

**CURSOS
TÉCNICOS**

**DESENVOLVIMENTO DE
SISTEMAS WEB II**

Eixo Informática para Internet

UNIDADE 1

SUMÁRIO

UNIDADE 1	4
1. Introdução ao PHP – conceitos, comandos básicos e instalação	4
1.1 Introdução.....	4
1.2 Conceitos importantes do PHP.....	5
1.3 O que é possível desenvolver com PHP?	5
1.4 Padrões de renderização na WEB	6
2. Primeiros Passos com o PHP	7
3. Escolhendo o ambiente de desenvolvimento	7
3.1 Visual Studio Code.....	7
3.2 PHP Storm	8
3.3 Escrevendo o primeiro código PHP	8
3.4 Conhecendo a sintaxe do PHP.....	9
3.5 Estruturas de controle.....	12
4. Referências.....	16

APRESENTAÇÃO DA DISCIPLINA

Boas-vindas à disciplina Desenvolvimento de Sistemas Web II

Nesta disciplina, exploraremos as bases tecnológicas essenciais para o desenvolvimento de sistemas web, com ênfase na linguagem PHP. Então, você conhecerá os conceitos fundamentais, como algoritmos em PHP, programação orientada a objetos, a estrutura organizacional da Internet e protocolos de navegação. Desenvolveremos habilidades sólidas em HTML e CSS para criar páginas web bem estruturadas e estilizadas, com um design atraente. Além disso, aprenderemos a utilizar template HTML e ferramentas que facilitam o desenvolvimento web, tornando o processo mais eficiente. Vamos discutir aspectos práticos, como a administração de sites, abordando tópicos como registro de domínios, serviços de hospedagem e procedimentos de publicação.

Bons estudos!

Núcleo de Ensino Técnico e Profissionalizante – NETeP.

UNIDADE 1

1. Introdução ao PHP – conceitos, comandos básicos e instalação

Neste tópico, o foco recai sobre elementos introdutórios do PHP, tais como conceito, comando básicos e instalação. Vejamos:

1.1 Introdução

Segundo a documentação oficial do PHP, disponível em português, O PHP (um acrônimo recursivo para **PHP: Hypertext Preprocessor**) é uma linguagem de *script open source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML.

Sim! É possível adicionar códigos da sintaxe PHP, delimitado por **instruções de processamento** ou tags de início `<?php` e fim `?>`. Confira o exemplo abaixo que insere o texto “Olá, eu sou um script PHP, que printa algo em uma página WEB”, dentro do body usando o comando `echo` do PHP:

Exemplo de código PHP embutido no HTML:

```

1  <!DOCTYPE html>
2  <html lang="pt-br">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Exemplo - PHP</title>
8  </head>
9
10 <body>
11   <?php
12     echo "Olá, eu sou um script PHP, que printa algo em uma página WEB"
13   ?>
14 </body>
15
16 </html>
  
```

O que distingue o PHP de algo como o **JavaScript**, sob a perspectiva **do cliente (client-side)** é que o código é executado no servidor, gerando o HTML que é então enviado para o navegador.

O principal ponto em optar por utilizar PHP é que ele é extremamente simples para um iniciante, mas oferece muitos recursos avançados para um programador profissional.

1.2 Conceitos importantes do PHP

Existem alguns conceitos que envolvem o PHP, são eles:

- **Linguagem interpretada:** O PHP é uma linguagem interpretada, que significa que o código-fonte PHP não é compilado em código de máquina diretamente. Em vez disso, o código PHP é lido e executado, linha por linha por um interpretador PHP, que converte o código em instruções executáveis em tempo de execução. Isso permite uma flexibilidade significativa, pois os desenvolvedores podem fazer alterações no código PHP e ver os resultados imediatamente, sem a necessidade de recompilar.
- **Multi paradigma:** O PHP é uma linguagem de programação que suporta múltiplos paradigmas de programação. Isso significa que você pode escrever código PHP de várias maneiras, dependendo dos requisitos do seu projeto e de sua preferência pessoal. Os principais paradigmas suportados pelo PHP incluem o **procedural**, **orientado a objetos**, **funcional** e **orientado a eventos**.
- **Tipagem dinâmica + manual:** O PHP é uma linguagem de tipagem dinâmica, o que significa que as variáveis não têm um tipo de dados fixos associados a elas em tempo de compilação. O tipo de uma variável é determinado durante a execução do programa, com base no valor que ela armazena.

