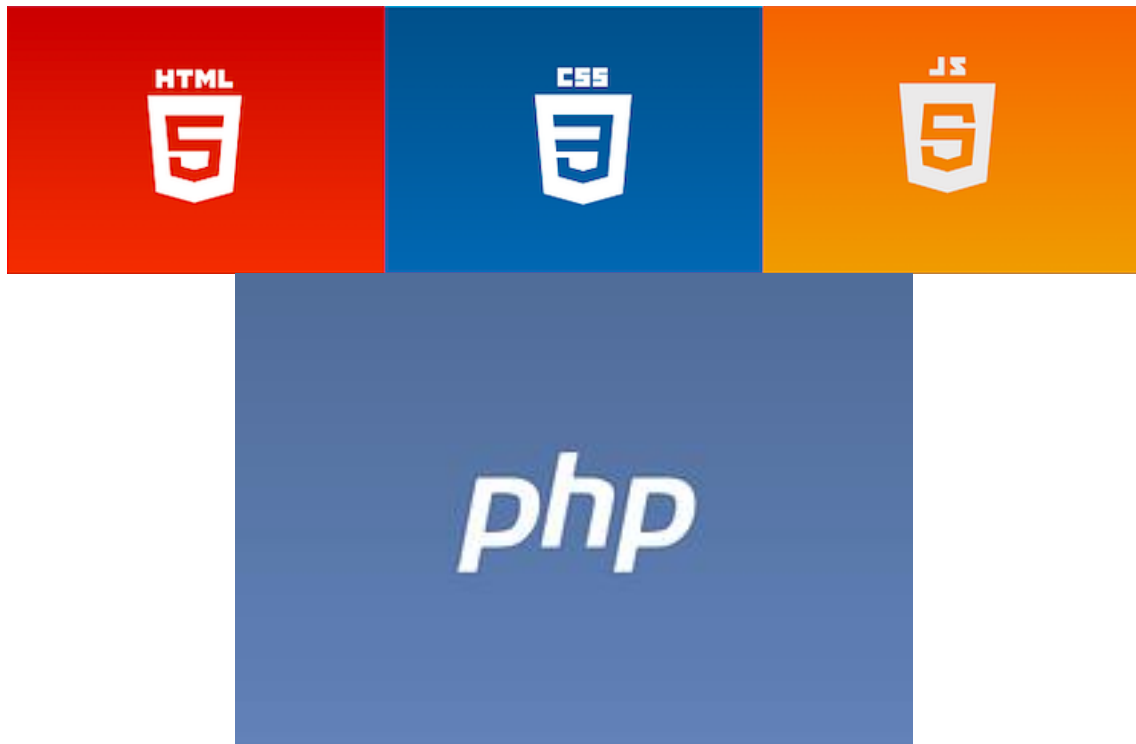


1 - Introdução ao PHP

PHP



Antes de tudo Sejam Bem Vinda(o)s!



Convido vocês a estudarem comigo. No começo algumas palavras podem assustar um pouco. Mas com calma vamos entrar nesse universo e vocês vão ver que tudo passa a ficar tranquilo.

Nós vamos estudar a programação Back End de Sites e Sistemas Web. Para isso é interessante que **antes de começar, você tenha conhecimentos básicos em HTML e Lógica de Programação**. Separei dois cursos que você pode fazer antes de começar os estudos se julgar necessário:

HTML: <https://ead.pbh.gov.br/course/view.php?id=460>

Lógica de Programação: <https://ead.pbh.gov.br/course/view.php?id=554>


Fique tranquilo! Essa é apenas uma recomendação e você pode ficar à vontade para começar nosso curso agora mesmo.

Vamos falar do PHP?

O PHP é uma linguagem de script que trabalha do lado do servidor (**Server-Side**), também conhecido como Back-End. É uma linguagem muito utilizada e bastante popular para criação de aplicações **Web dinâmicas e interativas**. Além disso, é uma ferramenta gratuita.

Existem diversas versões do **PHP** e, por questões de compatibilidade com vários sistemas antigos e mais recentes, nós vamos adotar a versão 7 do PHP.



designed by  freepik

O que significa a sigla PHP?

PHP é uma sigla para **Hypertext Preprocessor**, ou seja, um pré-processador de hipertextos. Hipertextos são: blocos de textos, imagens, palavras, sons etc.

O que é um arquivo PHP?

Os arquivos **PHP** podem conter texto, **HTML**, **CSS**, **JavaScript**, código e, claro, os códigos **PHP**. Se você já está habituado a criar arquivos **HTML** vai verificar que, agora, nós apenas estamos mudando a linguagem, porque o propósito é criar interação na página.

Todos os arquivos PHP que vamos criar devem ter a extensão **.php**. Por exemplo: **index.php**, ao invés do “index.html” que estamos habituados quando criamos páginas apenas com HTML e CSS. Isso é feito para que o servidor possa entender os códigos inseridos no arquivo e fazer o que o código foi programado para fazer.

O que podemos fazer com o PHP?

Coletar dados de um formulário, adicionar, excluir e modificar dados em um banco de dados, criar controle de acesso do usuário ao seu sistema de web, criptografar dados (quando você quer criar uma senha, por exemplo) e muito mais.

Por que utilizar o PHP?

O PHP pode ser executado em várias plataformas, como: **Windows**, **Linux**, **Mac** etc. O que é importante, porque um sistema multiplataforma traz muita flexibilidade para o seu projeto. Além disso, o PHP pode trabalhar com vários tipos de bancos de dados.

Existem vários outros motivos para utilizar o PHP, mas, ao longo de sua jornada como programador, você vai descobrindo tudo isso de forma natural. Então, preferimos deixar isso com você, ok?

1.1 - Instalação do PHP

Para utilizar o PHP é necessário que tenhamos um Software que instale um Servidor Web em nosso computador ou nós podemos utilizar os serviços de Hospedagem de um Servidor na internet.

Link para o Xampp: https://www.apachefriends.org/pt_br/index.html

1.2 Como instalar o VS Code

Para começar a criar nosso Sistema web (aplicação web) ou site para o web utilizando o PHP, é interessante utilizarmos um editor de código fonte mais completo. Por isso, em nosso curso nós vamos adotar o Visual Studio Code. O VS Code é uma ferramenta gratuita desenvolvida pela Microsoft que pode ajudar bastante com a produção de código.

Link para VS Code: <https://code.visualstudio.com/download>

No PHP nós trabalhamos com scripts e eles podem estar em qualquer parte do seu arquivo.

1.3 Sintaxe Básica do PHP

Todos os scripts devem estar entre os delimitadores do PHP

```
1  <?php
2
3  //Seu código será escrito aqui dentro entre esses dois delimitadores na cor vermelha
4
5  ?>
```

Fonte: Própria 2021

Você deve salvar os arquivos criados com a extensão padrão do PHP que é “.php.”

Vamos dar um exemplo de um arquivo PHP, simples, com uma instrução **echo** que imprime uma informação na tela do computador. Mais a frente, ainda vamos falar desta instrução, mas, no momento, preste

apenas atenção em como estamos criando esse arquivo. Olhe os delimitadores do PHP `<?php` `?>` e como o código php está sendo colocado juntamente com o HTML.

