

Programação Web II

PHP 7.4

Inclui a biblioteca PDO

Banco de Dados MySQL

Marcos Roberto de Moraes
Rodrigo Henrique Martins
Abril de 2021



Todos os direitos reservados.

O conteúdo pode ser citado em outras obras de acordo com a legislação atual. Trechos de autoria de conhecimento público foram referenciados.

Programação Web II PHP 7.4 (livro eletrônico): inclui a biblioteca PDO : banco de dados MySQL / Marcos Roberto de Moraes, Rodrigo Henrique Martins

São Paulo, Edição dos Autores, 2021

ISBN 978-65-00-20916-7

Moraes, Marcos Roberto de
Programação Web II PHP 7.4 [livro eletrônico] :
incluir a biblioteca PDO : banco de dados MySQL /
Marcos Roberto de Moraes, Rodrigo Henrique
Martins. -- 1. ed. -- Mogi Mirim, SP :
Ed. dos Autores, 2021.
PDF

Bibliografia
ISBN 978-65-00-20916-7

1. Banco de dados 2. Linguagens de programação
(Computadores) 3. Linguagens PHP 4. Programação
(Computadores) 5. WEB (Linguagem de programação)
I. Martins, Rodrigo Henrique. II. Título.

21-62699

CDD-005.1

Índices para catálogo sistemático:

1. Programação de computadores : Processamento de
dados 005.1

Professores Autores

Olá, leitor e estudante!

Seja bem-vindo ao curso técnico de nível médio no Centro Paula Souza. Neste curso, você contará com o apoio de nossos coordenadores, professores especialistas e tutores presenciais, que estarão sempre disponíveis para te ajudar quando precisar.

Na disciplina em questão, estudaremos programação para web com ênfase no lado servidor. Isso significa que, além dos fundamentos de programação para web e da linguagem de marcação HTML (lado cliente), que você já conheceu em Programação para Web I, você também aprenderá sobre as outras partes que compõem um sistema web.

Para o desenvolvimento dos sistemas web, utilizaremos a linguagem PHP, pois é uma linguagem popular, de fácil aprendizado e open source. Além disso, de acordo com o Índice do TIOBE, PHP está entre as dez linguagens mais populares do mundo. Isso significa que ao aprender PHP, você estará adquirindo habilidades altamente valorizadas no mercado de trabalho.

Apr 2021	Apr 2020	Change	Programming Language	Ratings	Change
1	2	▲	C	14.32%	-2.40%
2	1	▼	Java	11.23%	-5.49%
3	3		Python	11.03%	+1.72%
4	4		C++	7.14%	+0.36%
5	5		C#	4.91%	+0.16%
6	6		Visual Basic	4.55%	-0.18%
7	7		JavaScript	2.44%	+0.06%
8	14	▲	Assembly language	2.32%	+1.16%
9	8	▼	PHP	1.84%	-0.54%
10	9	▼	SQL	1.83%	-0.34%

Neste material, nós, professores Me. Marcos Roberto de Moraes e Esp. Rodrigo Henrique Martins, temos como objetivo explorar os conceitos apresentados na disciplina de Programação para WEB II de forma aprofundada e proporcionar uma série de exercícios

para reforço da aprendizagem. Nós acreditamos que, através deste material, você será capaz de compreender e aplicar os conceitos aprendidos de forma eficaz.

É importante lembrar que, para obter sucesso nesta jornada, é fundamental que você dedique tempo e esforço aos estudos. Além de ler e compreender o conteúdo apresentado neste material, é recomendado que você resolva os exercícios propostos e, caso tenha dificuldades, não hesite em buscar ajuda dos professores ou tutores.

Desejamos-lhe todo o sucesso nesta jornada e estamos ansiosos para acompanhar seu desenvolvimento e progressos na disciplina de Programação para WEB II.

Atenciosamente,

Prof. Me. Marcos Roberto de Moraes e
Prof. Esp. Rodrigo Henrique Martins.