Além disso, o PHP permite que os desenvolvedores definam **tipos de dados** manualmente usando declarações de tipo (introduzidas em versões mais recentes, como PHP 7 e superior). Isso é chamado de "tipagem manual". Por exemplo, você pode declarar uma função com tipos de parâmetros específicos, como int ou string.

Esses conceitos tornam o PHP **uma linguagem flexível** que **pode ser adaptada** para uma variedade de situações de desenvolvimento, permitindo que os desenvolvedores escolham o paradigma de programação, e o nível de tipagem que melhor atendam às necessidades de seus projetos.

1.3 O que é possível desenvolver com PHP?

Basicamente **qualquer coisa**. O PHP é focado, principalmente, em *scripts* que operam no **lado do servidor (server-side)**, portanto é possível executar qualquer tarefa que um programa do tipo **CGI (Common Gateway Interface)** faz, como coletar dados de um

DESENVOLVIMENTO DE SISTEMAS WEB II

formulário da WEB, construir páginas com conteúdo dinâmico ou até mesmo, enviar e receber **cookies**.

Em resumo temos três principais áreas onde *scripts PHP* são utilizados:

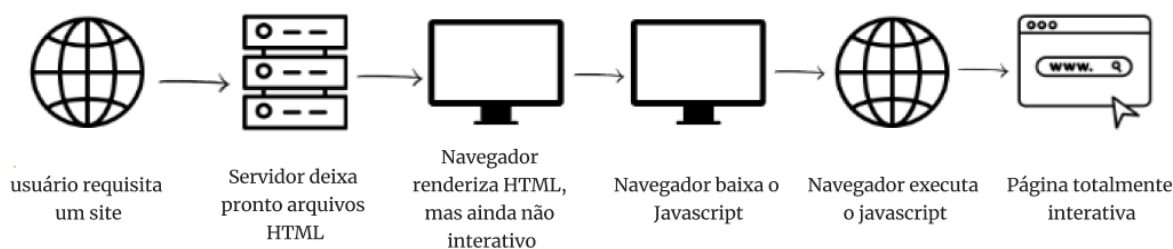
- Script no lado do servidor (*server-side*);
- Scripts na linha de comando (CLI), seja win, macOS ou Linux;
- Para desenvolver aplicações desktop.

1.4 Padrões de renderização na WEB

Vamos focar nos padrões de renderização? Pois bem, entre eles estão o *server side rendering* e o *client side rendering*. Vejamos as particularidades de cada um deles:

🚦 **Server side rendering** (SSR), em tradução literal “**renderização pelo lado do servidor**”, neste padrão a renderização do conteúdo ocorre no servidor web antes que a página seja enviada para o navegador do usuário. É o padrão de renderização que o PHP adota e, com isso, temos alguns benefícios, como:

- SEO (Search Engine Optimization) amigável:** o conteúdo é renderizado no servidor, tornando mais fácil para os motores de busca indexarem o conteúdo da página.
- Carregamento Inicial Rápido:** a primeira renderização da página é mais rápida porque o servidor envia o HTML já pré-renderizado.
- Suporte Universal:** funciona bem para aplicativos que têm uma presença forte em SEO e precisam de suporte para navegadores que não executam JavaScript.



Fonte: imagem adaptada e traduzida de tothenew.com.

