma **simples** e outra **complexa**. A **sintaxe simples** é a **mais comum e conveniente**, provendo uma maneira de interpretar uma variável, um valor de array ou uma propriedade de object.

A **sintaxe complexa** pode ser reconhecida por chaves ({}) envolvendo a expressão.

Sintaxe simples

Se um sinal de cifrão ($) é encontrado, o interpretador tentará obter tantos identificadores quanto possíveis para formar um nome de variável válido. Envolva o nome da variável com chaves se você deseja explicitamente especificar o fim do nome.

```
<?php
$cerveja = 'Heineken';
    echo "O sabor das '$cerveja's é otimo"; // funciona, "'"
é um caracter inválido para nome de variáveis
    echo "Ele bebeu algumas $cervejas";     // não funciona,
's' é um caracter válido para nome de variáveis
    echo "Ele bebeu algumas ${cerveja}s";   // funciona
    echo "Ele bebeu algumas {$cerveja}s";   // funciona
?>
```

3.8 – Formato de números

Quando mostramos um número em uma página web podemos querer que tenha um formato específico. Por exemplo, que tenha só dois decimais, ou que utilize vírgulas -ou pontos- para separar decimais, assim como as unidades de milhar. Típicos formatos de número poderiam ser:

```
1.000.505,56
5003.60
5,000.00
...
```

Apresentar os números com um ou outro formato é simples, já que em PHP existe uma função específica para dar formato aos números, que dependendo dos parâmetros recebidos os formatará de uma maneira ou outra.

3.8.1 - Função number_format()

Realizaremos a formatação dos números com a função de PHP `number_format()`. Esta função recebe um, dois ou quatro parâmetros. Ou seja, temos estas restrições:

- Há um único parâmetro obrigado (o número que queremos formatar).
- O segundo parâmetro é opcional, tal como o terceiro e o quarto.
- Porém, se especificarmos o terceiro parâmetro, estamos obrigados a especificar também o quarto.

Vejamos com detalhe os parâmetros da função de formatação de números de PHP, com diversos exemplos.

Parâmetro 1, o número:

O primeiro parâmetro é o número a formatar. Tal como dissemos, embora seja óbvio, é um parâmetro sempre necessário.

```
$numero = 15200.67;
      echo number_format($numero);
//devolve 15,201
```

Neste caso, a formatação do número nos dará o número sem decimais e com uma vírgula como separador de milhares. Este formato é o que se utiliza em inglês (as pessoas de fala inglesa separam os milhares com vírgula ao escrever os números), que certamente não servirá aos desenvolvedores que trabalham em português.

Cabe observar que a função number_format() realizou também um arredondamento dos decimais que não está mostrando. Podemos ver bem este arredondamento no seguinte exemplo:

```
$numero = 999999999.99;
      echo number_format($numero);
//devolve 1,000,000,000
```

Parâmetro 2, os decimais:

Com o segundo parâmetro, que é opcional, indicamos o número de decimais que queremos que apareçam no número formatado.

```
$numero = 15200.67;
      echo number_format($numero,2);
//devolve 15,201.67
```

Como vemos, neste caso se incorporaram os decimais ao formato do número. Utiliza vírgulas para separar os milhares e um ponto para separar as unidades de milhar. Como vemos, continua utilizando a notação inglesa para formatar números.

Outro exemplo, no qual podemos apreciar que sempre se faz um arredondamento do número, se os decimais a mostrar forem menos que os que tiverem o número original.

```
$numero = 1885200.89;
      echo number_format($numero,1);
//devolve 1,885,200.9
```

Parâmetros 3 e 4, separadores de decimais e de unidades de milhar

Os últimos parâmetros, que devemos utilizar sempre juntos, servem para especificar os separadores que queremos utilizar para os decimais e as unidades de milhar. Se quisermos formatar os números com a notação portuguesa teríamos que utilizar obrigatoriamente estes parâmetros.

Por exemplo, assim faríamos para formatar os números em português:

- Separar os decimais com vírgula
- Separar as unidades de milhar com um ponto.

```
$numero = 1002002.365;
      echo number_format($numero, 2, ",", ".");
//devolve 1.002.002,37
```

Se, por exemplo, não separar os milhares, simplesmente passamos como separador de unidades de milhar (quarto parâmetro) a cadeia vazia:

```
$numero = 9540.2;
      echo number_format($numero, 2, ",", "");
//devolve 9540,20
```

4. Manipulando dados de formulários

Um formulário HTML é apenas um "rosto bonito" para onde os usuários poderão inserir informações que serão interpretados de alguma maneira por algum script do lado do servidor. E no nosso caso, esse script é um script PHP.

Primeiro: antes para poder enviar as informações, seu formulário deve conter um botão "submit", isso se consegue através do comando:

```
<input type=submit value="Texto do Botão">
```

Segundo: todos os campos que serão tratados no script PHP devem conter o parâmetro "NAME", caso contrário, os dados não serão passados para o script PHP. Ex:

```
<input type=text name=nome_do_campo>
```

Existem 2 métodos como as informações podem ser passadas: GET e POST. O recomendável sempre, para todos os formulários é usar o método POST, onde os dados enviados não são visíveis nas URLs, ocultando possíveis importantes informações e permitindo o envio de longas informações. O GET é totalmente o contrário disso.

4.1 – Passando informações para o script PHP

Listagem 1: Formulário HTML

```
<form action="script.php" method="post">
Campo 1: <input type=text name="campo1"><br>
Campo 2: <input type=text name="campo2"><br>
<input type=submit value="OK">
</form>
```

Esse formulário usa o método POST para envio das informações, então em "script.php":

Listagem 2: Uso de método POST para envio de formulário

```php
<?php
    echo "O valor de CAMPO 1 é: " . $_POST["campo1"];
    echo "<br>O valor de CAMPO 2 é: " . $_POST["campo2"];
?>
```

Se o formulário tivesse sido enviado usando o método GET, você simplesmente usaria $_GET no lugar de $_POST.

4.2 – Campos Hidden

Os campos hidden são usados para passar informações que não podem ser alteradas pelo usuário que estará inserindo informações no formulário. Por exemplo: você tem um site com sistema de login e o usuário quer alterar as informações de login dele. O script que irá manipular esse formulário, precisa saber o ID do usuário para poder alterar as informações no banco de dados, então esse ID é um campo hidden.

Exemplos:

Listagem 3: hidden.html

```html
<form action="hidden.php" method="post">
<input type=hidden name=escondido value="valor do
escondido">
<input type=hidden name=id value="111">
<input type=submit>
</form>
```

Listagem 4: hidden.php

```php
<?php
    echo "Campo Hidden: " . $_POST["escondido"];
    echo "<br>Oi, seu ID é: " . $_POST["id"];
?>
```

4.3 – Campos Text e Textarea

Os campos text e textarea são os tipos mais simples, onde há somente um possível valor por campo.

Exemplos:

Listagem 5: texts.html

```
<form action="texts.php" method="post">
Nome: <input type=text name=nome><br>
Email: <input type=text name=email><br><br>
Mensagem: <textarea name=mensagem cols=8 rows=3></textarea><br>
<input type=submit>
</form>
```

Listagem 6: texts.php

```
<?php
    echo "Olá " . $_POST["nome"] . " (email: " . $_POST["email"] .") <br><br>";
    echo "Sua mensagem: " . $_POST["mensagem"];
?>
```

4.4 – Campos Radio

Campos Radio permitem um relacionamento de um para muitos entre identificador e valor, ou seja, eles têm múltiplos possíveis valores, mas somente um pode ser pré-exibido ou selecionado. Por exemplo: você tem um sistema de "quiz".

Cada pergunta possui 5 possíveis respostas. Cada resposta é um *radio*, onde os 5 *radios* dessa pergunta possuem o mesmo identificador, mas cada com valores diferentes.

Exemplos:

Listagem 7: radio.html

```
<form action="radio.php" method="post">
<B>Qual seu sistema operacional?</B><br>
<input type=radio name=sistema value="Windows 8.1"> Win 8.1
<input type=radio name=sistema value="Windows 10"> Win 10
<input type=radio name=sistema value="Linux"> Linux
<input type=radio name=sistema value="Mac"> Mac
<br><br>

<B>Qual a marca de seu monitor?</B><br>
<input type=radio name=monitor value="Samsung"> Samsung
<input type=radio name=monitor value="LG"> LG
<input type=radio name=monitor value="Desconhecido"> Desconhecido
<br><br>
<input type=submit>
</form>
```

Listagem 8: radio.php

```
<?php
    echo "Seu sistema operacional é: " . $_POST["sistema"];
    echo "<br>Seu monitor é: " . $_POST["monitor"];
?>
```

4.5 – Campos Checkbox

O tipo Checkbox tem somente um possível valor por entrada: on value (marcado) ou no value (desmarcado). No script você deve fazer a verificação para saber se o campo foi marcado ou não.

Se é possível também utilizar grupos de checkbox com o mesmo nome. Para você deve adicionar "[]" no final do nome, para o PHP interpretar como array, veja o código exemplo.

Exemplos:

Listagem 9: checkbox.html

```
<form action="checkbox.php" method="post">
<B>Escolha os numeros de sua preferência:</B><br>
<input type=checkbox name="numeros[]" value=10> 10<br>
<input type=checkbox name="numeros[]" value=100> 100<br>
<input type=checkbox name="numeros[]" value=1000> 1000<br>
<input type=checkbox name="numeros[]" value=10000> 10000<br>
<input type=checkbox name="numeros[]" value=90> 90<br>
<input type=checkbox name="numeros[]" value=50> 50<br>
<input type=checkbox name="numeros[]" value=30> 30<br>
<input type=checkbox name="numeros[]" value=15> 15<br><br>
<input type=checkbox name="news" value=1>
<B>Receber Newsletter?</B><br><br>
<input type=submit>
</form>
```

Listagem 10: checkbox.php

```
<?php

if(isset($_POST["numeros"]))
{
    echo "Os números de sua preferência são:<BR>";
    foreach($_POST["numeros"] as $numero)
    {
        echo "- " . $numero . "<BR>";
    }
}
else
{
    echo "Você não escolheu número preferido!<br>";
}
if(isset($_POST["news"]))
{
    echo "Você deseja receber as novidades por email!";
}
else
{
    echo "Você não quer receber novidades por email...";
}
?>
```

Obs: A função `isset` serve para saber se uma variável existe... Ela retornará true (verdadeiro) quando uma variável existir e false (falso) quando uma variável não existir.

5. Estruturas de controle

As estruturas controle servem para definir o fluxo do programa, ou seja, quais instruções devem ser executadas, de acordo com os dados de entrada.

A estrutura de controle no PHP é dividida em duas partes: comandos de seleção e comandos de repetição, veja abaixo, com mais detalhe essa divisão e seus respectivos comandos:

- **Comandos de seleção:** são também conhecidos como comandos condicionais, neles é possível executar comandos ou bloco de comandos com base em testes feitos durante a execução, os comandos são: IF e SWITCH.
- **Comandos de Repetição:** São comandos utilizados para que um conjunto de instruções seja executado repetidamente por um determinado número de vezes ou até que uma condição seja atingida, os comandos são: WHILE, DO..WHILE, FOR e FOR EACH.

5.1 – If Else

A condição "if expressão instrução" serve para validar uma condição, e mediante o resultado, executar o código correspondente. Esta condição é utilizada nas mais diversas situações na programação, bem como no nosso dia a dia.

Exemplo:

```
if "tenho dinheiro" "Vou ao Cinema"
```

A instrução **Else** serve para executar um pedaço de código, caso a condição seja Falsa:

```
if "não está chovendo" "vou à praia" else "Fico em casa"
```

Para delimitar um bloco de instruções em PHP, utilizamos as chaves. "**{**" marca o início do bloco, e o "**}**" o fim do mesmo. Utilizamos blocos de instruções para indicar o código que queremos executar, num determinado momento.

Exemplo:

```
if "for ao supermercado"
   {
      "Comprar pão";
      "Comprar bebidas";
      "Comprar frutas";
   }
else
   {
```

```
        "Vou ao cinema";
        "Vou ao show de rock";
    }
```

Listagem 1: Exemplo de uma estrutura IF básica

```
<?php
    if ($AA == $BB){
        comando1;
        comando2;
    }
?>
```

Um dos comandos mais fáceis de entender entre os que serão apresentados, com certeza é o IF, com ou sem o ELSE, o que ele faz? Ele verifica a condição e executa o comando indicado se o valor for TRUE (valor diferente de zero). Veja abaixo o exemplo:

Listagem 2: Exemplo de estrutura utilizando IF-ELSE

```
<?php
    if (expressão) {
        comandos;
    } else {
        comandos;
    }
?>
```

Vejamos um exemplo prático em que podemos utilizar uma estrutura de controle de seleção, na listagem 3:

Listagem 3: Exemplo prático de estrutura utilizando IF-ELSE.

```
<?php
    $nota1 = 5;
    $nota2 = 5;

    $notaTotal = ($nota1 + $nota2) / 2;

    if ($notaTotal < 6){
        echo "Aluno Reprovado";
    } else {
        echo "Aluno Aprovado";
    }
?>
```

Múltiplos IFs podem ser encadeados. Exemplo:

```
if "Dia=Sábado" "Fico em casa";
```

```
else if "Dia=Domingo" "Vou passear";
else "É dia da semana, vai trabalhar!!!";
```

Exemplos de expressões para validar as condições:

```
$a == $b Verdadeiro se $a é igual a $b.
$a != $b Verdadeiro se $a diferente de $b.
$a < $b Verdadeiro se $a menor que $b.
$a > $b Verdadeiro se $a maior que $b.
$a <= $b Verdadeiro se $a menor ou igual a $b.
$a >= $b Verdadeiro se $a maior ou igual a $b.
```

Podemos ainda utilizar operadores lógicos para otimizar as condições, conforme as nossas necessidades:

```
$a and $b - And - Verdadeiro se ambos $a e $b forem verdadeiros.
$a or $b - Or - Verdadeiro se $a ou $b forem verdadeiros.
$a xor $b - Or - Verdadeiro se $a ou $b forem verdadeiros, mas não os dois.
! $a - Not - Verdadeiro se $a for falso.
$a && $b - And - Verdadeiro se $a e $b forem verdadeiros.
$a || $b - Or - Verdadeiro se $a ou $b forem verdadeiros.
```

Exemplo:

```
if (($dinheiro > 5000) and !($pais_em_casa)) {
    echo "Vou pra farra!!";
}else {
    echo "Tenho que ficar em casa.. <br>";
    echo "Mas vou pra net!!!";
}
```

Traduzindo: Se tivermos mais de $5000 e se os pais não estiverem em casa, podemos ir para a "farra". Senão, temos de ficar em casa, e claro, ir para net!