Sumário

Professores Autores	2
PHP	8
Histórico da linguagem	9
Características da Linguagem	10
Bases de Dados	10
Open Source	11
Preparando o Ambiente	12
Ambiente Windows	12
Ambiente Linux Ubuntu (ou Mint)	15
Visual Studio Code no Windows	17
Visual Studio Code no Linux (Ubuntu ou Mint)	18
Testando o PHP	18
Entendendo o código	20
O comando echo	21
Executando o código diretamente pelo interpretador	21
Exercícios de Fixação	22
Cliente X Servidor	23
Exemplo de um request	24
Processo de Renderização	25
Exercícios de Fixação	26
Variáveis e Tipos de Dados	27
Os Dados	27
Variáveis	28
Exemplos válidos e inválidos	28
Constantes	29
Constantes Inteiras	29
Segundo Exemplo	30
Comentários	32
Terceiro Exemplo	32
Dados Alfanuméricos (Strings)	34
Quarto Exemplo	34
Quinto Exemplo	35
Interpretação de variáveis	36

Exercícios de Fixação	37
Operadores	38
Operadores Aritméticos	39
Operandos	39
Operadores Binários	41
Operadores de Comparaçāo	42
Operadores de Atribuição	43
Operadores Lógicos	45
Exercícios de Fixação	46
Estruturas de Controle	47
Estruturas de Decisão	48
Comando if	48
Comando switch	51
Comando if ternário	52
Exercícios de fixação	53
Estruturas de repetição	54
Tipos ou modelos de repetição	55
Comando While	55
Comando do...while	56
Comando for	57
Comando foreach	59
Controlando o Fluxo da execução	62
Exercícios de fixação	63
Escopo de Variáveis	65
Variáveis estáticas em PHP	66
Conversão de variáveis	67
Tabela de conversão (cast)	68
Exercícios de Fixação	68
Funções	70
Porque usamos funções ?	70
Funções integradas	71
Por que usar funções definidas pelo usuário?	71
Resumo	74
Exercícios de fixação	74

Array em PHP	76
Exercícios de Fixação	80
Formulários HTML	82
Elementos de um formulário	83
O elemento <input>	83
O elemento <label>	84
O elemento <select>	84
O elemento <textarea>	87
Enviando dados para um arquivo php	89
Método GET	95
Método POST	95
Exercícios de fixação	95
Sessions	97
O que é uma sessão PHP?	97
Iniciar uma sessão PHP	98
Obter valores de variáveis de sessão PHP	98
Modificando uma variável de sessão	101
Destruindo uma sessão PHP	101
Exercícios de Fixação	103
PHP com MySQL	106
Instalando o SGBD MySQL no Windows	107
A licença do XAMPP	107
Instalando o SGBD MySQL no Linux	115
Configurando o MySQL no Linux	118
Fazer login no servidor MySQL e criar um banco de dados	122
Conhecendo o PDO	123
Nota sobre as versões de APIs para Banco de Dados	124
PDO é Robusta	124
Conexão com Banco de Dados	124
Sobre os Erros	125
Criando um Banco de Dados	125
Aplicativo Workbench	126
Exercícios de Fixação	137
Aplicativo CRUD	138

Estrutura do Projeto em PHP	139
Estrutura do Banco de dados	140
Arquivo init	142
Arquivo functions	143
Arquivo index	144
Arquivo formAdd	147
Arquivo add	149
Arquivo formEdit	153
Arquivo edit	155
Arquivo delete	156
Exercícios de fixação	157
Considerações Finais	158
Repositório de Códigos	159
Erros ou faltas	159
Bibliografia	161

PHP

PHP é uma linguagem de programação Open Source de uso geral, criada em 1995 por Rasmus Lerdorf. O nome "PHP" é um acrônimo recursivo que significa "PHP: Hypertext Preprocessor". Ela é utilizada para o desenvolvimento de aplicações web que geram conteúdo dinâmico dentro do HTML.

Originalmente, a linguagem PHP era usada apenas para o desenvolvimento de aplicações no lado do servidor, capazes de gerar conteúdo dinâmico na World Wide Web, e foi criada para substituir um conjunto de scripts Perl que o criador utilizava em sua página pessoal.



Uma característica importante da linguagem PHP é que ela é uma linguagem de programação de domínio específico, ou seja, sua principal área de aplicação é o

desenvolvimento web. Embora existam variantes da linguagem, como o PHP-GTK, que são utilizadas para o desenvolvimento de aplicações desktop, o propósito principal do PHP é fornecer soluções web eficientes e simples.

Além disso, é uma linguagem interpretada, o que significa que ela é lida e executada diretamente pelo computador, sem precisar ser compilada antes. Isso facilita o desenvolvimento e o debugging de aplicações web.

Histórico da linguagem

Acompanhe a linha de tempo desde a criação da linguagem.