🚦 Client Side Rendering

Por outro lado, o **Client-Side-Rendering (CSR)**, em tradução literal “**Renderização pelo lado do cliente**”, faz com que o conteúdo da página ocorra no próprio navegador do cliente após o recebimento do HTML básico e da estrutura da página. Dos pontos positivos desta abordagem temos:

DESENVOLVIMENTO DE SISTEMAS WEB II

a) Experiência do usuário interativa: permite a criação de aplicativos web altamente interativos e dinâmicos, onde as atualizações de conteúdo ocorrem sem a necessidade de recarregar a página.

b) Economia de recursos do servidor: reduz a carga do servidor, pois a maior parte do processamento ocorre no cliente.

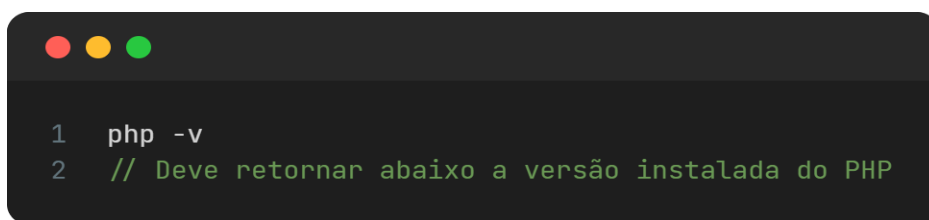
Ao mesmo tempo, temos um **SEO menos amigável**, pois os motores de busca podem ter dificuldade de indexar o conteúdo que é dinâmico, além de **possuir um carregamento mais lento**, afinal a renderização ao lado do cliente precisa buscar e renderizar os scripts necessários.

2. Primeiros Passos com o PHP

Para escrever o primeiro código PHP, precisamos **instalar a linguagem no sistema operacional**. Para isso, iremos utilizar um artigo da *alura.com.br* que traz de forma sucinta e gráfica a instalação da linguagem em Windows, Linux ou MacOS. Acompanhe!

Passo a passo:

1. Acesse o artigo: [Link artigo da alura - PHP: Instalação ao primeiro código.](#)
2. Leia a introdução.
3. Acesse a seção correspondente ao seu sistema operacional.
4. Execute o passo-a-passo da seção.
5. Verifique se a instalação funcionou rodando o comando `php -v`, no seu terminal.



```
1  php -v
2  // Deve retornar abaixo a versão instalada do PHP
```

3. Escolhendo o ambiente de desenvolvimento

Esta etapa é de fundamental importância, pois envolve a escolha do ambiente de desenvolvimento.

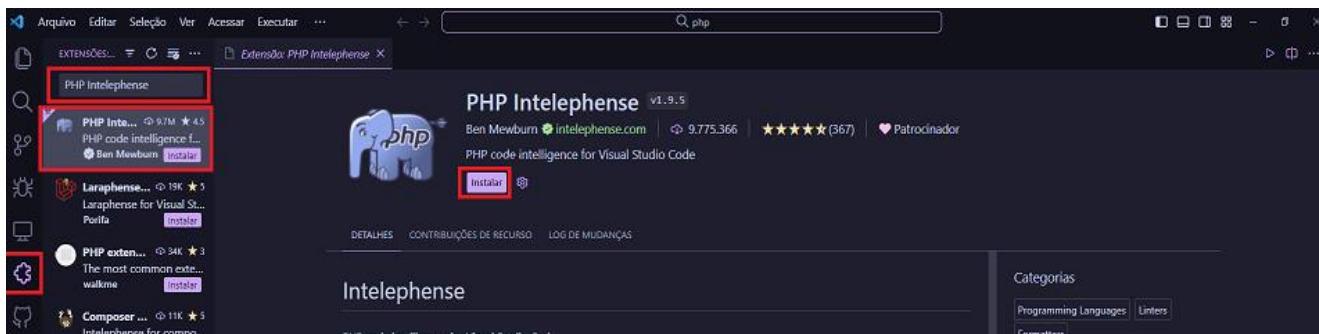
3.1 Visual Studio Code

É um editor de código gratuito, leve, acessível, de fácil personalização e configuração. É a ferramenta que vamos utilizar para desenvolver nossos códigos.