Podemos utilizar condições para decidir que blocos de código queremos executar. Podemos encadear várias condições para refinar diversas soluções.

5.2 – Switch/Case

A grosso modo, pode-se dizer que o SWITCH funciona como a união de vários IF, porém, de forma mais compacta e organizada. No entanto o switch trabalha basicamente com o operador de igualdade. Então em casos que devemos testar se nossa variável, ou expressão, é igual a uma série de valores, o switch é uma boa saída. Imagine, por exemplo, uma situação em que precise ser lido um valor inserido pelo usuário. Esse valor, por sua vez, pode assumir cinco valores. Seria necessário escrever cinco IFs seguidos, para avaliar todas as possibilidades. Vejamos como ficaria isso na prática.

Listagem 5: Verificação de várias possibilidades com IF

```
if($opcao == 1)
 //código 1
else if($opcao == 2)
 //código 2
else if($opcao == 3)
 //código 3
else if($opcao == 4)
 //código 4
else if($opcao == 5)
 //código 5
else
 //nenhuma das opções
```

Esse código poderia ser bastante reduzido utilizando o operador SWITCH, que avalia uma expressão com base em vários valores predeterminados. Se a expressão não possuir nenhum dos valores válidos, um valor padrão pode ser definido. Abaixo temos a sintaxe básica dessa estrutura.

Listagem 6: Sintaxe básica do SWITCH

```
switch(expressão)
{
case valor1:
//código 1
break;
case valor2:
//código 2
break;
default:
//código padrão
break;
}
```

Como se vê, em um único bloco são avaliadas várias possibilidades. O comando break é necessário ao final de cada bloco de código (que não precisa ser cercado por chaves) para que as demais opções sejam descartadas, caso uma seja atendida.

O exemplo acima poderia então ser reescrito da seguinte forma:

Listagem 7: Verificação de várias possibilidades com SWITCH

```
switch($opcao)
{
case 1:
//código 1
break;
case 2:
//código 2
break;
case 3:
//código 3
break;
case 4:
//código 4
break;
case 5:
//código 5
break;
default:
//nenhuma das opções
break;
}
```

Vamos agora ver um problema para entendermos melhor o switch.

Problema

Em um programa de televisão o expectador deve escolher números de 1 a 5 para um sorteio de prêmios, em que cada número representa um prêmio. Então faremos:

1. Ler a entrada, o número que o expectador escolheu
2. De acordo com o número que o expectador escolheu retornar um prêmio
3. Se escolher um numero que não seja de 1 a 5 retornar uma mensagem de erro.

O nosso problema poderia ser resolvido apenas utilizando as estruturas de controle if, else e elseif. Veja um exemplo de como poderia ser resolvido utilizando apenas if, else e elseif:

```php
$numero = 3;

if ($numero == 1){
  $mensagem = "uma bicicleta";
} elseif ($numero == 2) {
  $mensagem = "20 mil reais em barras de ouro";
} elseif ($numero == 3) {
  $mensagem = "uma casa";
} elseif ($numero == 4) {
  $mensagem = "um conjunto de panelas";
} elseif ($numero == 5 ) {
  $mensagem = "um carro zero";
} else {
  $mensagem = "nada, infelizmente";
}

echo "Parabéns o seu prêmio foi: $mensagem" ;
```

Observe o código que comparamos sempre a variável `$numero` a um valor de 1 a 5 e retornamos uma `$mensagem`, e se não fosse de 1 a 5 retornaríamos que ele não ganhou nada. Se estamos sempre comparando uma variável, ou expressão se são iguais a uma série de valores talvez seja à hora de trocarmos de estrutura de controle.

Ao utilizarmos o switch informamos a variável ou expressão que será testada em cada uma das cláusulas (case) até que seja encontrada uma cláusula em que seja verdadeira. Quando isto ocorre às instruções dentro do bloco de código da estrutura case é executado até encontrar a instrução *break*, se nenhuma cláusula for verdadeira a instrução *default* será executada.

```php
$numero = 3;

switch ( $numero ){
  case 1:
    $mensagem = "uma bicicleta";
    break;
  case 2:
    $mensagem = "20 mil reais em barras de ouro";
    break;
  case 3:
    $mensagem = "uma casa";
    break;
  case 4:
    $mensagem = "um conjunto de panelas";
    break;
  case 5:
    $mensagem = "um carro zero";
    break;
  default:
    $mensagem = "nada, infelizmente";
}

echo "Parabéns o seu prêmio foi: $mensagem";
```

6. Estruturas de repetição

Quando for necessário efetuar a repetição de um trecho de um programa um determinado número de vezes o que você faria? Escreveria de novo? Copiaria e colaria? Bem feio né? Afinal estamos utilizando de maquinas que são feitas para trabalhar para nós e não o contrário. Quando encontrarmos um problema como o apresentado anteriormente temos os laços de repetição, também conhecidos como loopings ou laços, malhas de repetição que poderão nos ajudar. Vamos a um problema para tentarmos resolver:

Problema

Bart Simpson ficou novamente na detenção no final da aula. E como punição terá que escrever: “Estou aprendendo loopings no PHP” 100 vezes no quadro, ou no nosso caso na tela.

Com o conhecimento que aprendemos até aqui faríamos algo como:

```
<?php

$mensagem = "Estou aprendendo loopings no PHP";

echo $mensagem . "<br>";
echo $mensagem . "<br>";
echo $mensagem . "<br>";

?>
```

Até repetirmos 100 vezes. O que não é muito inteligente né? Vamos conhecer agora a nossa primeira estrutura de repetição, o while.

6.1. While

O while executa um teste lógico, que retorne verdadeiro ou falso, no início do looping para verificar se é permitido ou não executar as instruções. Traduzindo while para português obtemos “enquanto” sendo assim as instruções serão executadas enquanto o teste do looping for considerado verdadeiro.

A estrutura while tem seu funcionamento controlado por decisões podendo executar um determinado conjunto de instruções enquanto a condição for verdadeira (`True`) e no momento em que a condição for avaliada como falsa (`False`) o processamento da rotina é desviado para fora do looping. Se desde o inicio a condição for tratada como falsa o looping não será executado.

O while é utilizado para executar um bloco de código várias vezes, enquanto uma determinada condição for atendida. Traduzindo, esta estrutura funciona da seguinte forma: **ENQUANTO** (condição for atendida) **FAÇA ALGO**. Onde **FAÇA ALGO** pode ser um ou vários comandos PHP.

A sintaxe básica de uso dessa estrutura é apresentada abaixo.

Listagem 8: Sintaxe básica do WHILE

```
while(condição)
{
 //comandos
}
```

Nesse caso, a condição avaliada também deve ser booleana, podendo assumir os valores true ou false.

A Listagem 9 mostra um exemplo prático de uso do operador WHILE.

Listagem 9: Exemplo de uso do WHILE

```
var $contador = 0;
while($contador < 10)
{
echo $contador;
$contador++;
}
```

Nesse exemplo, temos uma variável “contador” cujo valor é avaliado e, enquanto for menor que dez, um código é executado, imprimindo este valor e o incrementando em uma unidade.

Após a execução do bloco de comandos, o valor da expressão é novamente avaliado e, se for válido, os códigos são novamente executados.

Voltando ao nosso problema

Observe como ficaria a representação em PHP da nossa estrutura while para resolver o problema de Bart Simpson. E como Bart demorou muito ele ainda teve que numerar para termos certeza que ele escreveu às 100 vezes. Observe o código:

```
<?php

$mensagem = "Estou aprendendo loopings no PHP";

$i = 1;
while( $i <= 100 ){
  echo $i . ' - ' .$mensagem . '<br>';
  $i++;
}

?>
```

Iniciamos atribuindo um valor a variável `$mensagem` em seguida criamos uma variável `$i` que é conhecida como contador ou sentinela. Logo abaixo de nosso contador iniciamos o while que avalia se `$i` é menor que 100, se for verdadeiro ele executa a instrução se não ele sai do looping. A primeira volta de nosso looping a instrução é verdadeira então dentro do looping escrevemos a mensagem e somamos mais 1 a variável `$i` com o operador de pós incremento. Após terminar isto o looping volta e avalia `$i` aqui com o valor de 2 e se for verdadeira a expressão do while inicia tudo novamente.

Observação: Você sabe o porquê do `$i` no nosso looping? A utilidade dele ali é fazer em um momento a expressão se tornar falsa senão teremos um looping infinito. Como o nosso while avalia se `$i` é menor que 100 se sempre ele for menor que 100 o looping nunca parará, por isso que temos que **incrementá-lo** para poder uma hora ele chegar ao valor em que a instrução se tornará falsa. Observe o código em que não incrementamos o `$i` antes de sairmos do looping. Execute e rapidamente pause a execução do browser, pois como isto nunca terá fim seu browser vai ir ficando bem lento e podendo até travar.

```php
<?php

$mensagem = "Estou aprendendo loopings no PHP";

$i = 1;
while( $i <= 100 ){
  echo $i . ' - ' .$mensagem . '<br>';
}

?>
```

Observe agora o mesmo exemplo de Bart Simpson escrevendo de 1 a 100, no entanto agora ele escrevendo os números em ordem decrescente.

```php
<?php

$mensagem = "Estou aprendendo loopings no PHP";

$i = 100;
while( $i >= 1 ){
  echo $i . ' - ' .$mensagem . '<br>';
  $i--;
}

?>
```

Veja que alteramos apenas a lógica agora iniciamos o `$i` com 100 e executamos o looping enquanto `$i` for maior ou igual a 1 e a cada volta do looping diminuímos um para que em um momento termos `$i` com o valor de 1 e encerrarmos o looping.

6.2. For

A estrutura de repetição FOR é utilizada para se executar um conjunto de comandos por um número definido de vezes. Para esse operador, são passados uma situação inicial, uma condição e uma ação a ser executada a cada repetição.

Em geral informamos uma variável que serve como contador de repetições, com seu valor inicial, uma condição a ser atendida para que cada repetição seja executada e um incremento ao contador.

Observando a sintaxe a seguir fica mais fácil compreender o funcionamento.

Listagem 10: Sintaxe do operador FOR

```
for(valor inicial; condição; incremento)
{
 //comandos
}
```

Na Listagem 11 é mostrado um exemplo prático para facilitar a compreensão dessa estrutura.

Listagem 11: Exemplo de uso do FOR

```
for($contador = 0; $contador < 10; $contador++)
{
 //comandos
}
```

O código acima pode ser entendido como “com o contador partindo do zero e enquanto este for menor que 10, execute os comandos e incremente uma unidade em seu valor”.

Problema:

Bart Simpson ficou novamente na detenção no final da aula. E como punição terá que escrever: “Ainda estou aprendendo loopings no PHP” 100 vezes no quadro, ou no nosso caso na tela.

Sabemos que com o `while` podemos resolver este problema facilmente, no entanto agora vamos ver uma abordagem diferente com o uso do looping `for`.

Assim como o `while` o `for` é uma estrutura de controle responsável pela realização de loopings no PHP.

O comando `for` aceita 3 expressões na sua declaração sendo elas:

1. Expressão que é avaliada apenas uma vez na primeira iteração, volta, do looping. Geralmente a utilizamos para iniciar a nossa variável de controle, contador, como por exemplo `$i = 1`.

2. Expressão que é avaliada no início de cada iteração do looping, e caso retorne falso, o looping `for` será encerrado. A segunda expressão como você pode ver pode ser considerada a condicional do nosso looping, assim como tínhamos no `while`. Um exemplo seria `$i <= 100`.
3. Expressão que é avaliada no final de cada iteração do looping, normalmente utilizada para alterar o valor da variável de controle. Como por exemplo utilizar o operador de incremento `$i++`.

Agora vejamos como seria a representação em PHP da nossa estrutura de controle `for` para resolvermos o problema de Bart Simpson.

```
<?php
$mensagem = "Ainda estou aprendendo loopings no PHP";

for( $i = 1; $i <= 100; $i++ ){

  echo $i . ' - ' .$mensagem . '<br>';

}

?>
```

Como você pode observar diferentemente do `while` não necessitamos incrementar o contador dentro do nosso looping pois isto já fica implícito na própria declaração do for. E detalhe que o separador de instruções, expressões, no `for` é o ; (ponto e vírgula).

6.2.1. Estruturas de controle aninhadas

Também conhecido como desvio condicional encadeado, as estruturas de controle aninhadas são utilizadas em casos em que é necessário estabelecer verificações de condições sucessivas, ou seja, condições dentro de condições. Este tipo de estrutura pode possuir diversos níveis de condições aninhadas. Vamos a um exemplo.

Problema

Como acabamos de aprender o looping for utilizaremos ele neste exemplo. Nosso problema é o seguinte:

- criar um programa que escreve de 1 a 100
- imprimir todos os números
- ao lado de todos os número mostrar se o número é par ou impar

Vamos ver no PHP como ficaria no PHP:

```php
<?php

for( $i = 1; $i <= 100; $i++ ){

  if ( $i % 2 == 0 ){
    echo $i . " é par " . "<br>";
  } else {
    echo $i . " é impar " . "<br>";
  }

}

?>
```

Inicialmente o mesmo looping `for` que aprendemos, no entanto observe que dentro do looping surgiu uma condicional `if` que é a nossa estrutura de controle aninhada, pois se encontra dentro do looping `for`.