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5      <h1>Primeiro Código PHP</h1>
6
7      <?php
8          echo "Olá PHP";
9      ?>
10
11  </body>
12  </html>
13
```

Fonte: Própria 2021

As instruções no PHP sempre terminam com um ponto e vírgula (;).

1.4 Comentários em PHP

Os comentários em nossos códigos de programação, sejam eles em qualquer linguagem, servem para auxiliar nosso trabalho à medida em que nosso código se torna muito extenso. Os comentários não são entendidos como um código que deve ser interpretado e tenha alguma função. Eles estão ali, apenas, para quem tem acesso ao código saber o significado de determinada parte dele.

Em PHP existem três formas de comentar um código:

```

1  <!DOCTYPE html>
2  <html>
3  <body>
4
5      <?php
6
7
8          // Esse é um comentário de apenas uma linha que explica determinada parte do código e será ignorada na hora de processar o código
9
10         echo "Estou aprendendo sobre comentários, ainda não entendo você echo";
11
12     /*
13
14     Esse é um bloco de comentários que
15
16     pode conter diversas linhas que serão
17
18     ignoradas na hora de processar o código
19
20     */
21
22     echo "Calma Echo! Eu ainda estou aprendendo sobre comentários, não entendo você";
23
24     # Esse também é um comentário de apenas uma linha que explica determinada parte do código e será ignorada na hora de processar o código
25
26
27  <?php
28
29  </body>
30 </html>

```

2 - Variáveis em PHP

As linhas onde têm a instrução "**echo**" serão processadas mas as linhas que contêm os caracteres // ou /* ...*/ e #, serão ignoradas pelo interpretador do PHP.

Uma variável no PHP sempre vai começar com o sinal de cifrão (\$) e, depois, seguida pelo seu nome:

\$exemplo = 1;

Nesse pequeno exemplo, estamos criando uma variável que contém o valor: 1;

As variáveis podem ter vários tipos de nomes, é você programador que vai definir o melhor nome. O que você precisar é apenas seguir algumas regras para que a sua variável funcione da maneira correta no seu código.

**Lembre-se: utilize, sempre, nomes que tenham significado para sua variável, assim você não irá se perder enquanto estiver construindo seu código.*

Algumas Regras para variáveis que devem ser seguidas em PHP:

- Uma variável sempre começa com o sinal de cifrão \$ e, depois, pelo nome da variável
- O nome de uma variável deve começar com uma letra ou com um caractere sublinhado
- O nome de uma variável nunca pode começar com um número
- O nome de uma variável pode conter apenas caracteres alfanuméricos e sublinhados (Az, 0-9 e _)
- Os nomes das variáveis diferenciam maiúsculas de minúsculas (\$exemplo e \$EXEMPLO são duas variáveis diferentes).

Alguns exemplos de variáveis válidas:

\$exemplo = 100;

\$EXEMPLO = 200;

\$exemploTres = 300;

\$exemplo4 = 400 ;

\$exemplo_5 = 500;

\$Exemplo6 = 600;

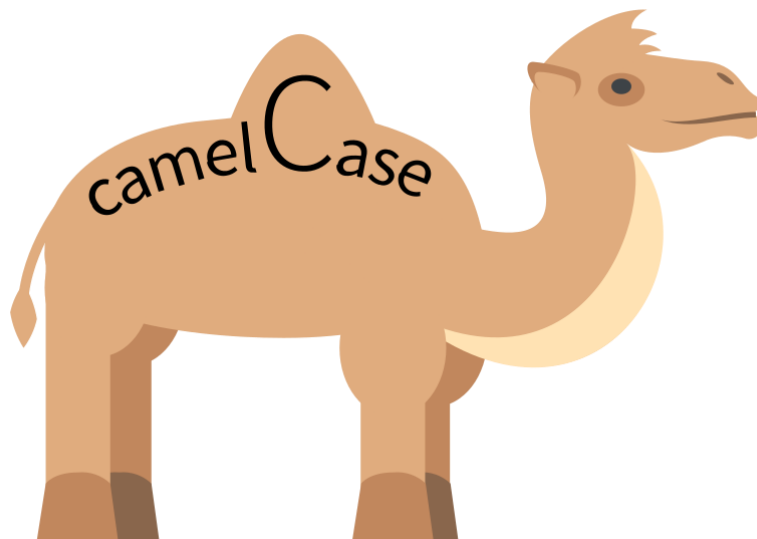
Alguns exemplos de variáveis inválidas:

\$1exemplo = 100;

\$EXEMPLO TRES = 200;

\$exemplo@ = 300;

Em programação de computadores, é muito comum usarmos uma pequena técnica para escrever os nomes de variáveis e outras coisas no código chamado de camelCase. Esse nome é para lembrar das corcovas do camelo. Existem alguns tipos de camelCase e nós gostamos de utilizar o lowerCamelCase, que consiste em colocar em maiúsculo a primeira letra das palavras que vem junto com a primeira palavra que descreve a variável. Como não podemos utilizar espaços, é uma boa prática de programação e ajuda a construir variáveis com nomes significativos.



Fonte: <https://pt.wikipedia.org/> 2021

Por exemplo: quando queremos criar uma variável e deixá-la com um bom nome, podemos utilizar:

```
$nomeCompletoPaciente = "João César de Lima";
```

3 - Comando Echo e Print



Fonte: Pixabay 2021

No PHP podemos utilizar duas instruções para imprimir dados na tela: **Echo** e **Print**. Elas possuem poucas diferenças, mas vamos focar no fato de que **Echo** é, ligeiramente, mais rápida que **Print**. Iremos nos concentrar nisso para não confundir a sua cabeça, porque esse tema pode ser tratado em outro momento, quando você já estiver habituado com a linguagem PHP.

Então, em nosso curso vamos nos dar ao luxo de utilizar apenas **echo**, em nossos exemplos. Se você tiver interesse em saber um pouco mais de **print**, acesse: <https://www.php.net/manual/en/function.print.php>

ECHO

A instrução **echo** pode ser usada com ou sem parênteses:

```
1 <?php
2     echo "<h2>PHP</h2>";
3     echo "Olá PHP<br>";
4     echo ("Olá PHP");
5 ?>
```

Resultado do Código no navegador

PHP

Olá PHP

Olá PHP

Fonte: Própria 2021

Todos esses três exemplos irão funcionar. Repare, podemos utilizar as tags HTML junto com outras coisas que estamos imprimindo, e a saída de tela vai entender que as tags estão presentes para formatar o texto.

PRINT

A instrução **print** pode ser usada com ou sem parênteses também:

```
1 <?php
2     print "<h2>PHP</h2>";
3     print "Olá!<br>";
4     print ("Olá PHP");
5 ?>
```

Resultado do Código no navegador

PHP

Olá!

Olá PHP

Fonte: Própria 2021

O comando **print** também pode conter marcação HTML.

Podemos, também, gerar textos unindo palavras e variáveis

O exemplo abaixo mostra como utilizar o sinal de . (ponto), para concatenar variáveis com as palavras.

```
1 <?php
2     $texto1 = "PHP";
3     $texto2 = "Professor";
4
5     echo "<h2>" . $texto1 . "</h2>";
6     echo "Estudando PHP com o " . $texto2 . " <br>";
7     echo "Lucas";
8 ?>
```

Resultado do Código no navegador

PHP

Estudando PHP com o Professor
Lucas

Fonte: Própria 2021

4 – Tipos de dados em PHP

4.1 - String (Cadeia de Caracteres)

As variáveis podem armazenar diferentes tipos de dados e cada tipo de dado tem uma função. Neste capítulo, nós vamos falar de alguns desses tipos.



Fonte: Pixabay 2021

Uma string é uma sequência de caracteres, como: **Olá, pessoal! Estamos falando sobre Strings.**

Uma string pode conter qualquer texto entre aspas.

Você pode usar aspas simples ou duplas.

```
<?php
$a = "Olá Pessoal!";
$b = 'Estamos falando sobre Strings.';
```

```
echo $a;
echo $b;
?>
```

ou

```
<?php

echo "Olá Pessoal! ";
echo 'Estamos falando sobre Strings.';

?>
```

Em ambos os exemplos estamos imprimindo uma frase: **Olá, Pessoal! Estamos falando sobre Strings.** A diferença é que no primeiro código estamos atribuindo a frase a uma variável e, depois, imprimindo ela, já no segundo código estamos imprimindo a frase diretamente na instrução **echo**.

A forma como você vai utilizar a string vai depender do que você quer fazer com o seu código, mas ambas estão corretas.

Saída dos códigos acima

Olá Pessoal! Estamos falando sobre Strings.

4.2 - Integer (Inteiro)

Os tipos de dados inteiros servem para representar os números não decimais, ou seja, que não possuem vírgula.

Algumas regras para números inteiros:

- Deve possuir pelo menos um dígito
- Não deve possuir vírgula para casas decimais
- Pode ser positivo ou negativo

```
<?php
```

```
$a = 40;
```

```
$b = 30;
```

```
echo $a + $b;
```

```
?>
```

Nesse exemplo eu estamos atribuindo o valor 40 à variável **\$a** e o valor 30 à variável **\$b** e, depois, imprimimos na tela o resultado. A saída desse código no navegador será 70.

4.3 - Float (Número de ponto flutuante)

Um Float (número de ponto flutuante) é um número composto por uma ou mais casas decimais após uma vírgula.

Exemplo:

```
<?php
```

```
$a = 11.55
```

```
?>
```

Lembre-se: em PHP nós vamos sempre utilizar o sinal de ponto nos códigos para representar um número decimal.

4.4 - Boolean

Um dado do tipo boolean (booleano) representa dois estados possíveis: TRUE (verdadeiro) ou FALSE (falso).