Para utilizar:

1. Acesse o link: <https://code.visualstudio.com/Download>.
2. Escolha o instalador correspondente ao seu sistema operacional.

3. Após instalado, abra o VSCode.
4. Acesse a **aba de extensões, no menu lateral na esquerda**.
5. Na barra de busca, **pesquise por “PHP Intelephense”**, acesse o resultado.
6. Em seguida, clique em **“instalar”**.
7. Pronto! Agora você tem a **marcação da sintaxe PHP** no VSCode.



3.2 PHP Storm

Trata-se de uma IDE completa voltada para o desenvolvimento PHP, porém tem um custo de utilização. É uma ferramenta bastante popular na comunidade PHP, porém não acessível ao público geral, e atualmente a ferramenta conta com uma avaliação gratuita de 30 dias.

Para utilizar:

1. **Acesse o link abaixo:** <https://www.jetbrains.com/pt-br/phpstorm/download/#section=windows>
2. **Escolha o instalador correspondente ao seu sistema operacional**

3.3 Escrevendo o primeiro código PHP

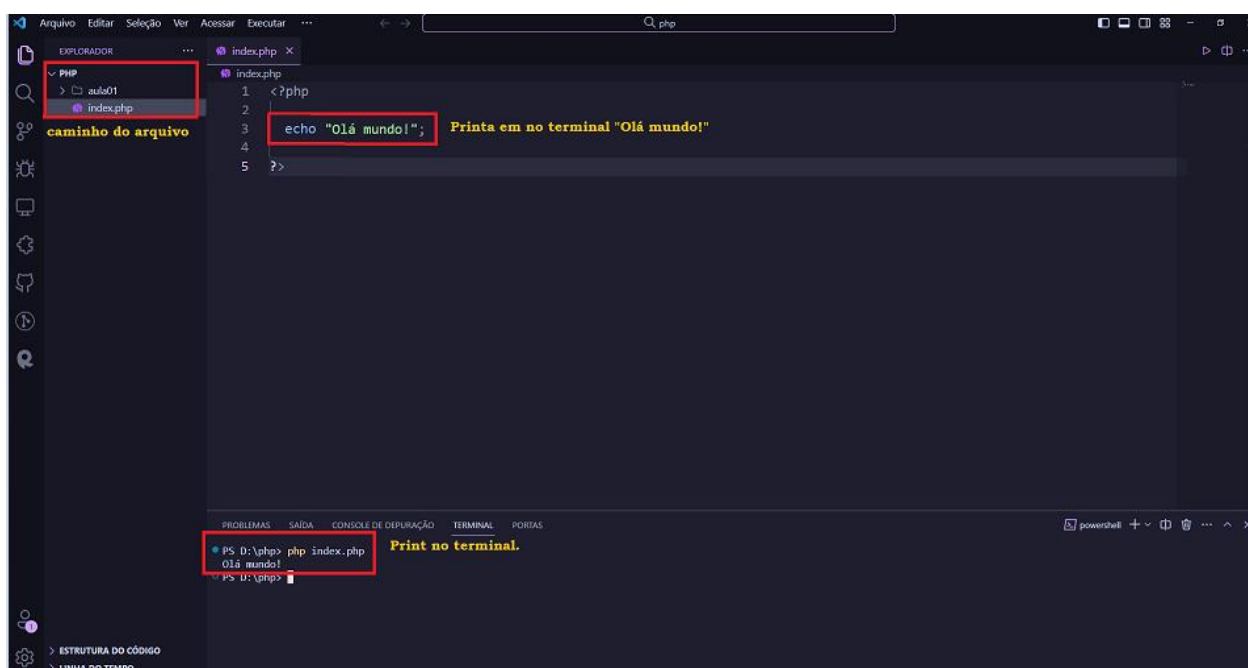
1. Entre no vscode.
2. Adicione um arquivo "hello-world.php".
3. Dentro do arquivo faça o código a seguir:

```
<?php //tag de abertura do php

echo "Olá mundo!"; //printa em tela "Olá mundo!"

?> //tag de fechamento do php
```