Versão Principal	Histórico
1.0	1995 - a linguagem é apresentada à comunidade pelo seu autor.
2.0	1997 - Considerada pelo seu criador como a "mais rápida e simples ferramenta" para criar páginas dinâmicas para a Web.
3.0	1998 - O desenvolvimento passou a ser feito por vários desenvolvedores em colaboração. Zeev Suraski e Andi Gutmans (Zend Framework) reescreveram toda a base do PHP nesta versão.
4.0	2001 - Foi adicionado um sistema de análise sintática (parser) chamado de motor Zend (Zend engine)
5.0	É adicionado o Zend Engine II, um novo modelo de objeto. Esta versão foi lançada em 2004 e com várias versões intermediárias lançadas até 2015.
7.0	2015 - Após muitos anos de desenvolvimento, finalmente a nova versão da linguagem foi lançada, porém com uma performance surpreendente. A versão não trouxe apenas melhorias em performance, mas também novas funcionalidades, além de implementar e fortalecer novos recursos na orientação a objetos.
8.0	O PHP 8.0 é uma atualização importante da linguagem. Ela contém muitos novos recursos e otimizações, incluindo argumentos nomeados, tipos de união, atributos, promoção de propriedade do construtor, expressão de correspondência,

operador nullsafe, JIT e melhorias no sistema de tipos, tratamento de erros e consistência

Características da Linguagem

Algumas características que se pode destacar sobre PHP:

- Velocidade e robustez.
- Orientação a objetos.
- Portabilidade - independência de plataforma - escreva uma vez, rode em qualquer lugar.
- Tipagem dinâmica. Na versão 8 é possível declarar explicitamente.
- Sintaxe similar a C/C++ e o Perl.
- Open-source.
- Server-side (O cliente manda o pedido e o servidor responde em página HTML).

Existem versões do PHP disponíveis para os seguintes sistemas operacionais, como::

- Windows,
- Linux,
- Mac OS,
- OS/2,
- AS/400,
- Novell Netware,
- RISC OS, IRIX e Solaris.

Nota: A Wikipédia funciona sobre um software inteiramente escrito em PHP, usando bases de dados MySQL: o MediaWiki.

Bases de Dados

Construir uma página dinâmica baseada em bases de dados é simples com PHP. Ele provê suporte a um grande número de bases de dados, tais como:

- Oracle,
- Sybase,
- PostgreSQL,

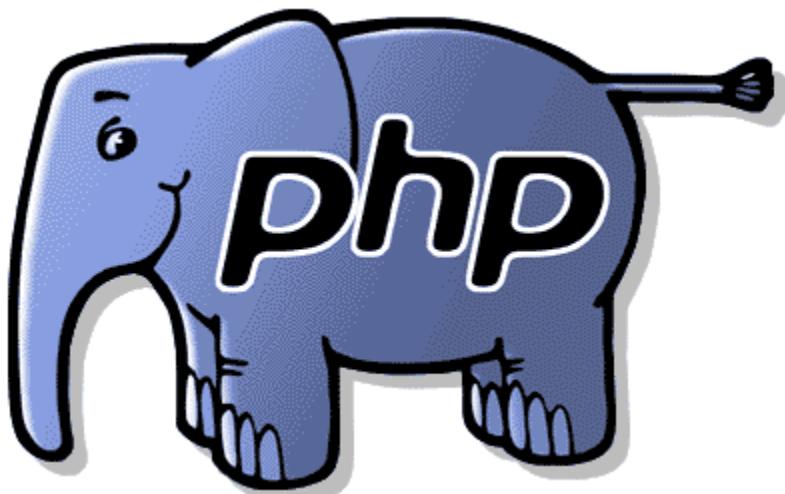
- InterBase,
- MySQL,
- SQLite,
- MSSQL etc, podendo abstrair o banco com a biblioteca ADOdb,
- entre outras.

Open Source

Uma das principais vantagens do PHP é que ele é distribuído gratuitamente através do site oficial <http://www.php.net>. Nele, é possível encontrar as versões mais recentes para download. Além disso, o código-fonte é aberto e a documentação completa do software também está disponível no site. Neste material, serão explicados detalhadamente os passos para realizar a instalação do PHP tanto no sistema operacional Windows quanto no Linux. Ainda, é possível encontrar imagens do logotipo do PHP no site oficial, cópia abaixo:



O Mascote do PHP é o **ElePHPant**.