A nossa lógica do `if` é: se `$i` módulo de 2 for igual a zero retorna verdadeiro caso contrário falso. Ou seja, se dividir `$i` por 2 e não houver resto significa que `$i` é par se não é ímpar.

6.3. Foreach

A estrutura foreach nos permite realizar uma varredura completa em um array, retornando seus valores e / ou chaves ou executando operações sobre esses valores. Funciona somente com arrays, e lançará um erro se tentar utilizá-lo em uma variável de qualquer tipo diferente ou em variáveis não inicializadas.

6.3.1. Array

Suponha que seja necessário armazenar e manipular dezenas de nomes de pessoas num programa de computador. De acordo com o que estudamos até aqui seriam necessárias dezenas de variáveis, cada uma armazenando um nome diferente, como por exemplo, `$nome1="Claudinelson"`, `$nome2="Creosméria"` e assim por diante.

Imagine também uma lista de compras que você está levando ao mercado:

1. Pão
2. Cerveja
3. Carne
4. Refrigerante

A lista de compras, que contém quatro elementos, vai ser chamada de **lista**. Perceba que se "procurarmos" o terceiro elemento da lista, veremos que é Carne.

Agora, imagine que você precise passar isso para programação, só que a sua lista de compras pode ter N elementos, e seria maluquice definir uma variável diferente para cada um desses elementos.

Nunca se esqueça: que cada variável ocupa espaço na memória do computador e faz o computador "perder" algum tempo procurando seu valor na memória.

Por isso existem os arrays: armazenar valores e/ou variáveis referentes a um mesmo grupo, a uma mesma origem.

Criar arrays no PHP é extremamente simples, veja dois exemplos onde criamos a nossa lista de compras:

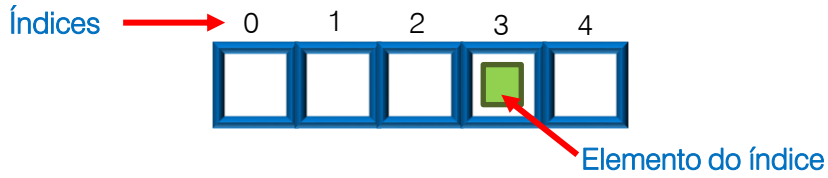
```php
<?php
    // Definição simples e rápida
    $lista = array('Pão', 'Ovos', 'Carne', 'Macarrão');

    // Definição mais longa, porém mais fácil de entender
    $lista = array();
    $lista[0] = 'Pão';
    $lista[1] = 'Ovos';
    $lista[2] = 'Carne';
    $lista[3] = 'Macarrão';

    // Outro exemplo
    $lista = array();
    $lista[] = 'Pão';
    $lista[] = 'Ovos';
    $lista[] = 'Carne';
    $lista[] = 'Macarrão';

?>
```

Em todos os três exemplos o resultado (`$lista`) será o mesmo… Vamos falar de cada um:

- No primeiro exemplo, definimos todos os quatro elementos na forma mais simples possível, separados por vírgula.
- Já no segundo exemplo definimos (antes) que `$lista` será um array, e logo após, definimos seus quatro elementos, perceba que temos um número entre colchetes agora: `$lista[1]` esse número significa o índice do array, a posição do elemento. O índice do elemento pode ser definido como numérico ou textual (string).
- E por fim, no terceiro exemplo, fazemos a mesma coisa que fizemos no segundo, só que omitimos os índices dos elementos, sendo assim, o PHP irá colocar cada elemento no fim do array, começando com a posição 0 (primeiro elemento) e crescendo a cada elemento.

Esse tipo de array é do tipo unidimensional, pois possui uma linha de elementos.

A figura abaixo ilustra exemplos de um array unidimensional:



É possível também definir índices (também chamados de keys ou chaves) como strings, veja um bom exemplo de uso:

```php
<?php
    // Definição longa
    $carro = array();
    $carro['cor'] = 'Vermelho';
    $carro['modelo'] = 'CrossFox';
    $carro['fabricante'] = 'Volkswagen';

    // Definição simples
    $carro = array('cor' => 'Vermelho', 'modelo' => 'CrossFox',
'fabricante' => 'Volkswagen');
?>
```

Quando você for exibir um elemento, é só seguir a mesma sintaxe da declaração longa:

```php
<?php
    // Carro
    $carro = array();
    $carro['cor'] = 'Vermelho';
    $carro['modelo'] = 'CrossFox';
    $carro['fabricante'] = 'Volkswagen';
    echo "A cor do carro é: " . $carro['cor'];
    // Resultado: A cor do meu carro é: Vermelho

    // Lista
    $lista = array('Pão', 'Ovos', 'Carne', 'Macarrão');
    echo "O segundo item da lista é: " . $lista[1];
    // Resultado: O segundo item da lista é: Ovos
?>
```

Vale lembrar, que quando estamos trabalhando com índices numéricos (ordenados), a posição inicial é a zero, então a quarta posição será o índice `[3]`.

6.3.2. Correndo um ARRAY de única linha com FOR

Como foi mencionado sobre arrays, pode-se percorrer um array com o `for`, confira no exemplo:

```php
<?php

$nomes[0] = 'Claudinelson';
$nomes[1] = 'Creosméria';
$nomes[2] = 'Ethelvina';
$nomes[3] = 'Wandercleisson';
$nomes[4] = 'Claraíde';

for ($i=0;$i<5;$i++){
      echo $nomes[$i].'<br>';
 }
?>
```

No exemplo acima, temos um array de única linha chamado `$nomes`, onde foram adicionados alguns nomes dentro dele. Depois no comando `for`, iniciou-se a variável `$i = 0` e enquanto a posição fosse menor que a quantidade de elementos, que no caso são 5, incrementou-se a `$i`.

Veja abaixo mais um exemplo:

```php
<?php

$frutas=array("maçã","banana","melancia","melão","abacaxi","laranja");

   for ($i=0; $i < count($frutas); $i++){
      echo "A fruta na posição: " . $i . " é " . $frutas[$i] . "<br>";
   }
?>
```

Nesse outro exemplo, temos um array de única linha, chamado de `$frutas` onde foram adicionados alguns valores dentro dele com o nome de algumas frutas. Depois no comando `for`, iniciou-se a variável `$i = 0` e enquanto a posição fosse menor que `count($frutas);` incrementou-se a variável `$i`.

Obs: a função COUNT conta a quantidade de itens dentro de um array. Ele vai retornar a quantidade de posições que tem dentro do array. Veja que a quantidade de posições é 6, pois temos 6 "frutas" dentro do array. Portanto, dentro da expressão do laço `for` não foi necessário especificar a quantidade de elementos.

6.3.3. Array bidimensional

Também conhecido como matrizes, em geral são caracterizadas por se tratarem de uma única variável de um determinado tamanho que guarda várias informações do mesmo tipo. Essas informações são gravadas na memória sequencialmente e são referenciadas através de índices.

6.3.3. Usando o foreach

Para cada elemento de um array, você executa um grupo de comandos utilizando o **foreach**. Veja o exemplo:

```php
<?php

$vetor = array("Claudinelson", "Creosméria", "Valdimila");

foreach ($vetor as $indice) {
    echo $indice . "<br>";

}
?>
```

A sintaxe do FOREACH é assim, inicia-se com `foreach`, abre parênteses, diz qual é o array a ser percorrido, define a palavra `as` e depois define o nome da variável que vai estar na posição atual. O próprio FOREACH vai inserir o valor na variável escolhida, e esta poderá ser usada dentro do laço, conforme o exemplo acima.

7. Gerenciamento de Banco de Dados com MYSQL

Neste capítulo faremos uma breve introdução ao Sistema Gerenciador de Banco de Dados **MySQL**.

Um Sistema Gerenciador de Banco de Dados (SGBD) é um conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de dados. O principal objetivo é "retirar" da aplicação/cliente a responsabilidade de gerenciar o acesso, manipulação e organização dos dados.

O SGBD disponibiliza uma interface para que as aplicações/clientes possam incluir, alterar ou consultar dados. Nos Bancos de Dados Relacionais a interface é constituída pelasAPIs ou drivers do SGBD, que executam comandos na linguagem SQL.

7.1 - Linguagem SQL

É uma linguagem de pesquisa declarativa para banco de dados relacional (bases de dados relacionais). Muitas das características originais do SQL foram inspiradas na álgebra relacional.

SQL é normalmente pronunciado em português como "**esse-quê-ele**", porém sua pronuncia correta deveria se "**síquel**", do inglês "**sequel**", ou "alguma coisa que segue outra coisa". **SQL** é uma brincadeira com o nome da primeira linguagem de consulta **QUEL**.

Embora, o SQL tenha sido originalmente criado pela IBM, rapidamente, surgiram vários "**dialectos**" desenvolvidos por outros produtores. Esta expansão levou à necessidade de ser criado e adaptado um padrão para a linguagem. Esta tarefa foi realizada pela American National Standards Institute (ANSI) em 1986 e ISO em 1987.

O SQL foi revisto em 1992 e a esta versão foi dado o nome de **SQL-92**. Foi revisto novamente em 1999 e 2003 para se tornar **SQL:1999 (SQL3)** e **SQL:2003**, respectivamente. O SQL:1999 usa expressões regulares de emparelhamento, **queries** recursivas e **gatilhos** (triggers). Também foi feita uma adição controversa de tipos não-escalados e algumas características de orientação a objeto. O SQL:2003 introduz características relacionadas ao XML, sequências padronizadas e colunas com valores de auto generalização (inclusive colunas-identidade).

7.2 - MySQL

O **MySQL** é um sistema de gerenciamento de banco de dados (SGBD), **gratuito**, que utiliza a linguagem SQL (Structured Query Language - Linguagem de Consulta Estruturada) como interface.

É atualmente um dos bancos de dados mais populares, com mais de 4 milhões de instalações pelo mundo.

7.2.1 - História

O **MySQL** foi criado na Suécia por dois Suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que trabalham juntos desde a década de 1980. Hoje seu desenvolvimento e manutenção empregam aproximadamente 70 profissionais no mundo inteiro, e mais de mil contribuem testando o software, integrando-o a outros produtos, e escrevendo a respeito do mesmo.

Nota: *O sucesso do MySQL deve-se em grande medida à fácil integração com o* ***PHP*** *incluído, quase que obrigatoriamente, nos pacotes de hospedagem de sites da Internetoferecidos atualmente. Empresas como Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics e Texas Instruments usam o MySQL em aplicações de missão crítica.*

O MySQL hoje suporta Unicode, Full Text Indexes, replicação, Hot Backup, GIS, OLAP e muitos outros recursos.

7.3 - Características

7.3.1 - Principais características:

- **Portabilidade** (suporta praticamente qualquer plataforma atual);
- **Compatibilidade** (existem drivers ODBC, JDBC e .NET e módulos de interface para diversas linguagens de programação, como Java, C/C++, Python, Perl, PHP e Ruby);
- **Excelente desempenho e estabilidade;**
- **Pouco exigente quanto a recursos de hardware;**
- **Facilidade de uso;**
- **Suporte a vários tipos de tabelas** (como MyISAM e InnoDB), **cada um específico para um fim**.

7.4 - Vantagens

- Outra grande vantagem é a de ter código aberto e funcionar em, quase, qualquer plataforma e sistema operacional : Windows, Linux, FreeBSD, BSDI, Solaris, Mac OS X,SunOS, SGI, etc.
- É reconhecido pelo seu desempenho e robustez e também por ser multitarefa e multi-usuário. A Wikipédia, usando o programa MediaWiki, utiliza o MySQL para gerenciar seu banco de

dados, demonstrando que é possível utilizá-lo em sistemas de produção de alta exigência e em aplicações sofisticadas.

- No passado, devido a não possuir (até a versão 3.x) funcionalidades consideradas essenciais em muitas áreas, como stored procedures, two-fase commit, subselects, foreign keys ou integridade referencial, é freqüentemente considerado um sistema mais "leve" e para aplicações menos exigentes, sendo preterido por outros sistemas como o PostgreSQL.

7.5 - Notas

- O MySQL a partir da versão 4.1 adicionou suporte a Transações, SubSelects, Foreign Keys e Integridade Referencial. Esse suporte foi graças ao database engine InnoDB.
- Com a versão 5.0, o MySQL incorporou mais recursos avançados ao sistema, incluindo views, triggers, storage procedures e transações XA.

7.6 - A Linguagem SQL e seus componentes:

7.6.1 - DML - Linguagem de Manipulação de Dados

Primeiro, há os elementos da DML (Data Manipulation Language - Linguagem de Manipulação de Dados). A DML é um subconjunto da linguagem usada para selecionar, inserir, atualizar e apagar dados.

SELECT é o comumente mais usado do DML, comanda e permite ao usuário especificar uma query como uma descrição do resultado desejado. A questão não especifica como os resultados deveriam ser localizados.

INSERT é usada para somar uma fila (formalmente uma tupla) a uma tabela existente.

UPDATE para mudar os valores de dados em uma fila de tabela existente.

DELETE permite remover filas existentes de uma tabela.

BEGIN WORK (ou START TRANSACTION, dependendo do dialeto SQL) pode ser usado para marcar o começo de uma transação de banco de dados que pode ser completada ou não.

COMMIT envia todos os dados das mudanças permanentemente.

ROLLBACK faz com que as mudanças nos dados existentes desde que o último COMMIT ou ROLLBACK sejam descartados.

COMMIT e **ROLLBACK** interagem com áreas de controle como transação e locação. Ambos terminam qualquer transação aberta e liberam qualquer cadeado ligado a dados. Na ausência de um BEGIN WORK ou uma declaração semelhante, a semântica de SQL é dependente da implementação.

7.6.2 - DDL-Linguagem de Definição de Dados

O segundo grupo é a **DDL** (Data Definition Language - Linguagem de Definição de Dados). Uma DDL permite ao usuário definir tabelas novas e elementos associados. A maioria dos bancos de dados de SQL comerciais tem extensões proprietárias no DDL.