4. Abra um terminal na pasta do seu projeto para executar o código PHP.
5. Execute o comando: **php <nome-do-arquivo>.php**
6. No seu terminal, deve retornar o texto "Olá mundo! "

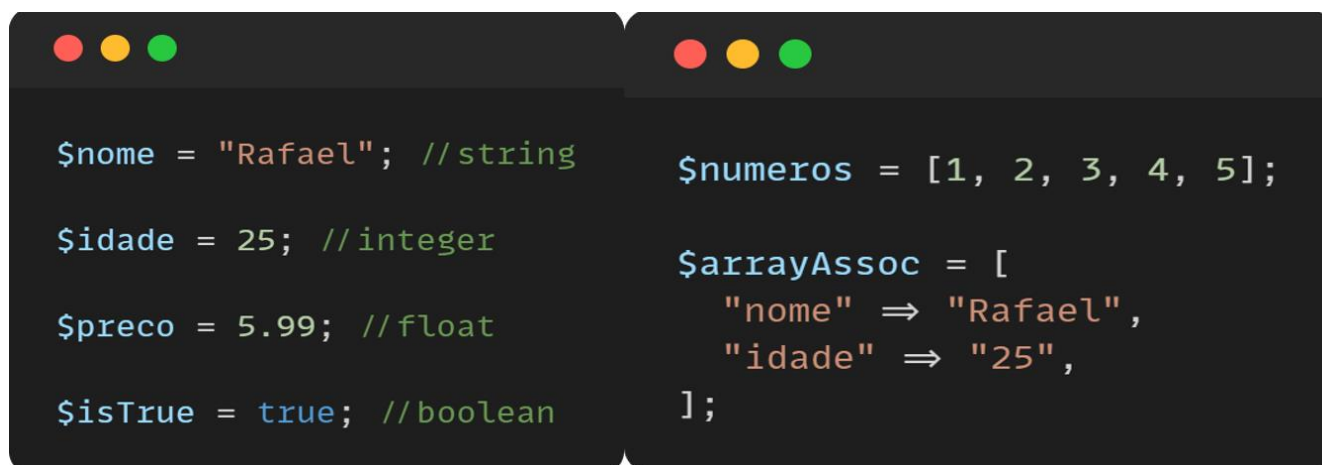


3.4 Conhecendo a sintaxe do PHP

Quanto à declaração de variáveis, temos que no PHP não inserimos tipos nas variáveis. Elas, por padrão, recebem seu tipo de acordo com o valor atribuído. Dos tipos mais usados do PHP temos: **integer**, **float**, **string**, **boolean**, **array**.

Note que para definir uma variável no PHP colocamos "\$", antes de declarar o nome.

Para verificar o tipo de uma variável podemos usar o comando `gettype($variavel);`



3.4.1 Trabalhando com strings

Trabalhamos com *strings* no PHP, inserindo textos envoltos tanto por aspas simples (‘ ’), quanto aspas duplas (“ ”). Para concatenar e exibir o valor de uma variável podemos

concatenar através de ponto (.), ou também, envolvendo a *string* com **aspas duplas** podemos chamar direto a variável que seu valor aparecerá.