Preparando o Ambiente

Neste tópico apresentaremos como preparar o seu ambiente de desenvolvimento. Serão necessárias as instalações de:

1. PHP 7.4 (versão estável da linguagem)
2. Visual Studio Code

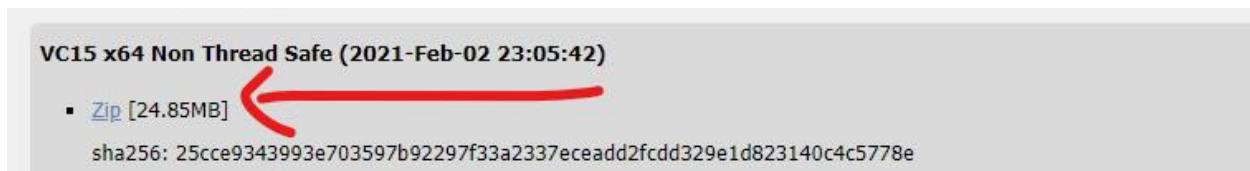
Ambiente Windows

Para a instalação no Windows, acesse o link oficial da comunidade:

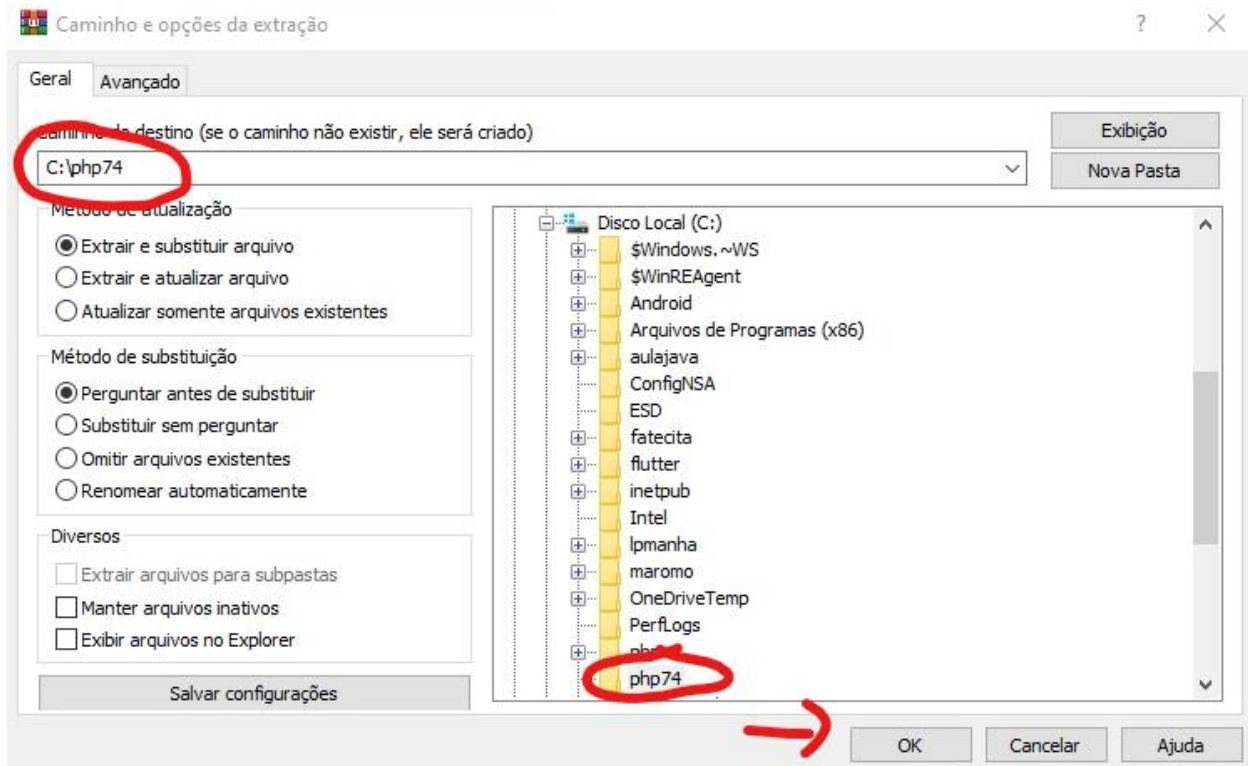
<https://windows.php.net/download#php-7.4>

1. Você deve baixar a versão correta para o seu sistema operacional. Aqui são apresentados os passos para instalação na versão de 64 bits do Windows.

Versão usada:



2. Depois de baixado o arquivo compactado, descompacte em uma localização do seu computador. Aqui foram descompactados no diretório “C:\php74”.



3. Localize nesta pasta o arquivo **php.ini-development** e depois renomeie este arquivo para **php.ini**.

Usando um editor de texto, pode ser o bloco de notas, vamos editar este arquivo (php.ini).