Os comandos básicos da DDL são:

CREATE cria um objeto (uma Tabela, por exemplo) dentro da base de dados.

DROP apaga um objeto do banco de dados.

Alguns sistemas de banco de dados usam o comando **ALTER**, que permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente.

outros comandos DDL:

ALTER TABLE

CREATE INDEX

ALTER INDEX

DROP INDEX

CREATE VIEW

DROP VIEW

7.6.3 - DCL-Linguagem de Controle de Dados

O terceiro grupo é o **DCL** (Data Control Language - Linguagem de Controle de Dados). DCL controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

Duas palavras-chaves da DCL:

GRANT - autoriza ao usuário executar, ou setar operações.

REVOKE - remove ou restringe a capacidade de um usuário de executar operações.

Outros comandos DCL:

ALTER PASSWORD

CREATE SYNONYM

7.6.4 - DQL-Linguagem de Consulta de Dados

Embora tenha apenas um comando a **DQL** é a parte da SQL mais utilizada. O comando **SELECT** é composto de várias cláusulas e opções, possibilitando elaborar consultas das mais simples às mais elaboradas. Exemplos:

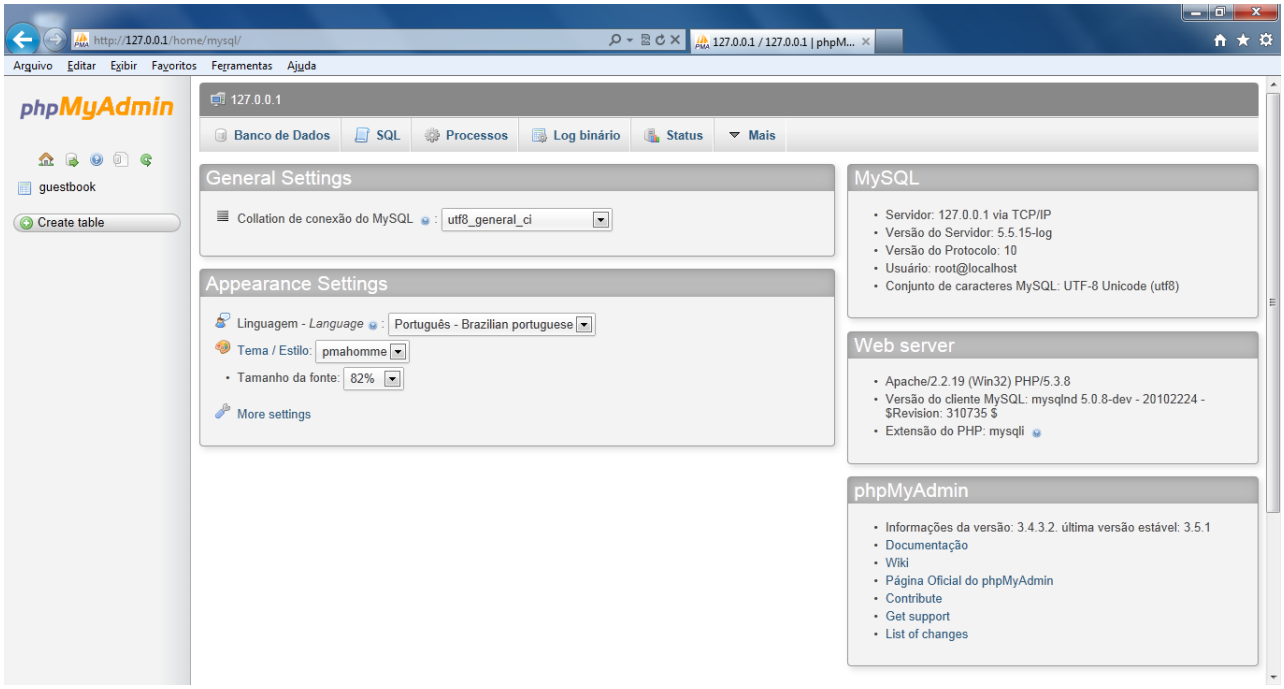
```
SELECT
  nome
FROM
  pessoas;

SELECT
  aP.codigo,
  aP.nome,
  aP.data_nascimento,
  aO.nome,
  aO.local
FROM
 pessoas aP,
 objetos aO,
WHERE
 aP.codigo = aO.codigo_pessoa and
 aP.codigo = (
   SELECT
     codigo_pessoa
   FROM
     catalogo
   WHERE
    cod_catalogo = 5
 );
```

8. Utilizando o PHPmyAdmin

8.1 - phpMyAdmin

É um programa desenvolvido em php para administração do MySQL que roda na plataforma WEB. A partir deste sistema é possível criar e remover bases de dados, criar/remover/alterar tabelas, inserir/deletar/editar campos, executar códigos SQL e manipular campos chaves.

Como já visto, quando instalamos o EasyPHP ele instala automaticamente o phpMyAdmin, para acessá-lo digite o seguinte endereço no seu web browser: http://localhost/home/mysql/. Se tudo estiver correto deverá surgir no browser a página parecida como a próxima figura.



Agora que o phpMyAdmin está carregado, poderemos começar a criar nossa base de dados, mas antes, vamos conhecer os tipos de dados suportados pelo MySQL para elaborar nossas tabelas.

8.2 - Tipos de dados mais comuns do MySQL

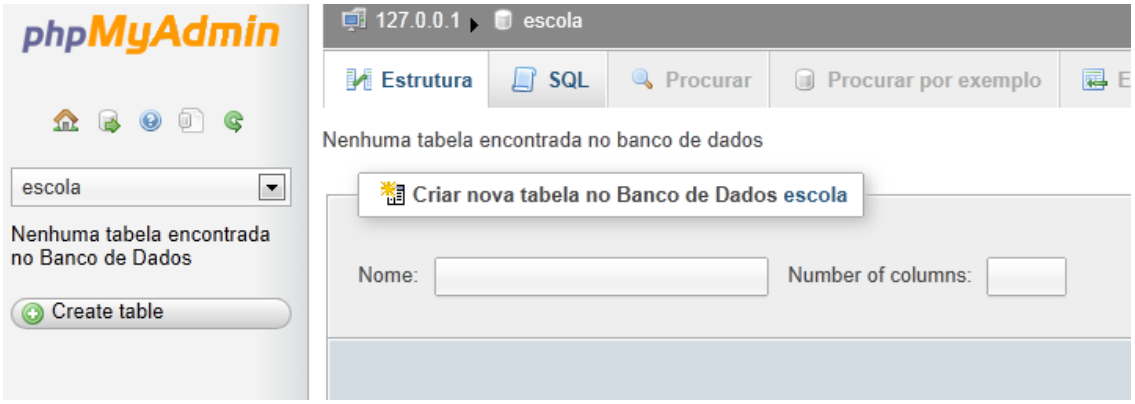
Tipo	Descrição
INT	Valor inteiro
REAL	Valor de ponto flutuante
CHAR (tamanho)	Valor de caractere de tamanho fixo. Valor inferior ao definido será deixado em branco.
TEXT (tamanho)	Valor de caractere de tamanho variável.
VARCHAR (tamanho)	Valor de caractere de tamanho variável. Valores inferiores ao definido serão suprimidos.
DATE	Valor para datas do tipo (AAAA-MM-DD)
TIME	Valor de tempo padrão
DATETIME	Valor para data e hora agregados

Agora que você sabe os tipos de dados suportados e a sintaxe de como criar uma tabela, é fundamental que você saiba o que deseja criar, isto é, como será a sua base de dados.

Iremos desenhar o modelo para a criação de uma base de dados simples, que seria o início do controle das informações de uma escola. Nossa base de dados será chamada de **ESCOLA** e nela teremos as tabelas **ALUNOS, CURSOS e DISCIPLINAS**. Vamos então ao nosso modelo.

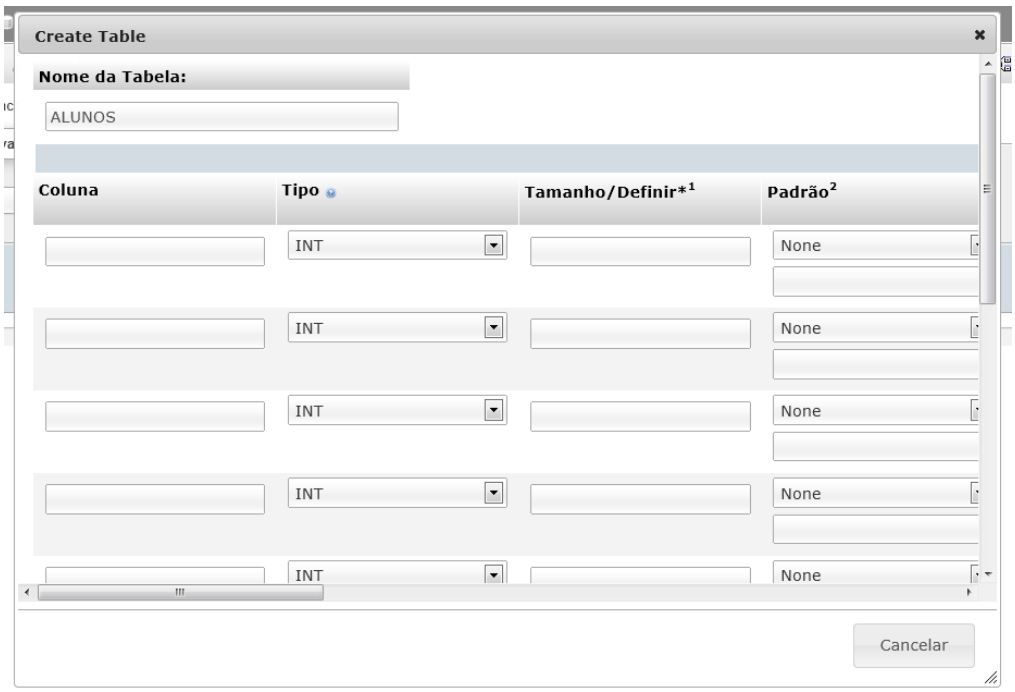
1) Primeiro passo, criar o Banco de Dados como nome: **ESCOLA**, para tanto no phpMyAdmin clique em **Banco de dados** e no campo **Criar novo Banco de Dados** digite "ESCOLA" no campo apropriado, veja a próxima figura:

2) Depois que você executar [depois de criada] clique no botão atualizar do seu browser, e do lado esquerdo na seção de Banco de Dados do MySQL escolha o banco criado.

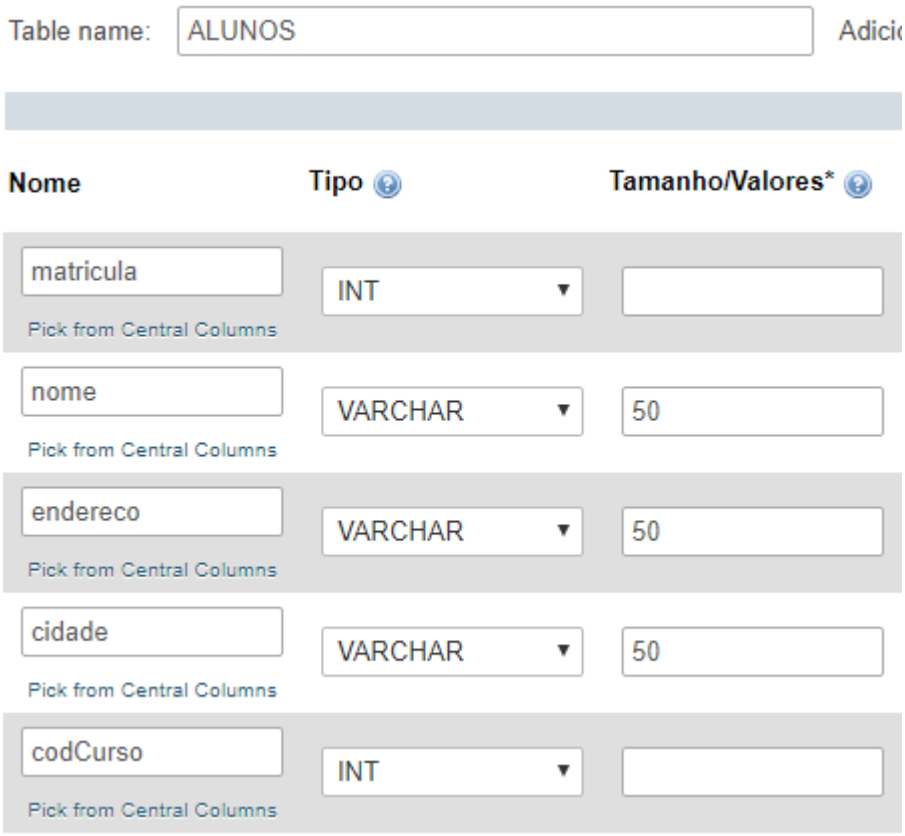


A figura acima mostra que o banco escola, ainda não possui nenhuma tabela, nesse momento iremos criar nossa primeira tabela chamada **ALUNOS**.