Use o comando **echo** na variável para verificar em tela o resultado do código:

```

1 <?php
2 $nome = "Rafael";
3 $idade = 25;
4
5
6 $exemplo1 = "Meu nome é " . $nome . " Tenho " . $idade . " anos." . PHP_EOL;
7 $exemplo2 = "Meu nome é $nome tenho $idade anos." . PHP_EOL;
8
9 echo $exemplo1;
10 echo $exemplo2;
11

```

Terminal Output:

```

PS D:\php> php index.php
Meu nome é Rafael Tenho 25 anos.
Meu nome é Rafael tenho 25 anos.

```

Note que na imagem acima utilizamos um comando **PHP_EOL**, do inglês *end-of-line*, que resulta em uma quebra de linha. Este comando é muito similar aos caracteres de escape de strings como o **\n**, que define uma “new line” ou nova linha. Desta forma, um exemplo ficou embaixo do outro na imagem do código desenvolvido acima.

3.4.2 Operações matemáticas e operadores no PHP

Os operadores para realizar cálculos matemáticos no PHP são muito similares aos de outras linguagens de programação do mercado. Seguem exemplos:

```

<?php

$idade = 20;
$idadeDaqui10Anos = $idade + 10;

$soma = 2 + 2; // 4
$subtracao = 2 - 2; // 0
$multiplicacao = 2 * 2; // 4
$divisao = 2 / 2; // 1

$doisAoCubo = 2 ** 3; // "**" indica potencia de...

$restoDaDivisao = 10 % 3; // O resto da divisão de 10/3 = 1.

echo $idadeDaqui10Anos; // veja o resultado em tela das suas variáveis

```

DESENVOLVIMENTO DE SISTEMAS WEB II

Note que para realizar uma atribuição de valor a uma variável usamos o “**=**”, que é um **operador de atribuição simples**. É possível utilizar de outros operadores de atribuição para simplificar a sintaxe:

a) Operadores de atribuição: atribuem valores em uma variável. Possuem uma sintaxe menor e são bastante usados.

Tipo da atribuição	Código
Adição (+=)	\$x = 10; \$x += 5; // \$x = 15 (10 + 5)
Subtração (-=)	\$x = 20; \$x -= 7; // \$x = 13 (20 - 7)
Multiplicação (*=)	\$x = 4; \$x *= 3; // \$x = 12 (4 * 3)
Divisão (/=)	\$x = 24; \$x /= 2; // \$x = 12 (24 / 2)
Módulo (%=)	\$x = 17; \$x %= 5; // \$x = 2 (17 % 5)
Concatenação (.=)	\$str = "Olá, "; \$str .= "Mundo!"; // \$str = "Olá, Mundo!"

b) Operadores de comparação/lógicos: comparam valores e/ou *strings*, e são muito utilizados com estruturas condicionais.

Operador	Finalidade
==	igual (compara igualdade)
!=	não igual (compara desigualdade)
&&	E lógico
 	OU lógico
!	Não lógico
>	maior que
>=	maior ou igual que
<	menor que
<=	menor ou igual que

3.5 Estruturas de controle

O PHP oferece diversas estruturas de controle que permitem que você controle o fluxo de execução de um programa. Essas estruturas são essenciais para a criação de programas PHP eficientes e funcionais. As principais estruturas de controle no PHP são:

3.5.1 IF-ELSE e ELSE-IF

```

1  <?php
2
3  $idade = 25;
4
5  //idade é maior ou igual a 18?
6  if ($idade ≥ 18) {
7      echo "Você é maior de idade.";
8  } else {
9      echo "Você é menor de idade.";
10 }
```

A estrutura condicional IF-ELSE permite **executar um bloco de código se uma condição for verdadeira** e outro bloco de código se a condição for falsa.

No exemplo ao lado, o bloco que printa em tela “Você é maior de idade” será executado, pois o resultado da expressão lógica dentro do if ($\$idade \geq 18$) é verdadeira (true), pois 25 é maior que 18.

Como uma extensão do IF-ELSE temos o **ELSE-IF que adiciona mais uma condição se a primeira não for realizada**. Perceba que, neste exemplo ao lado, foi adicionada mais uma comparação na linha 8 para verificar **se a idade fornecida é um valor válido**, ou seja, maior que zero.