4. Busque pela busca pela configuração `memory_limit = 128M` e mude de `128M` para **1G**.

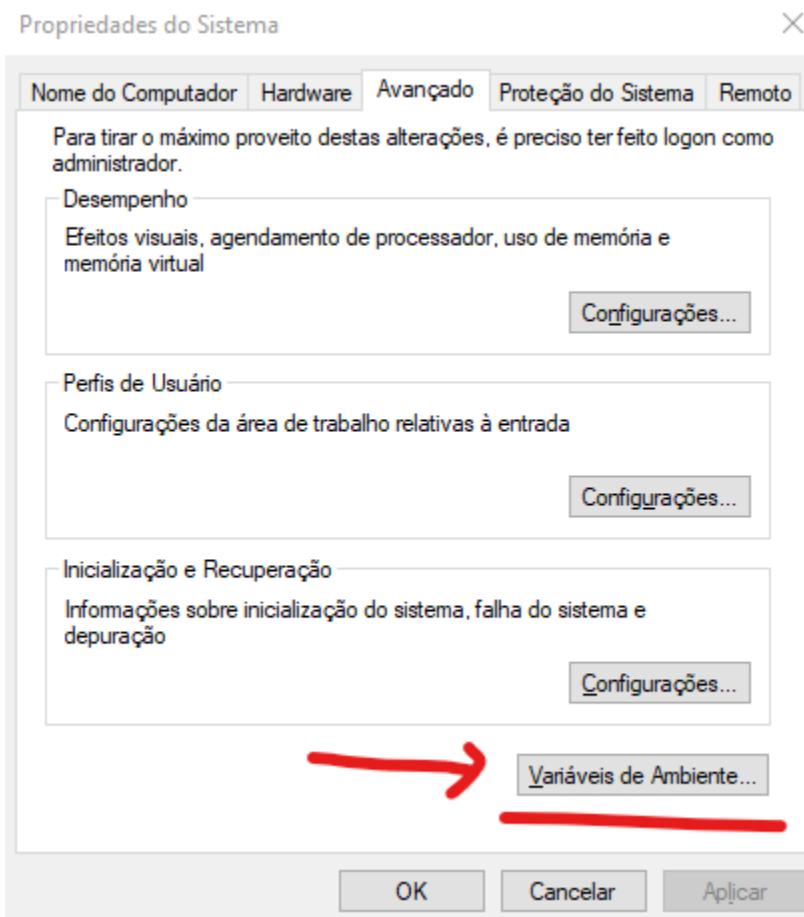
5. Busque também pela linha onde se encontra a configuração `extension_dir = "ext"` e descomente a mesma (retire o caractere ';' do início da linha).

6. Repita o procedimento (descomentar) as seguintes linhas no arquivo `php.ini`:

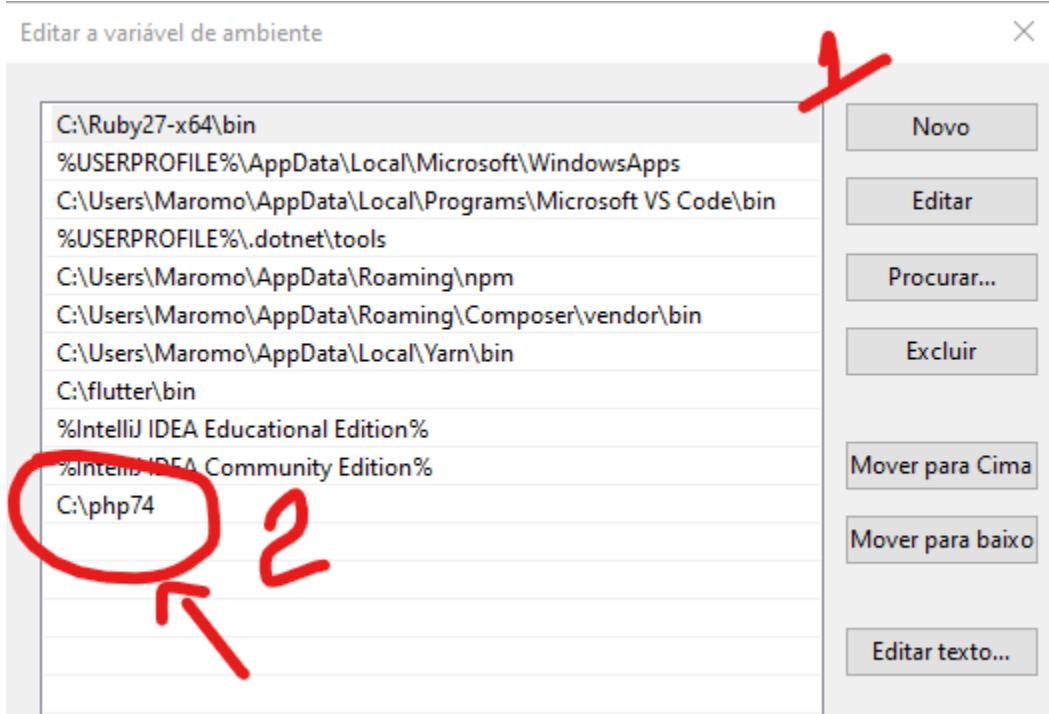
- `extension=gd2`
- `extension= curl`
- `extension=mbstring`
- `extension=openssl`
- `extension=pdo_mysql`
- `extension=pdo_pgsql`