```
$v = true;  
$f = false;
```

Os tipos de dados booleanos costumam ser usados em testes condicionais. Mais a frente, iremos estudar sobre estruturas condicionais e você vai entender a importância de utilizar o dado boolean (quando precisar fazer algumas coisas em seu código). Por tanto, fique tranquilo que ainda falaremos mais sobre esse assunto.

4.5 - Array (Matriz)

Quando você precisa inserir diversos valores em uma variável, você pode utilizar um tipo de dados chamado **Array** (matriz). Um **Array** armazena vários valores em uma única só variável.

Por exemplo:

```
<?php  
$carros = array("Fiat Uno", "Gol", "Fiesta");  
?>
```

Isso pode ser muito útil para diminuir a quantidade de código e diminuir o tempo de busca por algum dado. Um exemplo que podemos dar para te ajudar a entender a diferença básica de um **array** e uma variável, seria na hora de declarar o mesmo código que declaramos acima. Se não usamos o **array** para atribuir os valores a variáveis, em nosso código, deveríamos fazer uma estrutura parecida com isso:

```
<?php  
$carros1 = "Fiat Uno";  
$carros2 = "Gol";  
$carros3 = "Fiesta";  
?>
```


Agora, imagine se tivéssemos em meu **array** 1000 registros? teríamos que colocar 1000 variáveis diferentes para cada registro que o **array** possuir. Nesse momento, você pode estar um pouco confuso com o uso de **array**, mas voltamos a dizer, fique tranquilo, pois, agora precisamos que você conheça apenas um pouco desses conceitos. No futuro, tudo isso vai ficar claro para você no decorrer da sua vida como programador.

4.6 - Algumas Informações importantes sobre Tipos de dados em PHP

Existem, em PHP, outros tipos de dados aos quais não trataremos agora, para que isso não venha causar nenhuma confusão na cabeça de vocês. Se você tiver interesse e julgar que isso não vai atrapalhar seu conhecimento você pode

acessar: https://www.w3schools.com/php/php_datatypes.asp.

Sugeriremos que você deixe isso para um outro momento, mas gostaria de deixar registrado que existem outros tipos de dados que você pode trabalhar em PHP.

PHP é uma linguagem fracamente tipada

Observe: até agora, nós não precisamos dizer ao PHP qual é o tipo de dados das variáveis que criamos em nossos exemplos. Isso é porque, apenas com o valor da variável o PHP associa automaticamente o tipo de dados. Porém não quer dizer que você não possa declarar as variáveis no momento de sua criação, só não é uma condição necessária. Fique à vontade, se em algum projeto você encontrar a necessidade de criar uma variável e atribuir a ela um tipo.

Mais uma vez, dizemos: alguns assuntos são deixados para depois, para não atrapalhar o conhecimento do que é mais importante.

var_dump()

Se você quiser saber o tipo de uma variável você pode utilizar o comando **var_dump()**.

Exemplo:

```
<?php
$a = 1.55;
var_dump($a);

$b = "teste";
var_dump($b);

$c = 1;
var_dump($c);
?>
```

35 - Operadores aritméticos em PHP

Os operadores são utilizados, como na matemática, para realizar operações com as variáveis .

Os operadores aritméticos são:

Operador	Função	Exemplo	Resultado
+	Adição	\$a + \$b	Soma de \$a + \$b
-	Subtração	\$a - \$b	Subtração de \$a por \$b
*	Multiplicação	\$a * \$b	Multiplicação de \$a por \$b
/	Divisão	\$a / \$b	Divisão de \$a por \$b
%	Módulo	\$a % \$b	Pegar o valor do resto da divisão de \$a por \$b
**	Expoente	\$a ** \$b	\$a É elevado a \$b

5.1 - Operadores de atribuição em PHP

Os operadores de atribuição são utilizados para atribuir um valor a uma variável ou, se preferir, gravar um valor nessa variável.

O operador de atribuição mais simples é o sinal de igualdade =.

Quando fazemos **a = b** estamos dizendo que o valor de **b** vai ser atribuído à variável **b**.

Nós não estamos dizendo que **a** é igual **a b**. Isso pode parecer um pouco confuso, mas futuramente iremos estudar os operadores de comparação e vamos entender como fazer comparações. Agora, sempre se lembre que o sinal de =, sozinho, significa que o valor a direita vai ser atribuído à variável esquerda.

Exemplo:

```
<?php
```

```
$a;
```

```
$b = 3;
```

```
$a = $b;
```

```
echo $a;
```

```
?>
```

A saída desse código será **3**, porque a variável **\$a** está recebendo o valor de **\$b** que é **3**.

Podemos, também, escrever em PHP as atribuições de duas formas.

Exemplo:

```
$a = $a + $b;
```

```
$a = $a - $b;
```

```
$a = $a * $b;
```

`$a = $a / $b;`

`$a = $a % $b;`

é a mesma coisa que

`$a += $b;`

`$a -= $b;`

`$a *= $b;`

`$a /= $b;`

`$a %= $b;`

Isso são apenas formas mais reduzidas de fazer a mesma coisa, e em programação quanto menos código melhor. Lembre-se disso!

Então, o que fizemos até agora com o sinal de = foi realizar contas e atribuir a variável que está antes do sinal do resultado das contas.

No próximo tópico, você vai entender a diferença entre comparação, atribuição e a utilização do sinal de = de outras formas.

5.2 - Operadores de comparação em PHP

Agora, nós vamos utilizar o sinal de igualdade para fazer comparações, mas quando ele é utilizado para fazer comparações nós devemos utilizá-lo duas vezes seguidas ==.

Os operadores de comparação são muito úteis quando precisamos comparar números ou strings.

==	Igual	\$a == \$b
===	Idêntico	\$a === \$b
!=	Diferente	\$a != \$b
<>	Diferente	\$a <> \$b
!==	Não Idêntico	\$a !== \$b
>	Maior que	\$a > \$b
<	Menor que	\$a < \$b
>=	Maior ou igual	\$a >= \$b
<=	Menor ou igual	\$a <= \$b

Mais a frente, vamos trabalhar com algumas comparações e você vai entender toda a utilização desses sinais de forma clara.

5.3 - Operadores de comparação em PHP

Agora, nós vamos utilizar o sinal de igualdade para fazer comparações, mas quando ele é utilizado para fazer comparações nós devemos utilizá-lo duas vezes seguidas ==.

Os operadores de comparação são muito úteis quando precisamos comparar números ou strings.

Operador	Nome	Exemplo	Resultado
==	Igual	\$a == \$b	Compara se o valor
===	Idêntico	\$a === \$b	Compara se o valor
!=	Diferente	\$a != \$b	Retorna verdadeiro

<>	Diferente	\$a <> \$b	Faz a mesma coisa o
!==	Não Idêntico	\$a !== \$b	Compara se o valor
>	Maior que	\$a > \$b	Verifica se \$a é maio
<	Menor que	\$a < \$b	Verifica se \$a é men
>=	Maior ou igual	\$a >= \$b	Verifica se \$a é maio
<=	Menor ou igual	\$a <= \$b	Verifica se \$a é men

Mais a frente, vamos trabalhar com algumas comparações e você vai entender toda a utilização desses sinais de forma clara.

5.4 - Operadores de incremento e decremento em PHP

Os operadores de incremento são utilizados para incrementar o valor de uma variável.

Os operadores de decremento são utilizados para diminuir o valor de uma variável.

Operador	Nome	Descrição
++\$a	Pré Incremento	Incrementa em \$a o valor de número 1
\$a++	Pós Incremento	Retorna \$a e depois incrementa o valor de 1 em \$a
--\$a	Pré Decremento	Decrementa o valor de 1 na variável \$a
\$a--	Pós Decremento	Retorna \$a e depois decrementa o valor de 1 em \$a

5.5 - Operadores lógicos em PHP

Os operadores lógicos do PHP são usados para combinar declarações condicionais.

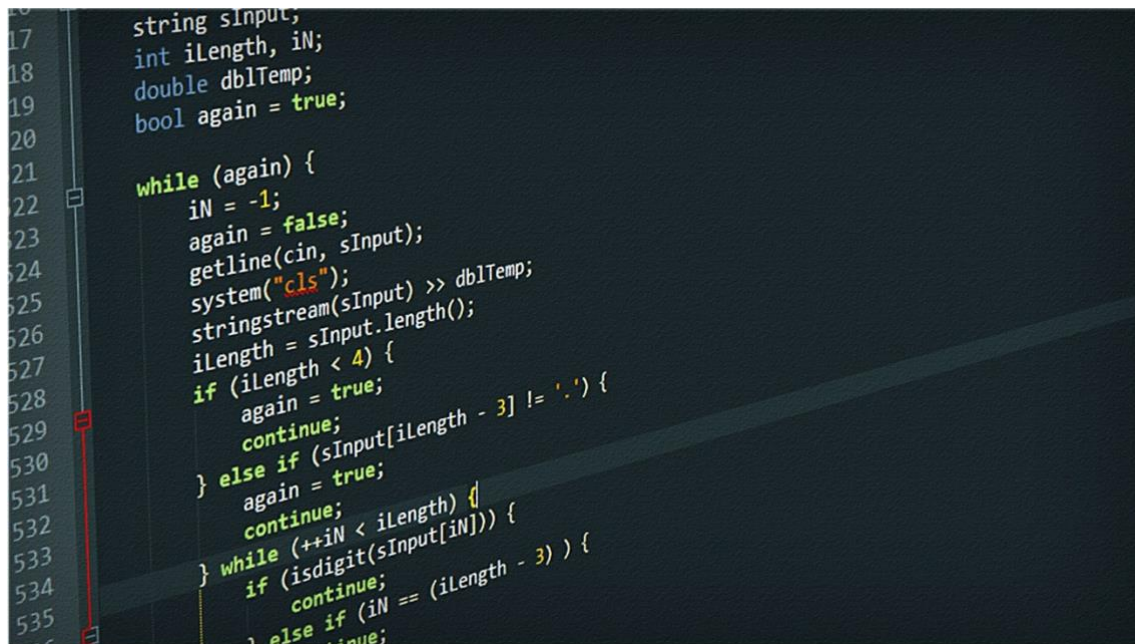
Operador	Nome	Exemplo
and	e	\$a and \$b

or	ou	\$a or \$b
xor	xou	\$a xor \$b
&&	e	\$a && \$b
	ou	\$a \$b
!	Negação	!\$a

6.1 - Comando IF

Instruções condicionais são utilizadas para executar determinada ação, se uma condição previamente definida for atendida

A instrução IF