É possível também desenvolver um código de comparação **com apenas um if**. O número de comparadores vai depender muito do objetivo do seu programa.

```

1  <?php
2
3  $idade = 25;
4
5  //idade é maior ou igual a 18?
6  if ($idade ≥ 18) {
7      echo "Você é maior de idade.";
8  } else if ($idade ≤ 0) {
9      echo "Você forneceu um valor de idade inválido.";
10 } else {
11     echo "Você é menor de idade.";
12 }
```

3.5.2 Switch

A estrutura SWITCH é usada para testar várias condições e executar diferentes blocos de código com base no valor de uma expressão. **Confira o exemplo abaixo:**

No código abaixo temos casos (**cases**), que são basicamente condições envoltas ao bloco de código do *switch*. A variável que terá seu valor comparado será a `$diaSemana`, e caso o seu valor bata com a condição nos **cases** então entrará no bloco da condição, executará o código dentro e abordará o bloco switch por conta do comando break. **O resultado será:** “Hoje é segunda-feira.”

```

1  <?php
2
3  $diaSemana = "segunda";
4
5  switch ($diaSemana) {
6      case "segunda":
7          echo "Hoje é segunda-feira.";
8          break;
9      case "terça":
10         echo "Hoje é terça-feira.";
11         break;
12     default:
13         echo "Dia da semana indefinido.";
14 }

```

3.5.3 Repetição com FOR

A estrutura de repetição FOR permite que você execute um bloco de código várias vezes, com base em uma condição. Aqui está um exemplo simples:

Exemplo de contador com for:

The screenshot shows a code editor with a file named `estruturas-condicionais.php`. The code is as follows:

```

1  <?php
2
3  for($contador = 1; $contador <= 10; $contador++) {
4      echo "Número: $contador" . PHP_EOL;
5  }

```

Annotations in the code highlight the three parts of the for loop: **Inicializador** (Initializator), **Condição** (Condition), and **Incremento/Decremento** (Increment/Decrement).

Below the code, the terminal output is shown:

```

PS D:\php> php estruturas-condicionais.php
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
Número: 7
Número: 8
Número: 9
Número: 10

```

A yellow box highlights the output, and a yellow text label **Resultado de cada iteração** (Result of each iteration) points to it.

Perceba que após declarar o for temos que fornecer **3 valores, separados por “;”**:

1. **Valor inicial**, neste caso, fornecemos 1, pois queríamos que a primeira iteração apresentasse o número 1 no terminal.
2. **Uma condição que determina o fim do laço de repetição**, no código acima, a lógica foi o `$contador` ser **menor ou igual** a 10, então finalize.
3. **Um incremento OU decremento para o valor inicial** a cada iteração, ou seja, para cada vez que o loop executasse acrescentava 1 ao valor atual. Basicamente o mesmo que: `$contador = $contador + 1`.

3.5.4 Repetição com While

A estrutura de repetição WHILE executa um bloco de código enquanto uma condição for verdadeira. **Confira o exemplo.**

```
33 $contador = 1;
34
35 // O $contador é menor OU igual a 5?
36 while ($contador <= 5) {
37     echo "Contador: $contador" . PHP_EOL;
38     $contador++;
39 }
```

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO **TERMINAL** PORTAS

```
PS D:\php> php estruturas-condicionais.php
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
```

Note que, enquanto a condição fosse verdadeira, o bloco de código dentro do while foi executado. Desta forma, foi mostrado em tela um **texto estático + o valor da iteração** de **\$contador**. A cada iteração **\$contador** tinha seu valor incrementado a 1 por conta do operador incremental “++”. Ou seja, **\$contador = \$contador + 1;**

3.5.5 Repetição com Do-WHILE

A estrutura de repetição DO-WHILE é semelhante à estrutura WHILE, mas garante que o bloco de código seja executado pelo menos uma vez, mesmo se a condição for falsa posteriormente.