- extension= pdo_sqlite
- extension= sockets

7. Agora é necessário editar as variáveis de ambiente no Windows 10. No seu Painel de Controle do Windows, navegue até **Sistema e Segurança > Sistema > Configurações avançadas** do sistema. Clique em Variáveis de Ambiente.



8. Na nova janela que se abriu, localize a variável PATH, dê um duplo clique nela. Abrirá uma nova janela onde você deve clicar em **Novo** e digitar C:\php74.



9. Agora vamos ver se tudo deu certo. Abra o prompt de comando do Windows e digite o seguinte comando:

```
php --version
```

```
C:\Users\Maromo>php --version
PHP 7.4.15 (cli) (built: Feb 2 2021 20:47:36) ( NTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies

C:\Users\Maromo>
```

Se o resultado for a apresentação da versão instalada, parabéns! o ambiente está configurado com a linguagem PHP.

Ambiente Linux Ubuntu (ou Mint)

Neste tópico apresentaremos como preparar o seu ambiente de desenvolvimento para o sistema operacional Linux.

Para a instalação no Linux, vamos instalar o servidor web Apache que é um dos mais populares no mundo.

1. Instale o Apache usando o gerenciador de pacotes do Ubuntu, apt:

```
$ sudo apt update  
$ sudo apt install apache2
```

Se for a primeira vez que você está usando o sudo nesta sessão, você será solicitado a fornecer a senha do seu usuário para confirmar que você tem os privilégios para gerenciar os pacotes de sistema com o apt. Você também será solicitado a confirmar a instalação do Apache pressionando Y e, depois, ENTER.

2. Após a instalação, você precisará ajustar as configurações de firewall para permitir o tráfego HTTPS. Para listar os perfis de aplicações disponíveis, execute:

```
$ sudo ufw app list
```

Como resultado você deve ver os aplicativos que estão disponíveis.

```
maromo@maromo-Inspiron-5548:~$ sudo ufw app list  
Aplicativos disponíveis:  
Apache  
Apache Full  
Apache Secure  
CUPS
```

- **Apache:** Este perfil abre apenas a porta 80 para tráfego normal web.
- **Apache Full:** Este perfil abre ambas as portas **80** e **443** (tráfego TLS/SSL criptografado).
- **Apache Secure:** Este perfil abre apenas a porta 443 (tráfego TLS/SSL criptografado).
- **CUPS:** gerenciador de impressão que pode ser utilizado via web.

3. Execute o seguinte comando para instalar o PHP 7.4:

```
$ sudo apt install php7.4
```

4. Após a conclusão. você pode confirmar a instalação pelo comando:

```
$ sudo php --version
```

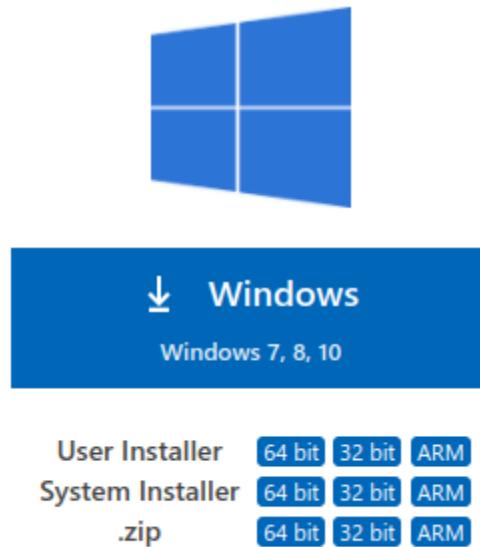
```
maromo@maromo-Inspiron-5548:~$ php --version
PHP 7.4.3 (cli) (built: Oct  6 2020 15:47:56) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
```

Pronto!! sua instalação foi completada.

Visual Studio Code no Windows

Para instalar, baixe o arquivo para o seu sistema operacional de 32 ou 64 bits, no link:

<https://code.visualstudio.com/download>



A instalação no Windows é relativamente simples, basta acessar o site oficial indicado acima e fazer o Download da sua última versão, clicando no botão “Baixar para Windows”. Em seguida, execute o instalador baixado e siga as instruções na tela, não esqueça de marcar a opção "**Add to Path**" para adicionar o Visual Studio Code nas variáveis de ambiente.