```

17 string sInput;
18 int iLength, iN;
19 double dblTemp;
20 bool again = true;
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     } while (++iN < iLength) {
36         if (isdigit(sInput[iN])) {
37             continue;
38         } else if (iN == (iLength - 3)) {
39             continue;
40         }
41     }
42 }

```

A instrução IF executa um código definido por você, se a condição que você definiu for atendida.

Exemplo:

```
if($nome == "Lucas") {
```

```
echo "Esse é o nome do Professor";  
}
```

Nesse exemplo, a porção de código `echo "Esse é o nome do Professor"` só vai ser executada se a variável **\$nome** possuir o valor Lucas. Se ela não contiver esse valor, o código vai seguir pelas próximas linhas sem executar essa instrução: `echo "Esse é o nome do Professor";`

Observações:

-Entende-se por bloco de código todo o código entre chaves { }.

-As chaves são opcionais quando o bloco de código tiver apenas uma linha

Pode parecer um pouco confuso, mas na medida em que estivermos fazendo nossos programas você verá que é simples.

Nos próximos tópicos, você vai ter um vídeo disponível falando sobre os operadores condicionais.

6.2 - Comando IF e ELSE

A instrução **IF** e **ELSE** funciona assim: primeiro, a condição estabelecida no comando **IF** é testada. Se ela for atendida o bloco de código do **IF** é executado e o bloco de código do **ELSE** é ignorado. Se a condição que está no **IF** não for atendida, então, o bloco de código do **IF** não é executado e o bloco de código que está no **ELSE** será executado.

Exemplo:

```
<?php  
  
if($hora < 18){  
  
    echo "Tenha um ótimo dia.";  
  
} else {  
  
} echo "Tenha uma ótima noite";  
  
?>
```


6.3 - Comando IF, ELSEIF e ELSE

A instrução **if**, **elseif**, **else** vai primeiramente testar o **IF**. Se a condição do IF for atendida, o bloco de código do **IF** vai ser executado e os blocos de códigos do **ELSEIF** e do **ELSE** vão ser ignorados. Se ela não for atendida, ele vai testar o **ELSEIF** e se o **ELSEIF** for atendido o bloco de código do **ELSEIF** é executado e os outros blocos de códigos são ignorados. Se a condição do **ELSEIF** não for atendida, então, o bloco de código do **ELSE** é que vai ser executado.

```
<?php  
  
if ($hora < 10) {  
    echo "Bom dia!";  
} elseif ($hora < 18) {  
    echo "Boa tarde!";  
} else {  
    echo "Boa noite!";  
}  
  
?>
```

6.4 - Comando Switch

A instrução switch é utilizada para selecionar apenas um dos blocos de código a serem executados, dentre todos os que você precisa definir.

```
<?php  
  
switch ($opcoes) {  
  
    case 1:  
  
        echo "Você escolheu a primeira opção";  
  
        break;
```

case 2:

echo "Você escolheu a segunda opção";

break;

case 3:

echo "Você escolheu a terceira opção";

break;

default:

echo "Essa opção não existe";

}

?>

Ela funciona da seguinte forma: primeiro, temos uma expressão (geralmente uma variável), no nosso caso a variável **\$opcoes**, que é avaliada uma vez. O valor que a variável **\$opcoes** recebe é, então, comparado com os valores para cada caso (case) na estrutura do código. Se esse valor for correspondente a algum valor definido em caso (case), o bloco de código associado a esse caso (case) vai ser executado.

O **Break** é utilizado para evitar que o código seja executado no próximo caso, faz com que apenas o caso que foi escolhido seja executado e nada mais que venham depois dele. Mais a frente, falaremos sobre a instrução **break** (pausa).

A instrução **default** é simples. Ela está ali para ser executada, caso as opções que pré-determinamos, ou seja, 1, 2 ou 3 não sejam escolhidas pelo usuário. Então, se o usuário digitar: 4, 5, 10 ou uma letra (ou qualquer coisa que não seja 1,2 ou 3) é a opção **default** que será executada.

7.1 - Comando While (Enquanto)

Quando você precisa que o mesmo bloco de código seja executado várias vezes, você pode utilizar estruturas de repetição (Loops) ao invés de adicionar várias linhas de código com a mesma instrução.

As estruturas de repetições (Loops) são usados para executar o mesmo bloco de código repetidas vezes, desde que uma condição determinada por você seja atendida.

No PHP, assim como em outras linguagens de programação, existem alguns tipos de estruturas de repetição, sendo que cada uma é utilizada para um determinado objetivo. Nos próximos tópicos, vamos tratar de cada uma dessas estruturas.

Observação: A palavra **loop** vem do inglês e significa ciclo. Justamente porque a estrutura de repetição fica em ciclo, até que a condição estabelecida não seja mais atendida e o ciclo é finalizado.

While (Enquanto)

A Estrutura de repetição (Loop) While (Enquanto) vai executar um bloco de código, enquanto uma condição pré estabelecida estiver sendo atendida.

Por exemplo: se você quer que todos os números de **0 a 9** sejam impressos na tela, enquanto o valor de determinada variável for menor que **10**.

```
<?php
$a = 0;

while($a < 10) {
    echo "$a <br>";
    $a++;
}
?>
```

7.2 - Comando Do, While (Faça, Enquanto)

O Comando Do... **While** (Faça ... Enquanto), faz um loop em um bloco de código uma vez antes de passar pelo While e, em seguida, repete o loop enquanto condição especificada por você para parar o Loop não for atendida.

A grande diferença de utilizar Do... **While**, ao invés de utilizar apenas **While**, é que o Loop será executado ao menos uma vez. No caso da utilização apenas do comando **While** a condição de parada pode ser atendida antes de entrar no Loop e ele não ser executado nem ao menos uma vez.

```
<?php
$a = 1;

do {
    echo "$a <br>";
    $a++;
} while ($a < 10);
?>
```

7.3 - Comando For

O Comando **For** (Para) é usado quando você sabe, com antecedência, quantas vezes você quer repetir determinado código.