Perceba que no exemplo ao lado, mesmo **\$contador** sendo menor que 5 o bloco de código envolto em **DO** foi executado ao menos uma vez mostrando “Contador: 10 no terminal”.

```
42 $contador = 10;
43 do {
44     echo "Contador: $contador \n";
45     $contador++;
46 } while ($contador < 5);
```

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO **TERMINAL** PORTAS

```
PS D:\php> php estruturas-condicionais.php
Contador: 10
PS D:\php>
```

Vamos praticar! Topa alguns desafios para fixação?!

1. Mostre em tela (**echo**) todos os números ímpares de 0 até 100. Realize uma quebra de linha para cada número.
2. Faça um código que mostre a tabuada de um número em tela até 10. Realize quebra de linhas para cada multiplicação.

Exemplo:

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
...
```

DESENVOLVIMENTO DE SISTEMAS WEB II

3. Calcule o IMC de uma pessoa, levando a fórmula em consideração, logo após mostre em tela a classificação do IMC, com base no valor calculado.

$$\text{IMC} = \frac{\text{Peso (kg)}}{\text{Altura (m)}^2}$$

IMC	Classificação
Menor que 18,5	Magreza
18,5 a 24,9	Normal
25 a 29,9	Sobrepeso
30 a 34,9	Obesidade grau I
35 a 39,9	Obesidade grau II
Maior que 40	Obesidade grau III

Fonte: tuasaude.com

4. Armazene o **nome do atendente** em uma variável, o **nome do cliente** e o **nome da empresa**. Mostre em tela uma mensagem de saudação dinâmica, ou seja, que se adapta com base nos valores recebidos nas variáveis.

Exemplo:

“Olá, <\$nomeCliente>! Eu sou <\$nomeAtendente> aqui na <\$nomeEmpresa>, em que posso te ajudar? ”

5. Armazene três notas de um aluno. Faça a média destas notas. Devolva o resultado da sua avaliação com base nos critérios abaixo:

Exemplo: “Seu conceito de avaliação foi: <\$avaliacao>”

Conceito da Avaliação	Média
A	Maior ou igual 8 E menor ou igual a 10
B	Maior ou igual a 6 E menor que 8
C	Maior ou igual a 3 E menor que 6
D	Maior ou igual a 0 E menor que 3
Indefinida	Menor que 0 OU maior que 10

Confira a resolução dos desafios:

→ <https://github.com/RafaelR4mos/php-desafios-I>

4. Referências

Múltiplos autores: PHP: **Manual do PHP**, 2023. Disponível em:

<https://www.php.net/manual/pt_BR/>. Acesso em: 22 de setembro de 2023.

FERNANDES, Arthur: **PHP: da instalação ao primeiro código**, 2023. Disponível em:

<<https://www.alura.com.br/artigos/php-instalacao-primeiro-codigo>>. Acesso em: 22 de setembro de 2023.

Múltiplos autores: **Documentation for Visual Studio Code**, 2023. Disponível em:

<<https://code.visualstudio.com/docs>>. Acesso em: 22 de setembro de 2023.

Múltiplos autores: PhpStorm: IDE PHP e editor de código da JetBrains, 2023. Disponível em: <<https://www.jetbrains.com/pt-br/phpstorm/>>. Acesso em: 22 de setembro de 2023.

TAYLOR, Dan: **Client-Side Vs. Server-Side Rendering**, 2023. Disponível em:

<<https://www.searchenginejournal.com/client-side-vs-server-side/>>. Acesso em: 23 de setembro de 2023.

VASHISHTHA, Rahul: **Server-side Rendering vs Static Site Generation in Next.js**, 2022. Disponível em: <<https://www.tothenew.com/blog/server-side-rendering-vs-static-site-generation-in-next-js/>>. Acesso em: 23 de setembro de 2023.