Visual Studio Code no Linux (Ubuntu ou Mint)

A instalação no Linux é relativamente simples, basta acessar o link oficial e fazer o Download da sua última versão, clicando no botão "Baixar para Linux".

Como resultado do Download você terá um pacote **.deb** na pasta Downloads. Abra um terminal, navegue até a mesma e execute o seguinte comando.

```
$ sudo apt install ./ARQUIVO_BAIXADO.deb
```

No linux para verificar se a instalação ocorreu sem maiores problemas, execute o comando:

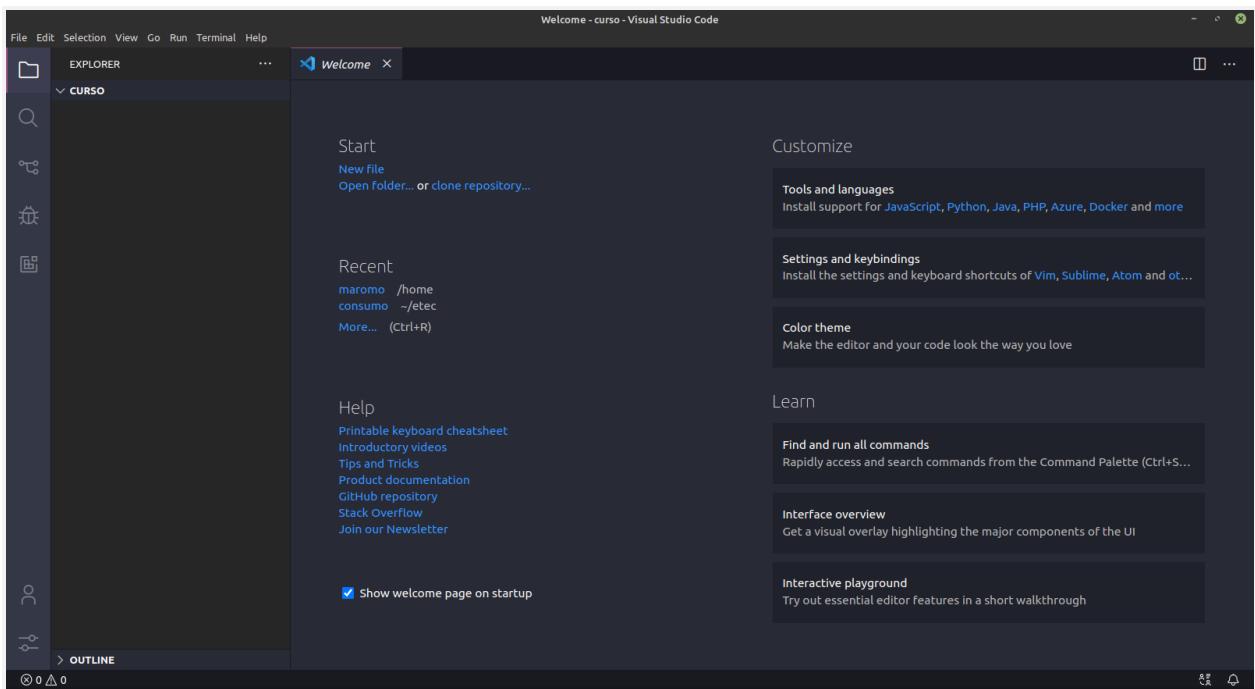
```
$ code .
```

Testando o PHP

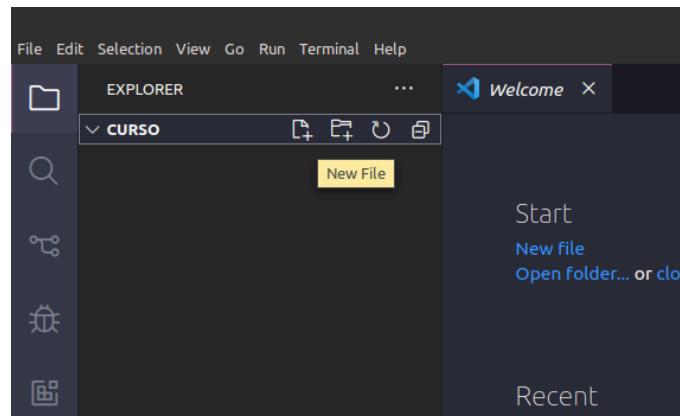
Para criar e editar scripts em PHP podemos utilizar qualquer editor de textos de preferência, ou até mesmo o bloco de notas. Mas neste material optamos por usar o Visual Studio Code.