```
<?php
for ($a = 0; $a <= 10; $a++) {
    echo "$a <br>";
}
?>
```

7.4 - Comando Foreach

O comando **Foreach** (Para cada) funciona apenas em **Arrays** e é usado para percorrer cada par de índice e valor em um **Array**.

Lembra quando falamos sobre **Arrays** e mostramos que um **Array** possui valores e que esses valores possuem um índice? Pois bem, agora nós conseguimos percorrer esses **Arrays** utilizando o comando **Foreach**.

```
<?php
$cores = array("vermelho", "verde", "branco");
foreach ($cores as $valor) {
    echo "$valor <br>";
}
?>
```

7.5 - Break e Continue

Break(Pausa)

Como havíamos dito no tópico sobre a instrução **Switch**, o comando **Break** serve para interromper um bloco de código e ignorar os outros blocos de código e também serve para sair de um **Loop**, veja a seguir.

Este exemplo sai do loop quando a variável **\$a** for igual a **8**:

```
<?php
for ($a = 0; $a < 10; $a++) {
    if ($a == 8) {
        break;
    }
    echo "$a <br>";
}
?>
```

Continue (Continuar)

O comando **Continue (Continuar)** interrompe uma iteração (no loop), se uma condição especificada for atendida e continua com a próxima iteração no loop.

Este exemplo pula o valor de **8**:

```
<?php
for ($a = 0; $a < 10; $a++) {
    if ($a == 8) {
        continue;
    }
    echo "$a <br>";
}
?>
```

8.1 - Criando Funções em PHP

Segundo o [W3Schools](https://www.w3schools.com/php/php_functions.asp) o verdadeiro poder do PHP vem das suas funções.

O PHP possui várias funções prontas que você pode utilizar em seus projetos e, também, conta com a opção de suas funções conforme suas necessidades.

Nesse link https://www.w3schools.com/php/php_functions.asp você encontra várias funções do PHP. Sugerimos que você guarde o link para no futuro, ele será útil para quando estiver construindo projetos.

Funções definidas por você no PHP

Funções são blocos de instruções que podem ser utilizados várias vezes em seu código. Elas são executadas quando você chama por elas, essa tarefa é chamada de: chamada para a função.

Criando uma Função:

Quando vamos criar uma função normalmente chamamos de “declarar uma função”. Esses termos são muito comuns em programação, é legal que você se habitue a essa forma de falar na programação.

Então para declarar uma função, nós precisamos utilizar a palavra `function`.

Algumas informações importantes: o nome de uma função deve começar com uma letra ou sublinhado, nunca utilize números para iniciar o nome de uma função. Sempre utilize nomes que tem q ver com o que aquela função faz, chamamos isso de: boa prática de programação. Os nomes das funções NÃO diferenciam maiúsculas de minúsculas.

Exemplo: No código abaixo estamos criando uma função chamada escreverNome (). Os sinais de chaves { } demonstram o escopo da função, ou seja, o início e o fim da função. A função do exemplo abaixo exibe a mensagem Meu nome é: Lucas. Para chamar a função, basta escrever seu nome seguido de parênteses ():

```
<?php  
  
function escreverNome() {  
  
    echo "Meu nome é: Lucas";  
  
}  
  
escreverNome(); //chamando a função  
  
?>
```

As Funções podem ser chamadas, podemos também dizer (invocadas), de qualquer ponto do seu código para executar uma tarefa que foi definida. Você pode chamar uma função quantas vezes for necessário. Quando uma função é chamada ela faz com que o código PHP que a chama pare, e fique esperando que o código que está dentro da função (escopo do função) seja executado. Depois que o código da função é executado, ele retorna para dar continuidade ao restante do seu código.

Por que usar funções?

As funções permitem o reaproveitamento do código já construído por você. Elas fazem com que um trecho de código não seja repetido várias vezes, dentro do mesmo programa que você está construindo. Com elas você consegue alterar um trecho de código de uma forma mais rápida e consegue separar o programa em blocos que podem ser compreendidos de forma isolada, tudo isso ajuda você na hora de criar e dar manutenção no seu código. Isso é chamado de boas práticas de programação.

Argumentos em Funções do PHP

Você pode passar informações para as funções se precisar. Essa passagem de informações para as funções são chamadas de Argumentos. Um argumento é como uma variável, ou seja, ele tem um valor e passa esse valor para a função.

Para passar os argumentos para as funções devemos colocá-los dentro dos parênteses da função. Você pode passar quantos argumentos achar necessário para uma função. É só colocar uma vírgula após cada argumento que você colocar dentro da função.

Utilizando a mesma função, que fizemos acima, de exemplo, mas passando o meu nome como argumento:

```
<?php
```

```
$nome = "Lucas";
```

```
function escreverNome($nome) {
```



```
echo "Meu nome é: $nome";  
}
```

```
escreverNome($nome); //chamando a função  
?>
```

Retornando valores nas funções

As funções, também, podem retornar valores ao invés de apenas imprimir uma informação na tela. Para isso, devemos utilizar a instrução `return`.

O código abaixo faz uma função que retorna a soma de dois números:

```
<?php
```

```
function soma($a, $b) {  
    $resultado = $a + $b;  
    return $resultado;  
}
```

```
echo "O resultado da soma de 10 + 10 é: " . soma(10,10);  
?>
```

O exemplo acima é um exemplo básico do funcionamento de uma função, que retorna um valor, mas é claro que podemos fazer coisas mais complexas e sofisticadas utilizando funções. O sinal de . é uma forma de concatenar (juntar) na instrução echo a string (frase). O resultado da soma de 10 + 10 é: com a chamada da função. Sendo assim, na linha em que o echo está escrito, já fazemos a chamada da função passando para ela os argumentos que deve utilizar para realizar a soma dentro do seu escopo em:

```
function soma($a, $b) {  
  
    $resultado = $a + $b;  
  
}
```

e voltar já com o valor de retorno sendo impresso e dando o resultado esperado da função que será na tela:

O resultado da soma de 10 + 10 é: 20.

Existem muitas coisas para serem ditas sobre funções, mas não trataremos sobre isso agora. Se você estiver seguro do seu aprendizado e quiser saber um pouco mais sobre funções você pode acessar: https://www.w3schools.com/php/php_functions.asp.

9.1 - Manipulando Formulários em PHP (Método GET)

Lá atrás, quando falamos que era desejado que você tivesse conhecimentos em HTML para fazer nosso curso era sobre os momentos em que precisamos juntar o HTML com o PHP para explicar algumas coisas.

Pois bem, esse momento chegou. Se você ainda tem dificuldades para entender o que são formulários em HTML pedimos que você acesse esse link: https://www.w3school.com/html/html_forms.asp e dê uma lida sobre o assunto, você vai ver que é simples.

Um formulário simples, em HTML, no qual o usuário de seu site vai inserir os dados como Nome e E-mail.

```
<html>
<body>

  <form action="dados.php" method="get">
    Name: <input type="text" name="nome"><br>
    E-mail: <input type="text" name="email"><br>
    <input type="submit">
  </form>

</body>
</html>
```

As informações enviadas de um formulário, via método **GET**, são visíveis para todos. Pois as informações serão enviadas pela URL (o que é uma url? calma, no vídeo abaixo falamos sobre isso).

O método **GET** também tem as limitações de quantidade de informações que você pode enviar. A limitação é algo em torno de 2.000 caracteres, ou seja, 2.000 teclas digitadas no teclado por quem está utilizando o formulário. Você pode se perguntar: mas quem irá escrever cerca de 2.000 caracteres em um formulário? Bom, os métodos de envio não são utilizados

apenas para formulários, eles podem ser utilizados para várias outras coisas. Assista ao vídeo abaixo em que falamos com mais detalhes sobre o método GET.

No entanto, nesse momento, lembre-se disso: **nunca utilize o método GET para enviar dados confidenciais.**

9.2 - Manipulando Formulários em PHP (Método POST)

Abaixo, temos um formulário simples, em HTML, para ilustrar um pouco nossa conversa:

```
<html>
<body>

  <form action="entrar.php" method="post">
    Usuário: <input type="text" name="usuario"><br>
    Senha: <input type="text" name="senha"><br>
    <input type="submit">
  </form>

</body>
</html>
```

Diferente no método **GET**, no método **POST** os dados que serão enviados pelo formulário não ficam visíveis na **URL** e são encapsulados e enviados, de forma que ninguém (nem o usuário) tenha acesso aos dados, a não ser o servidor que irá receber esses dados. O método **POST** não possui limitação na quantidade de dados que pode enviar para o servidor.

10.1 - Include

A instrução **include** é muito útil quando precisamos inserir em um arquivo (código), que estamos construindo, o código de outro arquivo que já fizemos e precisa ser utilizado, também, no código que estamos escrevendo. Em outras palavras, podemos inserir o mesmo PHP, HTML ou

texto, de outro arquivo, no arquivo que estamos construindo sem precisar escrever o código novamente.

Fiquem tranquilos, vocês vão ver nos próximos tópicos como vai ser interessante e, até, necessário utilizar instruções de inclusão de arquivos em nossos projetos de PHP.

Exemplo:

Suponha que temos um arquivo de rodapé padrão, chamado "rodape.php", que contenha o código:

```
<?php  
echo "<p>Desenvolvido por: Lucas Tavares 2021</p>";  
?>
```

E, por padrão, definimos que todas as nossas páginas irão conter o mesmo rodapé. Podemos utilizar a instrução include e incluir em cada uma das páginas o rodapé, ao invés de copiar essa parte de código em cada arquivo PHP que criamos.