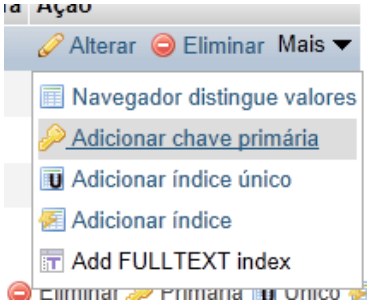
3) No campo "Nome" digite: **ALUNOS** e no campo "Número de colunas" digite: **5**, e em seguida clique no botão executar, veja figura:



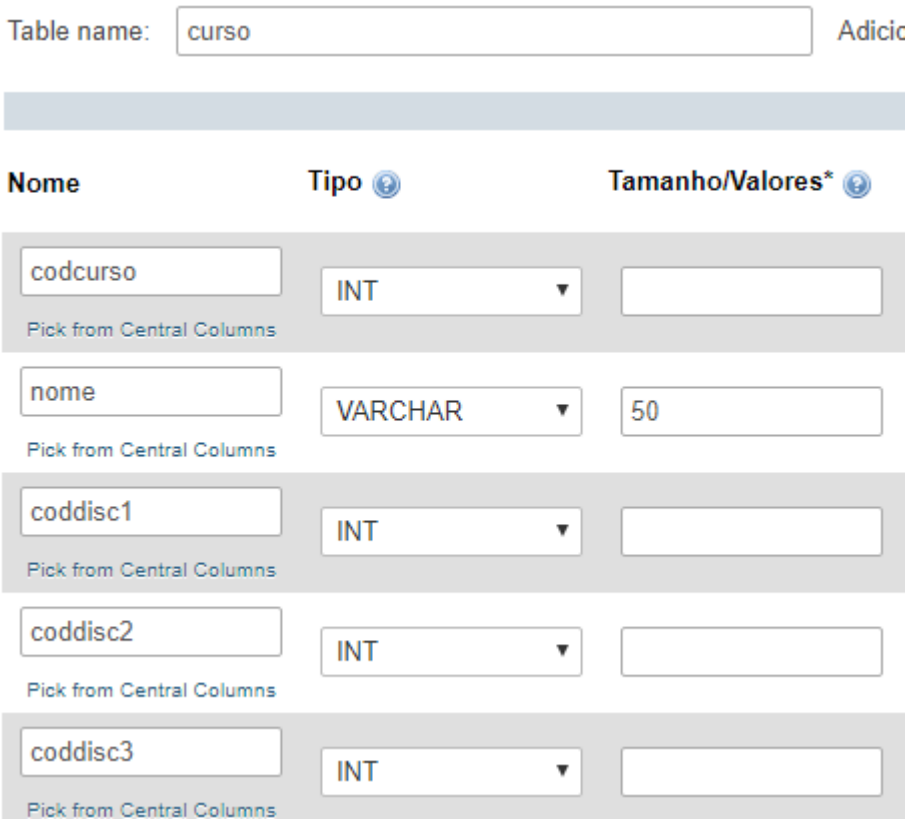
4) Nessa janela preencheremos a definição de cada campo, preencha para que fique exatamente igual à imagem abaixo e depois clique no botão [SALVAR]. Campos: matricula, nome, endereco, cidade, codcurso.



Após ter salvo abra a tabela e na opção "Mais" do campo **matricula** escolha **Adicionar chave primária**.



5) Vamos repetir o procedimento para criar a tabela **CURSOS**. Campos: codcurso, nome, coddisc1, coddisc2 e coddsic3.



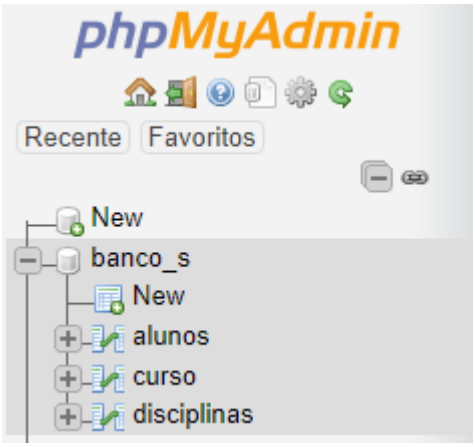
Obs: o campo **codcurso** deve ser chave primária da tabela.

6) Repetindo os procedimentos para criar a última tabela do nosso exemplo, ou seja, **DISCIPLINAS**, com os campos: coddisciplina, nomedisciplina.

	#	Coluna	Tipo	Collation	A
☐	1	coddisciplina	char(2)	latin1_swedish_ci	
☐	2	nomedisciplina	char(30)	latin1_swedish_ci	

Obs: o campo **coddisciplina** deverá ser chave primária da tabela.

7) Se tudo ocorrer de forma correta, selecionando o seu banco a esquerda, aparecerá o nome das três tabelas criadas.



8.3 - Explicando chave primária

Campo chave é o campo mais importante de nossa tabela, pois este é o campo que irá identificar a posição de todos os outros dados de um registro. Os dados deste campo são exclusivos, isto é, não poderão existir dois registros deste campo em sua tabela com o mesmo valor. Por isso, toda tabela deve ter um campo designado como chave primária para o controle dos registros.

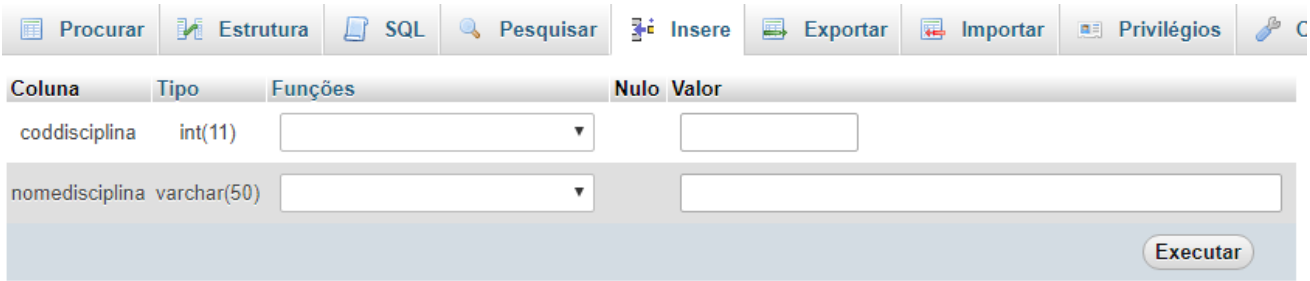
8.3.1 - Manipulando dados nas tabelas

Agora que já temos a nossa estrutura de tabelas prontas, é hora de começar a inserir os dados, ou seja, popular as tabelas, pois é para isso que elas servem, armazenam dados para que estes sejam pesquisados mais tarde.

Utilizando o **phpMyAdmin** nesse momento para popular as tabelas, poderíamos utilizar a instrução SQL chamada **INSERT INTO**, no entanto para simplificar o uso, vamos digitar diretamente na tabela. É simples basta você selecionar a tabela clicando sobre o nome dela no lado esquerdo da tela, e em seguida clicar no link insere.

8.3.2 - Digitando as Disciplinas

Clique no nome da tabela disciplinas e em seguida no link **Insere**, conforme figura:



No campo Valor na linha do **coddisciplina** digite "**11**", em nome da disciplina digite "**Banco de Dados**", escolha a opção inserir um novo registro, e repita o procedimento de inserção até que sua tabela tenha todos os registros abaixo:

coddisciplina	nomedisciplina
11	Banco de Dados
12	Lógica de Programação
13	Desenvolvimento de Software
14	Banco de Dados II
15	Projeto de Aplicações Web
16	Programação de Computadores I
17	Desenvolvimento e Design de Websites I
18	Gestão de SO I
19	Composição e Projeto

Para visualizar os dados na tabela, clique no menu **Procurar**.

8.3.3 - Digitando os Cursos

Agora que você já sabe como inserir dados nas tabelas digite os dados para a tabela **CURSOS** conforme próxima figura:

codCurso	nome	coddisc1	coddisc2	coddisc3
1	Técnico em Informática	11	12	13
2	Técnico em Manutenção de Computadores	12	14	18
3	Técnico em Informática para Internet	15	17	19

8.3.4 - Digitando os Alunos

Repita o procedimento para a tabela **ALUNOS** conforme próxima figura:

matricula	nome	endereco	cidade	codCurso
1	Marcos Moraes	Rua Pe Roque, 2057	Valentim Gentil	1
2	Richardson Lopes Souza	Rua Asfaltada, 9321	Parisi	3
3	Guilhermina Conceição Lopes	Rua Pará, 23	Jales	1
4	Ezelina Cristina Pimenta	Rua Altair Lopes, 33	Parisi	2
5	Claudinelson Tavares	Rua Novo Horizonte	Votuporanga	2
6	Pietra Luiza dos Santos	Rua Lopes Conte, 3343	Cosmorama	1
7	Elvira Savana Alencar	Rua Inglaterra, 496	Votuporanga	3
8	Alan Cripta Mausoléu	Av. da Saudade, 666	Jales	3
9	Milena Fatality	Rua Outworld, 43	Votuporanga	2

8.3.5 - Visualizando dados – Instrução SELECT

Não iria fazer sentido você armazenar dados se não pudesse visualizá-los quando fosse preciso, correto? Por isso, iremos aprender agora como visualizar estes dados. O comando a ser utilizado é o **SELECT**, veja a sintaxe dele abaixo:

```
SELECT {campo(s)} FROM {tabela(s)}
WHERE {condição}
ORDER BY {campo(s)}
GROUP BY {campo(s)}
```

8.4 - Esclarecendo as informações da sintaxe a cima:

SELECT comando para se chamar a visualização de registros campo(s) campos da tabela, para referência da visualização dos registros.

FROM chamada para indicar a tabela ou tabelas a serem utilizadas.

WHERE chamada para uma condição que deve ser verdadeira para que os registros sejam visualizados.

ORDER BY permite ordenar a visualização de registros em função de um campo especifico.

GROUP BY permite agrupar a visualização de registros em função de um campo especifico

Obs: O "*" (asterisco), terá um papel importante neste comando.

Exemplo: Consulta para pesquisar todos os alunos da cidade de Votuporanga.

1) Clique na tabela **ALUNOS**, e em seguida no link de menu SQL.

2) Digite a instrução abaixo do quadro “Fazer procura(s) SQL no banco de dados escola”.

```
SELECT *
FROM alunos
WHERE cidade = "Votuporanga"
```

3) Clique no botão Executar, e você terá como resultado da pesquisa a figura a seguir:

matricula	nome	endereco	cidade	codCurso
5	Claudinelson Tavares	Rua Novo Horizonte	Votuporanga	2
7	Elvira Savana Alencar	Rua Inglaterra, 496	Votuporanga	3
9	Milena Fatality	Rua Outworld, 43	Votuporanga	2

Exemplo: Consulta para pesquisar todos os alunos ordenados por ordem de curso.

1) Digite o seguinte comando SQL e execute

```
SELECT *
FROM alunos
ORDER BY codcurso
```

8.4.1 - Outros exemplos de consulta

1) Pesquisar o nome do aluno de matrícula = 10008.

```
SELECT nome
FROM alunos
WHERE matricula = "10008"
```

2) Selecionar todos os dados dos alunos com número de matrícula maior que 10011.

```
SELECT *
FROM alunos
WHERE matricula >10011
```

3) Visualizar nome e endereço dos alunos que moram em São Paulo.

```
SELECT nome, endereco
FROM alunos
WHERE cidade = "Jales"
```

4) Visualizar todos os dados da tabela cursos.

```
SELECT *
FROM cursos
```

5) Visualizar todos os dados do curso de código = “01”

```
SELECT *
FROM cursos
WHERE codcurso = "01"
```

8.5 - Visualizando dados de mais de uma tabela

Para que isto ocorra, será **fundamental** o campo chave (chave primária), que ajudará a fazer a ligação com um campo comum de outra tabela (chave estrangeira), mas que tem o mesmo tipo de dado.

Sintaxe:

```
select TABELA1.campo, TABELA2.campo
from TABELAS
where TABELA1.campo_chave = TABELA2.campo_comum and TABELA.campo =
valor;
```

Exemplo: Consulta que exibe todos os alunos que frequentam cursos que frequentam o curso de Auxiliar de Informática.

```
SELECT alunos . * , cursos.nome
FROM alunos, cursos
WHERE cursos.codcurso = alunos.codcurso
AND cursos.nome = "Auxiliar de Informática"
ORDER BY alunos.nome
```

Resultado da consulta:

```
SELECT alunos . * , cursos.nome
FROM alunos, cursos
WHERE cursos.codcurso = alunos.codcurso
AND cursos.nome = "Auxiliar de Informática"
ORDER BY alunos.nome
LIMIT 0 , 30
```

Perfil [I

Mostrar : 30 row(s) starting from row # 0 no modo horizontal células

+ Opções

matricula	nome	endereco	cidade	codcurso	nome
10012	Ana Paula Moreira	Rua do Pinhal, 332	Urânia	03	Auxiliar de Informática
10011	Letícia Savana Silva	Rua do Pinhal, 332	Jales	03	Auxiliar de Informática
10014	Richardson Lopes Souza	Rua Asfaltada, 9321	Pontalinda	03	Auxiliar de Informática
10013	Valéria Elvira dos Reis	Rua Inglaterra, 496	Jales	03	Auxiliar de Informática

8.5.1 - Alterando dados

Você poder alterar uma informação através do comando UPDATE, veja a sintaxe abaixo:

```
UPDATE tabela
SET campo = valor
WHERE {condição};
```

Vamos a uma explicação prática desta sintaxe:

O exemplo abaixo atualiza a tabela alunos, altera a cidade para “Marinópolis” quando a matrícula for igual a “10002”.

```
UPDATE alunos
SET cidade="Marinópolis"
WHERE matricula="10002"
```

Executando essa consulta a cidade será alterada para Mogi Mirim, quando o MySQL encontrar uma ocorrência de matrícula igual a “10002”. Visualize a tabela depois de alterado para observar a atualização.

10002	Maria Conceição Lopes	Rua Pará, 23	Marinópolis	01

8.5.3 - Excluindo Dados

Para excluir uma informação do Banco de Dados, utilizamos o comando **DELETE**, veja a sintaxe a seguir:

```
DELETE FROM tabela
WHERE condição;
```

Excluindo o aluno de Matrícula 10001

```
DELETE FROM alunos
WHERE matricula="10001"
```

OBS: indique sempre a condição para evitar possíveis desastres.

9. Fazendo o PHP se comunicar com o MySQL

Vamos criar, agora, os scripts que farão a comunicação com o banco de dados. Primeiramente crie uma pasta, denominada **aplicacao**.