```
<html>  
<body>  
  
<h1>Bem Vindo!</h1>  
<p>Página do Prof. Lucas Tavares.</p>  
  
<?php include 'rodape.php';?>  
  
</body>  
</html>
```

Esse é o exemplo de utilização simples do conceito apenas para aprendizado, Porém a inclusão de arquivos em PHP pode ser utilizada para coisas mais complexas, o que pode ajudar bastante na criação de códigos.

10.2 - Require

A instrução **require** faz a mesma coisa que a função **include**, ou seja, inclui outros arquivos dentro um arquivo. Existe apenas uma característica da instrução **require** que a difere da instrução **include**.

Essa característica está relacionada em como as duas instruções lidam com as falhas:

Se, porventura, ocorrer alguma falha no carregamento do arquivo **incluído**, ou em seu processamento, o **require** irá reproduzir um erro fatal e parar o código.

Diferente da instrução **include** que vai apenas reproduzir um aviso (**Warning**) (palavra utilizada pelo PHP para exibir avisos) e continuará com o script.

Portanto, se você quiser que a execução continue, mesmo se o arquivo de **inclusão** estiver ausente, use a instrução **include**.

Caso contrário, no caso de uma codificação de PHP mais complexa em que essa parte do código seja imprescindível para que as coisas funcionem da forma correta, sempre use a instrução **require**. Isso ajuda a evitar o comprometimento da segurança e integridade do seu código.

Em resumo, a tradução da palavra **require** é: exigir e da palavra **include** é: incluir.

Como utilizar a instrução require?

Para utilizar a instrução **require** basta substituir a palavra **include** por **require**, o exemplo utilizado no tópico **include** pode ser utilizado. Repare que no código abaixo a única diferença está na palavra **require** substituindo a palavra **include**.

```
<?php require "rodape.php";?>
```

Incluir arquivos economiza trabalho. Você pode criar um cabeçalho, rodapé ou arquivo de menu padrão para todas as suas páginas e quando precisar atualizar alguma dessas partes você irá precisar atualizar o arquivo de **inclusão**. Assim, todos os arquivos do seu site que contém esse arquivo incluído serão atualizados.

11.1 - Cookies

Já reparou quando você acessa uma página da internet e ela pergunta a você se você deseja aceitar Cookies? Pois bem. Os Cookies são pequenos arquivos que o servidor onde está hospedado aquele site incorpora no computador de quem está acessando o site para salvar algumas coisas importantes para o funcionamento do site.

Os Cookies são definidos por você na hora de construir seu site e escrever o código. Um site pode conter diversos Cookies para as mais variadas situações. Tais, como: Armazenar preferências de idioma, seu e-mail, senhas usadas para acessar esse site, sua localização, etc. De modo que todas as vezes que você voltar aquele site essas informações vão ser buscadas e você não precisará colocar o email e a senha novamente para fazer login em algum site por exemplo.

É claro que, além do lado bom, os Cookies também possuem algumas desvantagens como a segurança desses dados. Uma vez que quem tem um pouco de conhecimento em informática, com acesso ao computador, consegue acessá-los.

Assista ao vídeo no final desse tópico onde falo um pouco mais sobre esses detalhes.

Criar cookies com PHP

Para criar um Cookie podemos utilizar a função `setcookie()`;

```
<?php  
  
setcookie();  
  
?>
```

O exemplo abaixo cria um cookie chamado "usuário" com o valor "Lucas Tavares". O cookie irá expirar após 30 dias ($86400 * 30$). A barra / significa que o cookie está disponível em todo o site. Caso contrário, você pode selecionar apenas as pastas do seu site onde este Cookie pode ser usado. A função `setcookie()` deve estar sempre antes da tag `<html>`.

Em seguida, recuperamos o valor do cookie "usuário" utilizando a variável global `$_COOKIE`. Dizer que uma variável é global é dizer que ela faz parte de todos os escopos do código, você pode utilizar essa variável `$_COOKIE` que já é definida pelo PHP para recuperar Cookies, para recuperar o valor do Cookie que você criou, em qualquer parte do seu código. Diferente de uma variável que tem seu escopo local, em funções, por exemplo, as variáveis que estão dentro da função só podem ser visualizadas e utilizadas dentro da função. Pois, possuem seu escopo apenas dentro da função e não global.

```
<?php  
  
$nomeDoCookie = "usuario";  
  
$valorDoCookie = "Lucas Tavares";  
  
setcookie($nomeDoCookie, $valorDoCookie, time() + (86400 * 30), "/");  
  
?>  
  
<html>  
  
<body>  
  
<?php  
  
if (!isset($_COOKIE[$nomeDoCookie])) {  
  
    echo "O Cookie " . $nomeDoCookie . " não existe.";  
  
} else {
```



```

        echo "Cookie: " . $nomeDoCookie . "<br>";

        echo "Valor: " . $valorDoCookie;

    }

?>

</body>

</html>

```

Para excluir um cookie basta utilizar a função **setcookie()** com uma data de validade no passado. Quando colocamos o **time()** tempo como -3600 o navegador entende que aquele Cookie está vencido e, por isso, precisa ser deletado.

```

<?php
    setcookie("usuário", "", time() - 3600);
?>
<html>
<body>

    <?php
        echo "O Cookie foi deletado.";
    ?>

</body>
</html>

```

11.2 - Sessions (Sessões)

As Sessões ao contrario dos Cookies não são armazenadas no computador do Usuário. Mas é uma forma também de você armazenar informações do usuário para utilizar em várias paginas do seu site.

Quando você acessa o site do seu banco por exemplo, a cada interação que você tem com o site, ele tem que saber que é você que está ali sem salvar nada no seu computador por questões obvias de segurança. Assim toda vez

que você acessa o site do seu banco ele tem que saber que é você que está ali do começo ao fim da conexão (utilização do site). As sessões são uma forma fazer isso.

A vantagem das informações salvas na Sessão para as informações salvas em Cookies é que elas só ficam salvas enquanto você está navegando pela página. Se você fechar o navegador, por exemplo, todas as informações de senha ou outras coisas que você colocou vão ser perdidas. Assim, as variáveis de Sessão contêm informações sobre um único usuário e estão disponíveis para todas as páginas em um web site apenas no momento de sua utilização.

Se você acessar o internet ranking de qualquer banco pelo computador vai conseguir notar esse caso que foi descrito acima. No momento que você fecha o navegador deve ter que fazer todo o acesso novamente para conseguir entrar em sua conta bancária.

Iniciando uma Sessão(Session)

Uma sessão é iniciada com a função **session_start()**. E as variáveis de sessão serão definidas com a variável global do PHP: `$_SESSION`.

A função **session_start()** deve ser a primeira coisa em seu documento. Antes de qualquer tag HTML.

No exemplo abaixo eu construo um código(arquivo chamado: index.php) para criar uma sessão e definir quais valores quero que ficam salvos enquanto a seção estiver funcionando:

```
<?php
//Iniciando a sessão
session_start();
?>
<!DOCTYPE html>
<html>
```

```
<body>

<?php

    $_SESSION["nome"] = "Lucas";
    $_SESSION["sobrenome"] = "Tavares";
    echo "As variáveis da sessão estão definidas.";
?>

</body>
</html>
```

Obter valores de variáveis de sessão PHP

Agora vou criar um outro código(arquivo chamado: pagina2.php) que vou colocar em outra página para acessar essas informações salvas nesse arquivo anterior.

Observe que as variáveis de sessão não são passadas individualmente para cada nova página, em vez disso, são recuperadas da sessão que abrimos no início de cada página **session_start()**.

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
    <body>

        <?php
            // Recuperando as variáveis de outro arquivo
            echo "O nome do usuário da sessão é: " . $_SESSION["nome"]
            . "<br>";
            echo "O Sobrenome dele é: " . $_SESSION["sobrenome"] . ".";
        ?>

    </body>
</html>
```

Como o navegador sabe quem eu sou?

As sessões definem uma chave de usuário no computador de quem está acessando a página parecida com: 765487cf34ert8dede5a562e4f3a7e12. Quando uma sessão é aberta em outra página, ela busca uma chave de usuário. Se a chave for a mesma, ela acessa aquela sessão, caso seja diferente, ela inicia uma nova sessão.

Destrua uma sessão PHP

As vezes é necessário destruir uma sessão, ou seja, encerra-lá. O internet banking é um bom exemplo. Geralmente os sites de banco tem um botão de sair para que você encerre sua conexão com o banco.

Para destruir uma sessão devemos utilizar **session_unset()** e **session_destroy()**:

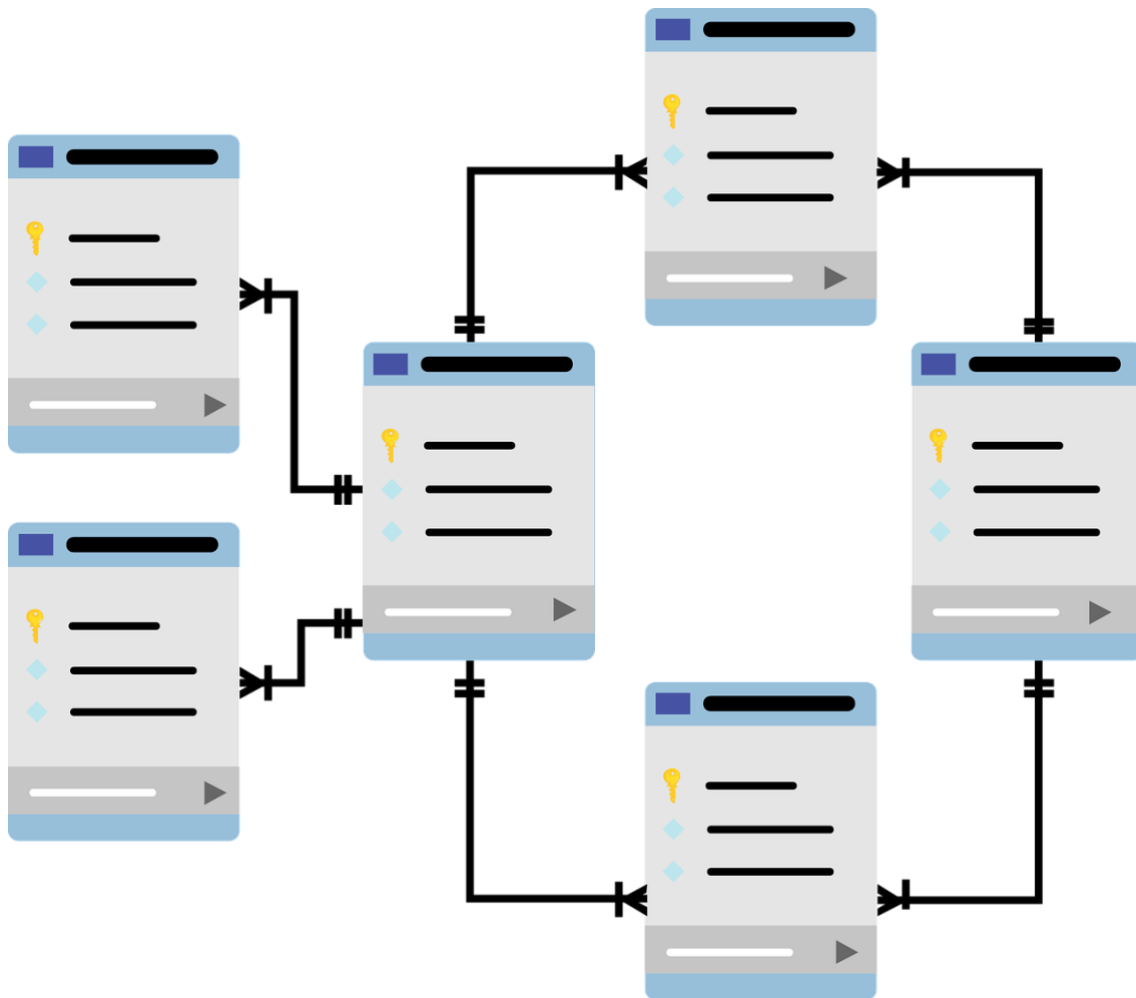
```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
    <body>

        <?php
            //Remove os valores das variáveis
            session_unset();

            //Destrói a sessão
            session_destroy();
        ?>

    </body>
</html>
```

12.1 - O que é MySQL? (Pequena Revisão)



Com o PHP é possível conectar e manipular um banco de dados. Essa tarefa não só é útil, mas é essencial para quem quer criar um Sistema web completo (**Aplicativo Web**).

Segundo o w3schools, o **MySQL** é o sistema de banco de dados mais popular utilizado com o PHP.

É importante lembrar que existem diversos bancos de dados que podemos utilizar, mas em nosso curso vamos adotar apenas o **MySQL**, para não gerar um acúmulo de informações nesse momento. Isso pode causar um pouco de confusão na cabeça de quem ainda está aprendendo a programar, mas deixamos aqui o nosso apoio para que no decorrer de sua

carreira, como programador, sempre procure e aprenda sobre diferentes tecnologias.

O que é MySQL?

O **MySQL** é um sistema de banco de dados usado na web para ser **MySQL** executado em um servidor. Pode ser utilizado tanto em sistemas pequenos e em sistemas grandes e robustos. Ele é eficiente, pois é rápido, confiável, sua utilização não é complicada e você pode utilizá-lo gratuitamente.

Os dados no banco **MySQL** são armazenados em tabelas, uma tabela é um conjunto de dados relacionados que possuem linhas e colunas.

Como usamos o Awardspace ou o Xampp, como exemplo, iremos utilizar o PhpMyAdmin. Essa é uma ferramenta disponível em ambas as plataformas para trabalharmos com banco de dados.

Se você quiser estudar um pouco mais apenas sobre a linguagem **SQL**, acesse: <https://www.w3schools.com/sql/default.asp>

Nesse link, existe um tutorial que trata de vários assuntos relacionados a linguagem **SQL**, mas, nos próximos tópicos, iremos tratar dos assuntos em banco de dados que são necessários para o andamento de nosso curso.

Site do oficial do **MYSQL**: <http://www.mysql.com>

Empresas que utilizam

o **MYSQL**: http://www.mysql.com/custo_mers/

Lembre-se: **SQL** é uma linguagem para manipulação de banco de dados e **MySQL** é um sistema para gerenciar um banco de dados.

Sobre as abordagens de Bancos de dados em PHP

Na linguagem **PHP**, existem algumas formas de trabalhar com Banco de dados que foram se modificando no decorrer dos anos.

Iremos trabalhar com **MySQLi** (Orientado a objetos) e no decorrer dos outros tópicos você vai entender do que estamos falando.

Para não tornar o tema de banco de dados confuso, iremos trabalhar apenas com essa abordagem. No futuro, você pode estudar essas novas abordagens conforme seu conhecimento for mais sólido em programação.

Para saber mais sobre essas outras abordagens, clique em: https://www.w3schools.com/php/php_mysql_connect.asp

12.2 - Criação do Banco de dados em PHP

Para criar um Banco de dados devemos utilizar a instrução **CREATE DATABASE**.