9.1 - Páginas HTML do Sistema de Cadastros

Nosso objetivo é criar um pequeno sistema de gerenciamento de escola, o ponto inicial será a criação de um arquivo, tipo **PHP**, que será a página principal da aplicação.

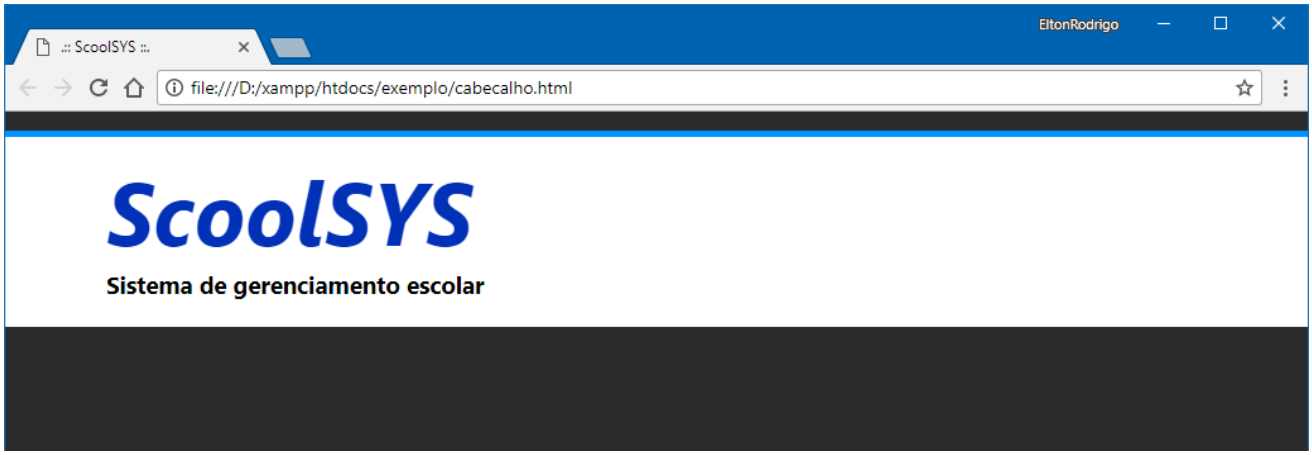
Inicie um novo projeto **PHP**, digite o código abaixo:

```
<?php
    include "cabecalho.html";
    include "menu.html";
    include "conteudo.php";
    include "rodape.html";
?>
```

O comando `include` irá concatenar os três arquivos respectivamente em uma página.

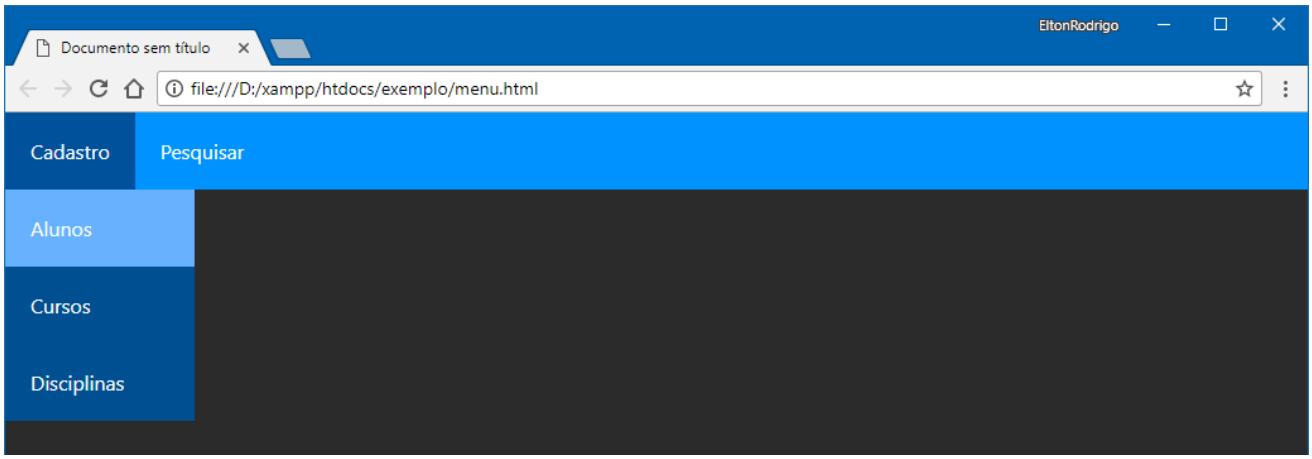
Salve o arquivo com o nome **index.php**.

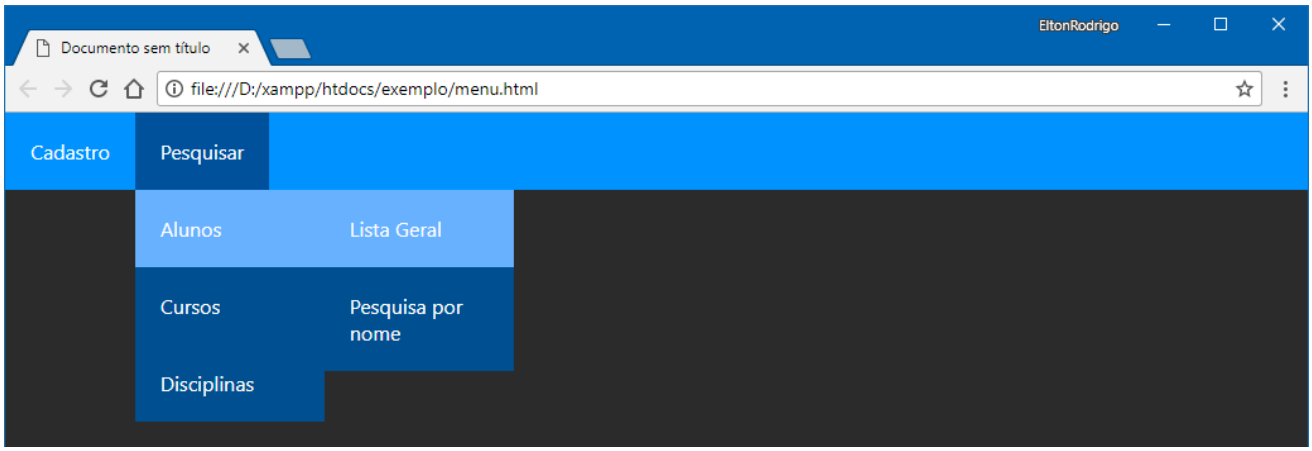
Crie o arquivo **cabecalho.html** como mostra a figura abaixo:



```
<div id="topo">

<h1>ScoolSYS</h1>
<h3>Sistema de gerenciamento escolar</h3>

</div>
```

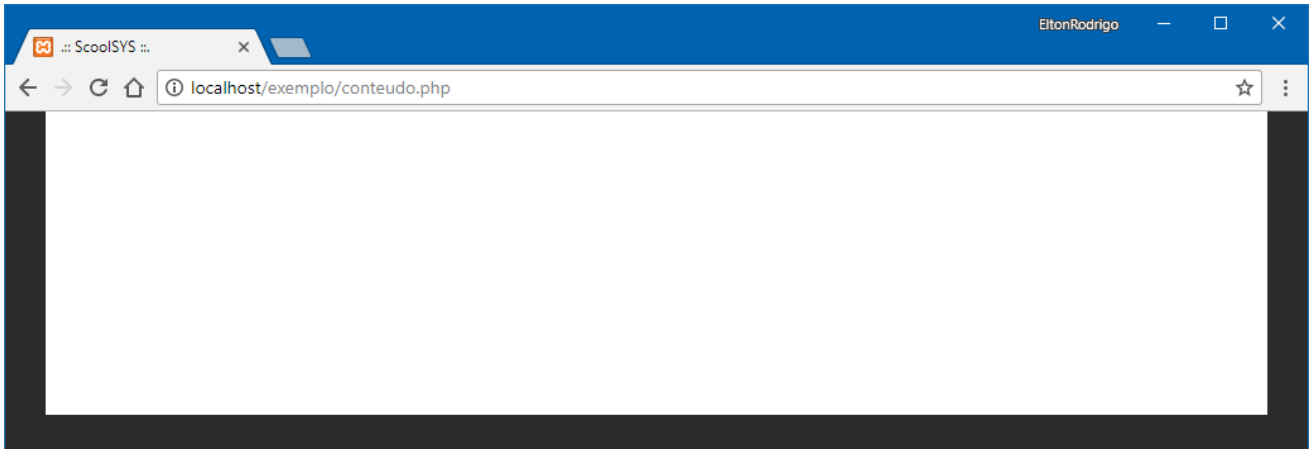
Crie o arquivo **menu.html**.





```
<div class="menu-container">
    <ul class="menu clearfix">
      <li><a href="#">Cadastro</a>
        <ul class="sub-menu">
                    <li><a href="#">Alunos</a></li>
                    <li><a href="#">Cursos</a></li>
                    <li><a href="#">Disciplinas</a></li>
            </ul>
      </li>
        <li><a href="#">Pesquisar</a>
               <ul class="sub-menu">
                   <li><a href="#">Alunos</a>
                       <ul class="sub-menu">
                           <li><a href="#">Lista Geral</a></li>
                           <li><a href="#">Pesquisa por nome</a></li>
                           </ul>
                   </li>
                     <li><a href="#">Cursos</a>
                        <ul class="sub-menu">
                           <li><a href="#">Lista Geral</a></li>
                           <li><a href="#">Pesquisa por nome</a></li>
                       </ul>
                     </li>
                     <li><a href="#">Disciplinas</a>
                     <ul class="sub-menu">
                           <li><a href="#">Lista Geral</a></li>
                           <li><a href="#">Pesquisa por nome</a></li>
                     </ul>
                     </li>
               </ul>
        </li>
    </ul>
</div>
```

Crie o arquivo **conteudo.php**. Será apenas uma div vazia, por enquanto.



```
<div id="conteudo">


</div>
```

E por fim, crie o arquivo **rodape.html**

```
<div id="rodape">
        <h4>Copyright ® 2015 "Seu nome aqui", TODOS OS DIREITOS RESERVADOS.</h4>

    </div>
```

Segue abaixo o código CSS **estilos.css**.

```
@charset "utf-8";
/* CSS Document */

*{
 margin:0;
 border:0;
 padding:0;
 }

body{
 background-color:#2B2B2B;
 }

#topo{
 padding-left:80px;
 height:150px;
 background-color:#FFFFFF;
 border-top:#0093FF 5px solid;
 margin-top:15px;
 }

#topo h1{
 padding-top:10px;
 font-family:Segoe, "Segoe UI", "DejaVu Sans", "Trebuchet MS",
Verdana, sans-serif;
 font-size:70px;
 color:#0031B7;
 font-style:italic;
 }

#topo h3{
 font-family:Segoe, "Segoe UI", "DejaVu Sans", "Trebuchet MS",
Verdana, sans-serif;
 }
```

```
#conteudo{
 width:80%;
 margin:0 auto;
 background-color:#fff;
 padding:50px;
 height:auto;
 font-family:Gotham, "Helvetica Neue", Helvetica, Arial, sans-serif;
 font-size:20px;
 }

.campo{
 font-family:"Lucida Grande", "Lucida Sans Unicode", "Lucida Sans",
"DejaVu Sans", Verdana, sans-serif;
 font-size:20px;
 color:#535353;
 border:#585858 solid 1px;
 border-radius:15px;
 padding:8px;
}

.botao{
 padding:15px;
 font-size:20px;
 background-color:#0070D9;
 color:#FFF;
 font-family:Gotham, "Helvetica Neue", Helvetica, Arial, sans-serif;
 border-radius:15px;
 transition:0.5s;
 }

.botao:hover{
 background-color:#5AAAFF;
 }


textarea:focus, input:focus, select:focus {
    outline: 0;
}
/*------------MENU----------------*/
.clearfix:after{
    content: '.';
    display: block;
    clear: both;
    height: 0;
    line-height: 0;
    font-size: 0;
    visibility: hidden;
    overflow: hidden;

}

.menu{
 margin-bottom:20px;
```

```
 }

.menu,
.sub-menu {
    margin: 0;
    padding: 0;
 margin-bottom:20px;
 transition:0.5s;
 }

.clearfix:after{
    content: '.';
    display: block;
    clear: both;
    height: 0;
    line-height: 0;
    font-size: 0;
    visibility: hidden;
    overflow: hidden;
}

.menu,
.sub-menu {
    list-style: none;
    background: #0093FF;
}

.sub-menu {
    background:#004F90;
}

.menu a {
    text-decoration: none;
    display: block;
    padding: 20px;
    color: #fff;
    font-family:Segoe, "Segoe UI", "DejaVu Sans", "Trebuchet MS",
Verdana, sans-serif;
    font-size: 16px;
}

.menu li {
    position: relative;
 transition:0.5s;
}

.menu > li {
    float: left;
 transition:0.5s;
}

.menu > li:hover {
    background:#00519C;
}
```

```
.menu li:hover > .sub-menu {
    display: block;
}

.sub-menu {
    display: none;
    position: absolute;
    min-width: 150px;
}

.sub-menu li:hover {
    background:#67B1FF;
}

.sub-menu .sub-menu {
    top: 0;
    left: 100%;
 transition:0.5s;
}
/*------------MENU----------------*/

#rodape{
 margin-top:30px;
 height:50px;
 background-color:#8C8C8C;
 font-family:Segoe, "Segoe UI", "DejaVu Sans", "Trebuchet MS",
Verdana, sans-serif;
 color:#fff;
 line-height:50px;
 }

#rodape h4{
 font-weight:normal;
 text-align:center;
 }

table{
 margin-top:20px;
 width:100%;
 }

th{
 line-height:50px;
 }

tr{
 line-height:40px;
 }
```

Visualize o arquivo **index.php** no navegador e terá o seguinte resultado:

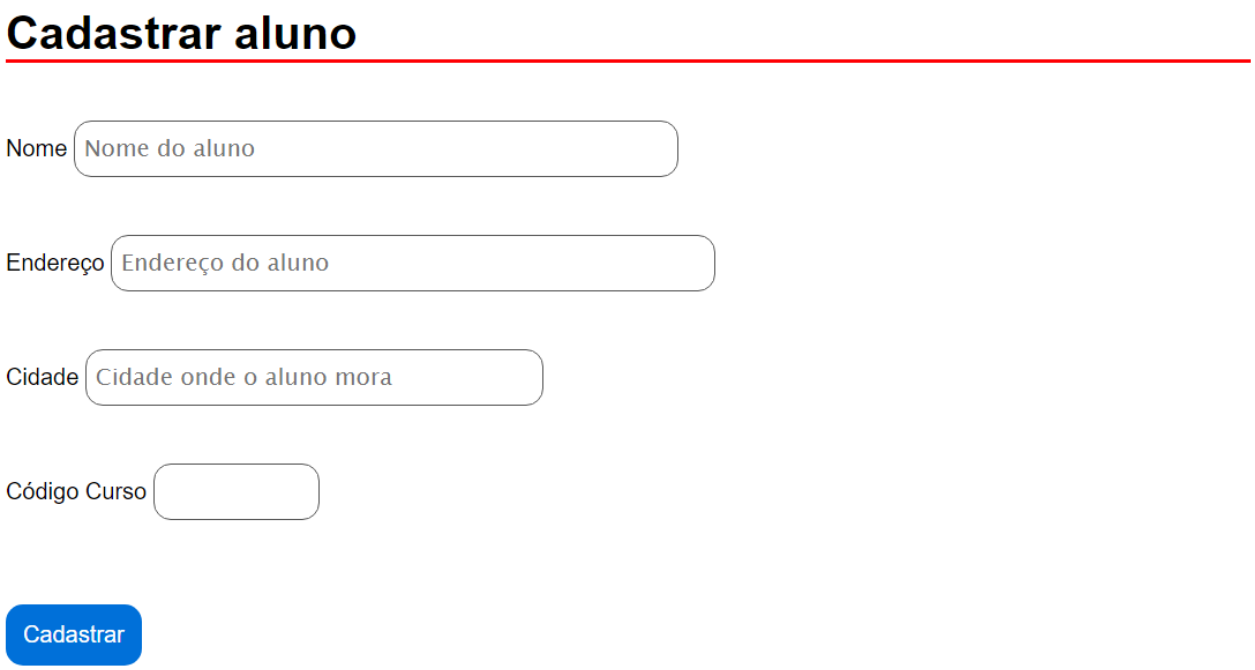
Agora vamos criar as demais páginas, html, necessárias ao projeto, criando a página cadastro de alunos (**formcadaluno.html**)

3) Inicie um novo projeto HTML, digite o código abaixo:

```
<body>

<h2>Cadastrar aluno</h2>

<form action="cadaluno.php" method="post">

    Nome <input type="text" size="40" name="txtnome" class="campo" placeholder=
"Nome do aluno" required><br>
    Endereço <input type="text" size="40" name="txtendereco" class="campo"
placeholder="Endereço do aluno" required><br>
    Cidade <input type="text" size="30" name="txtcidade" class="campo" placeholder=
"Cidade onde o aluno mora" required><br>
    Código Curso <input type="text" size="10" name="txtcodcurso" class="campo"
required><br><br>

    <input type="submit" value="Cadastrar" class="botao">

</form>

</body>
```

Repare que na propriedade *action* da tag *form* já foi definido o nome do arquivo **cadaluno.php** que irá receber os dados preenchidos no formulário.

Salve na mesma pasta que os demais arquivos. Ele vai ter a aparência da imagem abaixo:

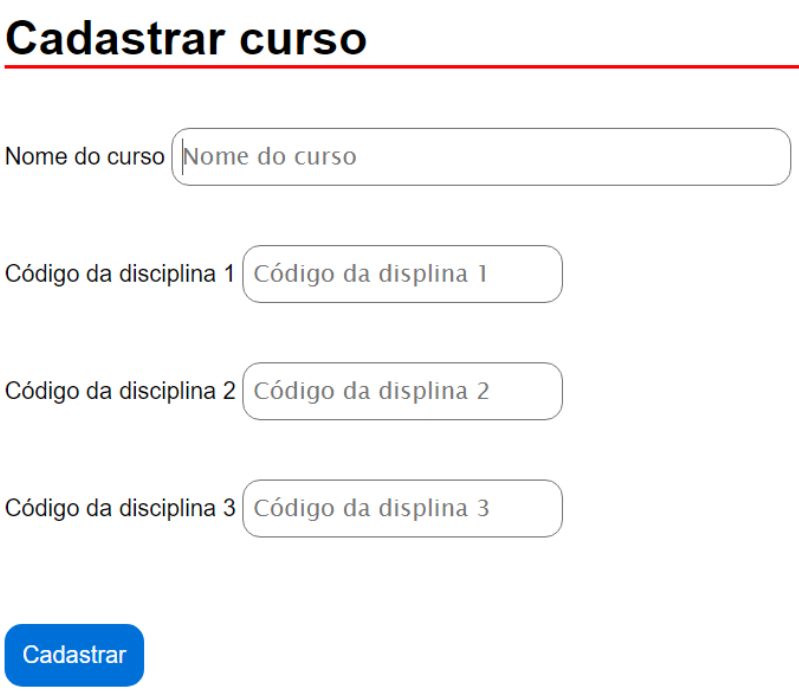


Vamos, agora, criar a página de cadastro de curso (**formcadcurso.html**). Inicie um novo projeto HTML, digite o código abaixo:

```
<body>

<h2>Cadastrar curso</h2>

<form action="cadcurso.php" method="post">

    Nome do curso <input type="text" size="40" name="txtnome" class="campo"
placeholder="Nome do curso" required><br>
    Código da disciplina 1 <input type="text" size="20" name="txtcoddisc1"
class="campo" placeholder="Código da displina 1" required><br>
    Código da disciplina 2 <input type="text" size="20" name="txtcoddisc2"
class="campo" placeholder="Código da displina 2" required><br>
    Código da disciplina 3 <input type="text" size="20" name="txtcoddisc3"
class="campo" placeholder="Código da displina 3" required><br><br>

    <input type="submit" value="Cadastrar" class="botao">

</form>

</body>
</html>
```

Repare que na propriedade *action* da tag *form* também foi definido o nome do arquivo **cadcurso.php** que irá receber os dados preenchidos no formulário.

Salve na mesma pasta que os demais arquivos.



Vamos agora criar a página de cadastro de disciplinas. Inicie um novo projeto HTML, digite o código abaixo:

```
<body>

<h2>Cadastrar disciplina</h2>

<form action="caddisciplina.php" method="post">

    Nome da disciplina <input type="text" size="40" name="txtnome" class=
"campo" placeholder="Nome da disciplina" required><br><br>

    <input type="submit" value="Cadastrar" class="botao">

</form>

</body>
```

Repare que na propriedade *action* da tag *form* também foi definido o nome do arquivo **caddisciplina.php** que irá receber os dados preenchidos no formulário.

Salve na mesma pasta que os demais arquivos.

Cadastrar disciplina

Nome da disciplina Nome da disciplina

Cadastrar

9.2 - Scripts PHP de paginação

Com as páginas de formulário de cadastro prontas, vamos realizar a inclusão delas no site. Abra o arquivo **conteudo.php** (aquele com a div vazia) e insira a estrutura **switch** / **case** abaixo dentro da div:

```
<div id="conteudo">

    <?php
        $paginas = (isset( $_GET["pg"] )  ? $_GET["pg"] : null);

        switch($paginas){

            case "cadaluno":
            include "formcadaluno.html";
            break;

            case "cadcurso":
            include "formcadcurso.html";
            break;

            case "caddisciplina":
            include "formcaddisciplina.html";
            break;

            }
    ?>

</div>
```

Foi declarada uma variável $paginas recebendo uma string “pg” pelo método GET. O valor recebido é verificado pela função isset( ) para que não exiba um aviso de erro ao verificar o valor da variável.

Repare que cada case está verificando um conteúdo string da variável $paginas. Esse mesmo conteúdo será atribuído no valor do href de cada item do menu do site.

Abra o arquivo menu.html e adicione os valores do href `?pg=cadaluno` no item **Alunos**, `?pg=cadcurso` no item **Cursos** e `?pg=caddisciplina` no item **Disciplinas**, como mostra a imagem abaixo:

```
<div class="menu-container">
    <ul class="menu clearfix">
      <li><a href="#">Cadastro</a>
        <ul class="sub-menu">
                    <li><a href="?pg=cadaluno">Alunos</a></li>
                    <li><a href="?pg=cadcurso">Cursos</a></li>
                    <li><a href="?pg=caddisciplina">Disciplinas</a></li>
            </ul>
      </li>
        <li><a href="#">Pesquisar</a>
```

O valor “pg” será enviado pelo método GET e armazenado na variável $paginas no arquivo **conteudo.php** que será verificado de acordo com cada instancia para o carregamento de cada página.

9.3 - Script PHP de conexão com banco de dados

Para que possamos realizar os cadastros, e as pesquisas no Banco de Dados **Escola** precisamos, inicialmente, realizar a conexão com o Banco de Dados propriamente dito.

Inicie um novo projeto **PHP** e após digitar o código abaixo, salve com o nome **conectar.php**.

```php
<?php

$servidor = "localhost";
$usuario  = "root";
$senha    = "";
$db       = "escola";

$con = mysqli_connect($servidor, $usuario, $senha, $db);

if(!$con){
        die("Falha na conexao: " . mysqli_connect_error());
    }else{

      echo "Conexao realizada com sucesso";
    }

?>
```

Observação: *Este script será utilizado pelos demais scripts que necessitaram se comunicar com o banco de dados.*

9.4. Scripts de envio de Dados dos Cadastros

Agora, crie 03 novos scripts para envio de dados dos cadastros anteriormente criados, ou seja, **Alunos**, **Cursos** e **Disciplinas** ao nosso banco de dados **escola**.

Observação: *Observe que o* ***form action,*** *de cada um dos arquivos* ***html*** *criados para os cadastros, apontam para os* ***scripts php*** *que criaremos a partir de agora.*

Inicie um novo projeto **PHP** e após digitar o código abaixo, salve com o nome **cadaluno.php** dentro da pasta.

```php
<?php

include "conectar.php";

$nome      = $_POST["txtnome"];
$endereco  = $_POST["txtendereco"];
$cidade    = $_POST["txtcidade"];
$codcurso  = $_POST["txtcodcurso"];


$sql = "INSERT INTO alunos (nome,endereco,cidade,codcurso) VALUES
('$nome','$endereco','$cidade','$codcurso')";

mysqli_query($con, $sql);

    if(mysqli_affected_rows($con) != 0){
                echo "Usuario cadastrado com Sucesso.";
            }else{
                echo "O Usuario não foi cadastrado com Sucesso.";
            }

?>
```

O comando `$_POST` representa uma variável '**superglobal**'. Significa que ela está disponível em todos os níveis (escopo) de um script. Já, a instrução `include` (inclui e avalia o arquivo informado). nesse caso o de conexão ao Banco de Dados Escola.

Observação: Caso ocorra erro de acentuação nos registros inseridos na tabela do banco de dados, insira as linhas com a codificação de caracteres em UTF-8 no arquivo **conectar.php** abaixo da linha da função `mysqli_connect`, como mostra a imagem abaixo:

```
$db       =  'banco_s' ;

$con = mysqli_connect($servidor, $usuario, $senha, $

$con->query("SET NAMES 'utf8'");
$con->query("SET character_set_connection=utf8");
$con->query("SET character_set_client=utf8");
$con->query("SET character_set_results=utf8");


if(!$con){
```

Inicie um novo projeto **PHP** e após digitar o código abaixo, salve com o nome **cadcurso.php**.

```
<?php

include "conectar.php";

$nome      = $_POST["txtnome"];
$coddisc1  = $_POST["txtcoddisc1"];
$coddisc2  = $_POST["txtcoddisc2"];
$coddisc3  = $_POST["txtcoddisc3"];


$sql = "INSERT INTO curso (nome,coddisc1,coddisc2,coddisc3) VALUES
('$nome','$coddisc1','$coddisc2','$coddisc3')";

mysqli_query($con, $sql);

    if(mysqli_affected_rows($con) != 0){
                echo "Curso cadastrado com Sucesso.";
            }else{
                echo "O Curso não foi cadastrado com Sucesso.";
            }

?>
```

Inicie um novo projeto **PHP** e após digitar o código abaixo, salve com o nome **caddisciplina.php**.

```php
<?php

include "conectar.php";

$nomedisc = $_POST["txtnome"];


$sql = "INSERT INTO disciplinas (nomedisciplina) VALUES ('$nomedisc')";

mysqli_query($con, $sql);

    if(mysqli_affected_rows($con) != 0){
                echo "Disciplina cadastrada com Sucesso.";
            }else{
                echo "A Disciplina não foi cadastrada com Sucesso.";
            }

?>
```

9.5 Exibição de dados

Vamos criar as páginas, HTML, necessárias para exibição das tabelas do nosso banco de dados.

Inicie um novo projeto **PHP**, e após digitar o código abaixo, salve com o nome **lista_alunos.php**.

```php
<body>

<h2>Lista geral de alunos cadastrados</h2>

<?php

include "conectar.php";

$sql = "SELECT * FROM alunos order by nome";

$query = $con->query($sql);

?>
<table width="100%" border="0">
  <tbody>
    <tr>
      <th>Matrícula</th>
      <th>Nome do Aluno</th>
      <th>Endereço</th>
      <th>Cidade</th>
      <th>Cód. Curso</th>
    </tr>
```

```
<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>
    <tr>
      <td><?php echo $dado['matricula'];?></td>
      <td><?php echo $dado['nome'];?></td>
      <td><?php echo $dado['endereco'];?></td>
      <td><?php echo $dado['cidade'];?></td>
      <td><?php echo $dado['codCurso'];?></td>

 <?php
}

 ?>
  </tbody>
</table>

</body>
```

O script acima recebe o resultado da pesquisa de dados de aluno encontrados na tabela **alunos** do banco de dados **Escola**.