O código abaixo cria um Banco de dados chamado **aulaBd**.

```
<?php
$servidor = "localhost";
$usuario = "usuario";
$senha = "12346";

//Criando a conexão
$conexao = new mysqli($servidor, $usuario, $senha);
//Testando a conexão
if ($conexao-> connect_error) {
    die("Conexão falhou: " . $conexao->connect_error);
}
```

```
}
```

```
//Criando o Banco de dados  
$sql = "CREATE DATABASE aulaBd";  
if ($conexao->query($sql) === TRUE) {  
    echo "Banco de dados Criado com sucesso";  
} else {  
    echo "Erro ao criar o Banco de dados: " . $conexao->error;  
}  
  
$conexao->close();  
?>
```

12.3 - Conexão do Banco de dados em PHP

O código abaixo é um exemplo de como se conectar ao banco de dados:

```
<?php  
$servidor = "localhost";  
$usuario = "usuario";  
$senha = "123456";  
  
//Criando a conexão  
$conexao = new mysqli($servidor, $usuario, $senha);  
  
//Testando a conexão  
if ($conexao->connect_error) {  
    die("A conexão falhou: " . $conexao->connect_error);  
}  
echo "Conectado com Sucesso";  
?>
```

12.4 - Criação de Tabela no Banco de dados em PHP

As tabelas em um banco de dados tem seu nome exclusivo e possuem linhas e colunas.

Para criar as tabelas devemos utilizar a instrução CREATE TABLE.

```
<?php
```

```
$servidor = "localhost";  
$usuario = "usuario";  
$senha = "123456";  
$nomeBD = "aulaDb";  
  
//Criando a conexão  
$conexao = new mysqli($servidor, $usuario, $senha, $nomeBD);  
//Testando a conexão  
if ($conexao->connect_error) {  
    die("Erro na conexão: " . $conexao->connect_error);  
}  
  
//Criando a tabela  
$sql = "CREATE TABLE usuarios (  
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
nome VARCHAR(30) NOT NULL,  
sobrenome VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
)";  
  
if ($conexao->query($sql) === TRUE) {  
    echo "Tabela Usuários criada com sucesso";  
} else {  
    echo "Erro ao criar a tabela: " . $conexao->error;  
}  
  
$conexao->close();  
?>
```

12.5 - Inserindo dados no Banco de dados

Depois de criar o banco de dados e uma tabela, podemos começar a adicionar dados neles.

Algumas regras de sintaxe devem ser seguidas para a inserção de dados funcionar. São elas:

- Os dados da inserção(consulta SQL) devem estar entre aspas

-Os valores numéricos não devem estar entre aspas

-A palavra NULL não deve ser citada

Para inserir dados em uma tabela podemos utilizar a instrução INSERT INTO:

```
<?php
```

```
$servidor = "localhost";
```

```
$usuario = "usuario";
```

```
$senha = "123456";
```

```
$nomeBD = "aulaDb";
```

```
//Criando a conexão
```

```
$conexao = new mysqli($servidor, $usuario, $senha, $nomeBD);
```

```
//Testando a conexão
```

```
if ($conexao->connect_error) {
```

```
    die("Erro na conexão: " . $conexao->connect_error);
```

```
}
```

```
$sql = "INSERT INTO usuarios (nome, sobrenome, email)
```

```
VALUES ('Lucas', 'Tavares', 'lucas.d.silva@pbh.gov.br')";
```

```
if ($conexao->query($sql) === TRUE) {
```

```
    echo "Dados gravados com sucesso.";
```

```
} else {
```

```
    echo "Error: " . $sql . "<br>" . $conexao->error;
```

```
}
```

```
$conexao->close();
```

```
?>
```

12.5.1 - Selecionando os dados inseridos no Banco de dados

Quando queremos selecionar dados, de um banco de dados, devemos utilizar a instrução SELECT.

```
<?php
```

```

$servidor = "localhost";
$usuario = "usuario";
$senha = "123456";
$nomeBD = "aulaDb";

//Criando a conexão
$conexao = new mysqli($servidor, $usuario, $senha, $nomeBD);
//Testando a conexão
if ($conexao->connect_error) {
    die("Erro na conexão: " . $conexao->connect_error);
}

$sql = "SELECT id, nome, sobrenome FROM usuarios";
$resultado = $conexao->query($sql);

if ($resultado->num_rows > 0) {
    //Saída de cada linha
    while($linha = $resultado->fetch_assoc()) {
        echo "id: " . $linha["id"]. " - Nome: " . $linha["nome"]. " " .
$linha["sobrenome"]. "<br>";
    }
} else {
    echo "Sem registros";
}
$conexao->close();
?>

```

Explicando o código acima:

Primeiro, configuramos uma consulta **SQL** que seleciona as colunas id, nome e sobrenome da tabela usuários. A próxima linha executa a consulta e coloca os dados resultantes na variável **\$resultado**.

Em seguida, utilizamos a função: **num_rows()**, para verificar se a seleção retornou algum resultado. A função **num_rows()**, nada mais é que um contador de linhas. **Num** vem de número e **rows** traduzido para o português significa linhas, ou seja, número de linhas. Assim, a função **num_rows()** vai retornar um valor. Se maior que zero esse valor, significa que a seleção encontrou um ou mais registros.

Se houver um ou mais registros podemos utilizar a função **fetch_assoc()** para colocar todos os resultados em uma matriz

associativa que podemos percorrer utilizando o loop **while()**. Fazemos essa matriz associativa, ou seja, associamos a cada registro do banco de dados a coluna a qual ele faz parte e utilizamos o **loop while** para que possamos imprimir essa tabela de dados selecionados no **PHP**. Tudo isso deve ser feito porque o PHP não entende diretamente os dados que vem do Banco de dados, por isso, tudo que recuperamos das tabelas do Banco de dados deve ser tratado para ser utilizado da forma correta em nosso código.

12.5.2 Selecionando e filtrando os dados inseridos no Banco de dados

Podemos utilizar a cláusula **WHERE** para filtrar os dados e extrair apenas os dados que atendem a uma condição específica que definimos.

<?php

```
$servidor = "localhost";
```

```
$usuario = "usuario";
```

```
$senha = "123456";
```

```
$nomeBD = "aulaDb";
```

```
//Criando a conexão
```

```
$conexao = new mysqli($servidor, $usuario, $senha, $nomeBD);
```

```
//Testando a conexão
```

```
if ($conexao->connect_error) {
```

```
    die("Erro na conexão: " . $conexao->connect_error);
```

```
}
```

```
$sql = "SELECT id, nome, sobrenome FROM usuarios WHERE
```

```
sobrenome='Lucas';
```

```
$resultado = $conexao->query($sql);
```

```
if ($resultado->num_rows > 0) {
```

```
    //Saída de cada linha
```

```
    while($linha = $resultado->fetch_assoc()) {
```

```
        echo "id: " . $linha["id"]. " - Nome: " . $linha["nome"]. " " .
```

```
$linha["sobrenome"]. "<br>";
```

```
    }
```

```
} else {
```

```
    echo "0 results";
```

```
}  
$conexao->close();  
?>
```

12.5.3 - Excluindo os dados inseridos na tabela do Banco de dados

Podemos utilizar a instrução **DELETE** para excluir dados de uma tabela:

```
<?php  
  
$servidor = "localhost";  
$usuario = "usuario";  
$senha = "123456";  
$nomeBD = "aulaDb";  
  
//Criando a conexão  
$conexao = new mysqli($servidor, $usuario, $senha, $nomeBD);  
//Testando a conexão  
if ($conexao->connect_error) {  
    die("Erro na conexão: " . $conexao->connect_error);  
}  
  
//Consulta para Deletar os dados  
$sql = "DELETE FROM usuarios WHERE id=5";  
  
if ($conexao->query($sql) === TRUE) {  
    echo "Dados deletados com sucesso";  
} else {  
    echo "Erro ao deletar dados: " . $conexao->error;  
}  
  
$conexao->close();  
?>
```

12.5.4 - Atualizando os dados inseridos na tabela do Banco de dados

Para atualizar dados já existentes em um tabela podemos utilizar a instrução UPDATE.

<?php

\$servidor = "localhost";

\$usuario = "usuario";

\$senha = "123456";

\$nomeBD = "aulaDb";

//Criando a conexão

\$conexao = new mysqli(\$servidor, \$usuario, \$senha, \$nomeBD);

//Testando a conexão

if (\$conexao->connect_error) {

die("Erro na conexão: " . \$conexao->connect_error);

}

\$sql = "UPDATE usuarios SET sobrenome='Silva' WHERE id=3";

if (\$conexao->query(\$sql) === TRUE) {

echo "Dados atualizados com sucesso";

} else {

echo "Erro ao atualizar os dados: " . \$conexao->error;

}

\$conexao->close();

?>