A variável `"$query"`, por sua vez, irá receber todos os dados encontrados nesta conexão.

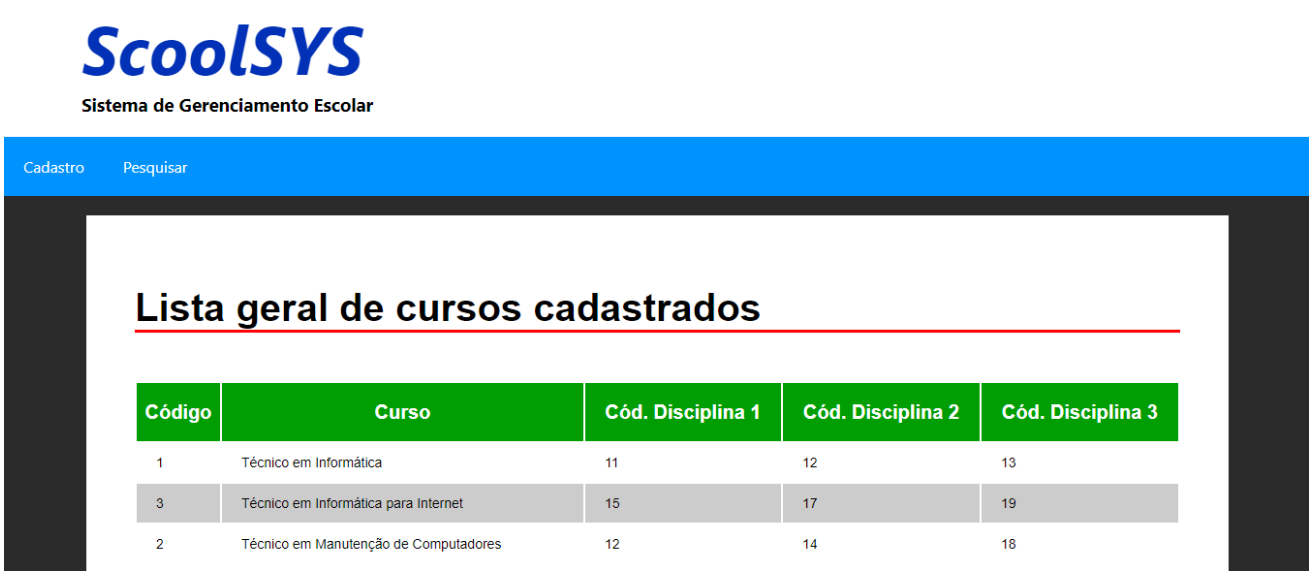
A função `"fetch_assoc"`, nativa do PHP, retorna uma matriz que corresponde à linha buscada, ou FALSE se não houver linhas.

A imagem abaixo ilustra o resultado do código acima.

Inicie um novo projeto **PHP**, e após digitar o código abaixo, salve com o nome **lista_cursos.php**.

```
<body>

<h2>Lista geral de cursos cadastrados</h2>

<?php

include "conectar.php";

$sql = "SELECT * FROM cursos order by nome";

$query = $con->query($sql);

?>
<table width="100%" border="0">
  <tbody>
    <tr>
      <th>Código</th>
      <th>Curso</th>
      <th>Cód. Disciplina 1</th>
      <th>Cód. Disciplina 2</th>
      <th>Cód. Disciplina 3</th>
    </tr>

<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>

    <tr>
      <td><?php echo $dado['codCurso'];?></td>
      <td><?php echo $dado['nome'];?></td>
      <td><?php echo $dado['codDisc1'];?></td>
      <td><?php echo $dado['codDisc2'];?></td>
      <td><?php echo $dado['codDisc3'];?></td>
    </tr>
 <?php
}

 ?>
  </tbody>
</table>

</body>
```

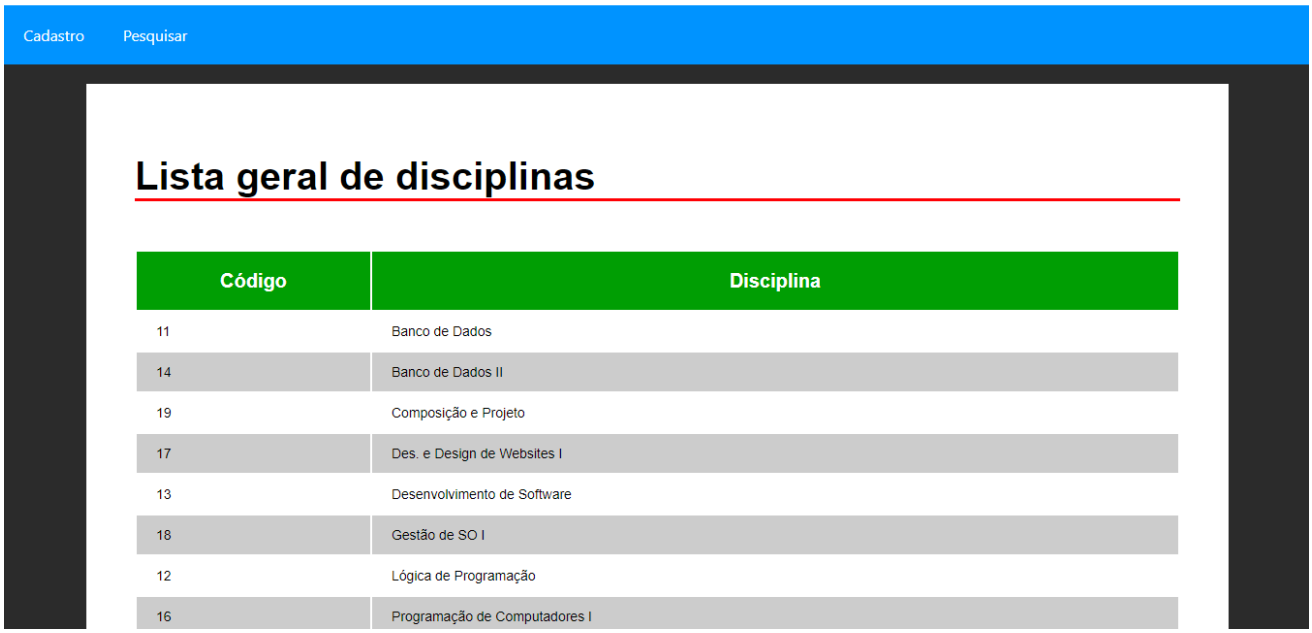
A imagem abaixo ilustra o resultado do código acima.



Inicie um novo projeto **PHP**, e após digitar o código abaixo, salve com o nome **lista_disciplinas.php**.

```
<body>
<h2>Lista geral de disciplinas</h2>
<?php
include "conectar.php";
$sql = "SELECT * FROM disciplinas order by nomedisciplina";
$query = $con->query($sql);
?>
<table width="100%" border="0">
    <tr>
      <th>Código</th>
      <th>Disciplina</th>
    </tr>
<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>
    <tr>
      <td><?php echo $dado['coddisciplina'];?></td>
      <td><?php echo $dado['nomedisciplina'];?></td>
    </tr>
 <?php
}
 ?>
</table>
</body>
```

A imagem abaixo ilustra o resultado do código acima.



Agora insira no arquivo **menu.html** a referencia dessas três páginas nos sub menus **Lista geral** de cada item correspondente. Veja abaixo:

```
<li><a href="#">Pesquisar</a>
        <ul class="sub-menu">
            <li><a href="#">Alunos</a>
                <ul class="sub-menu">
                    <li><a href="?pg=lista_alunos">Lista Geral</a></li>
                    <li><a href="#">Pesquisa por nome</a></li>
                    </ul>
            </li>
              <li><a href="#">Cursos</a>
                 <ul class="sub-menu">
                    <li><a href="?pg=lista_cursos">Lista Geral</a></li>
                    <li><a href="#">Pesquisa por nome</a></li>
                </ul>
              </li>
              <li><a href="#">Disciplinas</a>
              <ul class="sub-menu">
                    <li><a href="?pg=lista_disciplinas">Lista Geral</a></li>
                    <li><a href="#">Pesquisa por nome</a></li>
```

Acrescente também os itens na estrutura switch/case da página **conteudo.php**.

```php
$paginas = (isset($_GET["pg"]) ? $_GET["pg"]: null);

switch($paginas){

    case "cadaluno":
    include "formcadaluno.html";
    break;

    case "cadcurso":
    include "formcadcurso.html";
    break;

    case "caddisciplina":
    include "formcaddisciplina.html";
    break;

    case "lista_alunos":
    include "lista_alunos.php";
    break;

    case "lista_cursos":
    include "lista_cursos.php";
    break;

    case "lista_disciplinas":
    include "lista_disciplinas.php";
    break;
```

9.5 Sistema de Pesquisas

Vamos criar as páginas, HTML, necessárias para o nosso sistema de pesquisa por critérios de nome. Faremos isto para as três tabelas (Alunos, Cursos e Disciplinas).

Inicie um novo projeto **HTML**, e após digitar o código abaixo, salve com o nome **pesqalu_nome.html**.

```html
<h2>Pesquisar dados de Aluno</h2>

<form action="pesqalu_nome.php" method="post">

    Nome do Aluno <input type="text" size="40" name="txtnome" class="campo"
placeholder="Nome do aluno" required>

    <input type="submit" value="Buscar" class="botao">

</form>
```

Inicie um novo projeto **HTML** e após digitar o código abaixo, salve com o nome **pesqcur_nome.html**.

```
<h2>Pesquisar Cursos</h2>

<form action="pesqcur_nome.php" method="post">

    Curso <input type="text" size="40" name="txtnome" class="campo"
placeholder="Nome do curso" required>

    <input type="submit" value="Buscar" class="botao">

</form>
```

Inicie um novo projeto **HTML**, e após digitar o código abaixo, salve com o nome **pesqdisc_nome.html**.

```
<h2>Pesquisar dados das Disciplinas</h2>

<form action="pesqdisc_nome.php" method="post">

    Nome da disciplina <input type="text" size="40" name="txtnome" class="campo"
placeholder="Nome do aluno" required>

    <input type="submit" value="Buscar" name="btnEnviar" class="botao">

</form>
```

9.6 Script PHP do Sistema de Pesquisa

Abaixo, os 03 scripts necessários para a realização de todas as consultas.

Inicie um novo projeto **PHP**, digite o código abaixo, e salve como **pesqalu_nome.php** dentro da pasta **aplicacao**.

```
<body>

<?php
include "conectar.php";
?>
<div id="conteudo">
<h2>Pesquisar dados de Aluno</h2>
<?php
$nome = $_POST["txtnome"];

$sql = "SELECT * FROM alunos where nome like '%$nome%' order by nome";

$query = $con->query($sql);

?>
```

```
<table width="100%" border="0">
  <tbody>
    <tr>
      <th>Matrícula</th>
      <th>Nome do Aluno</th>
      <th>Endereço</th>
      <th>Cidade</th>
      <th>Cód. Curso</th>
    </tr>

<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>
    <tr>
      <td><?php echo $dado['matricula'];?></td>
      <td><?php echo $dado['nome'];?></td>
      <td><?php echo $dado['endereco'];?></td>
      <td><?php echo $dado['cidade'];?></td>
      <td><?php echo $dado['codCurso'];?></td>
    </tr>
 <?php
}

 ?>
  </tbody>
</table>
</div>

</body>
```

Obs.: O script acima recebe o resultado da pesquisa de dados de aluno enviada pelo arquivo **pesqalu_nome.html**.

NOTA: *O comando "**LIKE '%$nome%'**" faz uma busca sobre qualquer parte do nome do aluno, sendo que neste caso ele irá retornar os registros cujo nome case com a string digitada.*

Agora vamos digitar o script para pesquisar dados de cursos.

Inicie um novo projeto **PHP**, digite o código abaixo e salve como **pesqcur_nome.php**.

```
<body>

<?php

include "conectar.php";

?>
<div id="conteudo">
<h2>Resultado de pesquisa de alunos</h2>
<?php
$nome = $_POST["txtnome"];

$sql = "SELECT * FROM cursos where nome like '%$nome%' order by nome";

$query = $con->query($sql);

?>
<table width="100%" border="0">
  <tbody>
    <tr>
      <th>Código</th>
      <th>Curso</th>
      <th>Cód. Disciplina 1</th>
      <th>Cód. Disciplina 2</th>
      <th>Cód. Disciplina 3</th>
    </tr>

<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>

    <tr>
      <td><?php echo $dado['codCurso'];?></td>
      <td><?php echo $dado['nome'];?></td>
      <td><?php echo $dado['codDisc1'];?></td>
      <td><?php echo $dado['codDisc2'];?></td>
      <td><?php echo $dado['codDisc3'];?></td>
    </tr>

 <?php
}

 ?>
  </tbody>
</table>
</div>

</body>
```

Agora para encerrar, vamos digitar o último script, para a pesquisa específica e geral de disciplinas cadastradas.

Inicie um novo projeto **PHP**, digite o código abaixo e salve como **pesqdisc_nome.php**.

```
<body>

<?php
    include "conectar.php";
?>
<div id="conteudo">
<h2>Pesquisar dados das Disciplinas</h2>
<?php

$nome = $_POST["txtnome"];

$sql = "SELECT * FROM disciplinas where nomedisciplina like '%$nome%'
order by nomedisciplina";

$query = $con->query($sql);

?>
<table width="100%" border="1">
    <tr>
      <th>Código</th>
      <th>Disciplina</th>
    </tr>

<?php
//exibe as linhas encontradas na consulta
while ($dado = $query->fetch_assoc()) {
?>
    <tr>
      <td><?php echo $dado['coddisciplina'];?></td>
      <td><?php echo $dado['nomedisciplina'];?></td>

    </tr>
 <?php
}
 ?>

</table>

</body>
```