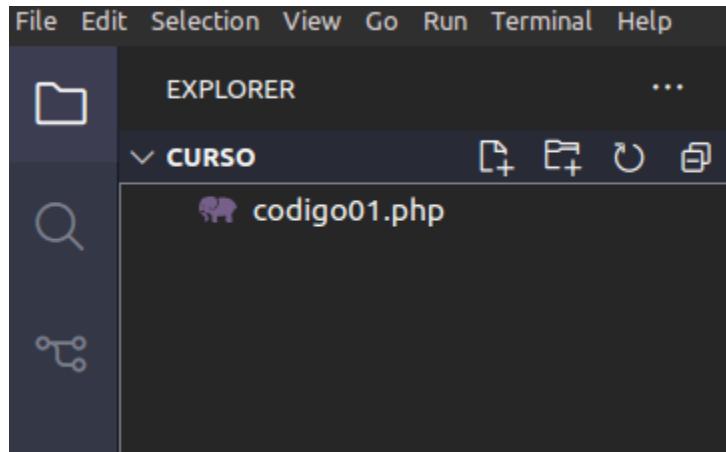
1. Para criar um primeiro script em PHP crie um diretório chamado curso, entre nele e execute o comando (no Windows ou Linux):

```
code .
```



2. Feche a janela Welcome, e clique sobre o botão **New File** para criar um novo arquivo. Chame-o de **codigo01.php**:





3. Agora na janela da direita escreva o seu primeiro script em php. Digitando o que segue:

A screenshot of the VS Code code editor. The title bar shows 'Welcome' and 'codigo01.php'. The editor window displays the following PHP code:

```
1 <?php
2 echo "Olá leitor \n";
```

The code is syntax-highlighted, with 'codigo01.php' in blue, '<?php' in pink, 'echo' in light blue, and the string 'Olá leitor \n' in yellow.

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”.

Neste material, considera-se que o aluno tenha conhecimentos básicos de HTML, quando necessário explicaremos o código.

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

Neste material optou-se pela escolha comum da comunidade. <?php ou <?php ?>

Nota: Em arquivos que possuem apenas código PHP o fechamento da tag é opcional. Aliás, considera-se que nesses casos a TAG php não deve ser fechada, pois assim evitamos a inserção accidental de uma quebra de linha na resposta PHP gerada.

O comando echo

O comando echo retorna uma string para o resultado em HTML. Podemos passar esta string diretamente, como no exemplo, ou uma variável contendo uma string.

Quando o PHP interpreta um arquivo, ele repassa o texto do arquivo até encontrar uma das tags especiais que lhe diz para começar a interpretar o texto como código PHP.

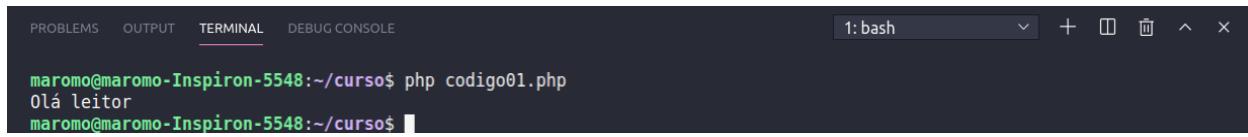
O interpretador então executa todo o código que encontra, até chegar em uma tag de fechamento PHP ou o fim do arquivo.

Executando o código diretamente pelo interpretador

Para executar o código diretamente pelo interpretador PHP no ambiente do Visual Studio Code, acesse o menu **View >> Terminal** ou use as teclas de atalho **[CTRL + SHIFT + ‘]** (acento agudo). Aparecerá o terminal logo abaixo na tela, como visto na figura a seguir.

A screenshot of the Visual Studio Code interface focusing on the terminal tab. The tab bar at the top shows 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is underlined in purple), and 'DEBUG CONSOLE'. Below the tab bar, the terminal window has a dark background. At the top left of the terminal window, it says 'maromo@maromo-Inspiron-5548:~/curso\$'. There is a small cursor icon at the bottom right of the terminal window.

Salve o arquivo digitado **[CTRL + S]**, e depois digite o comando:

A screenshot of the Visual Studio Code terminal showing the output of a PHP script. The terminal window shows the command 'php codigo01.php' being run. The output of the script, 'Olá leitor', is displayed below the command. The terminal window has a dark background with a light gray border around the text area.

Pronto, como resultado você verá a mensagem “Olá leitor” e uma quebra de linha indicada pelo caractere de escape “\n”.

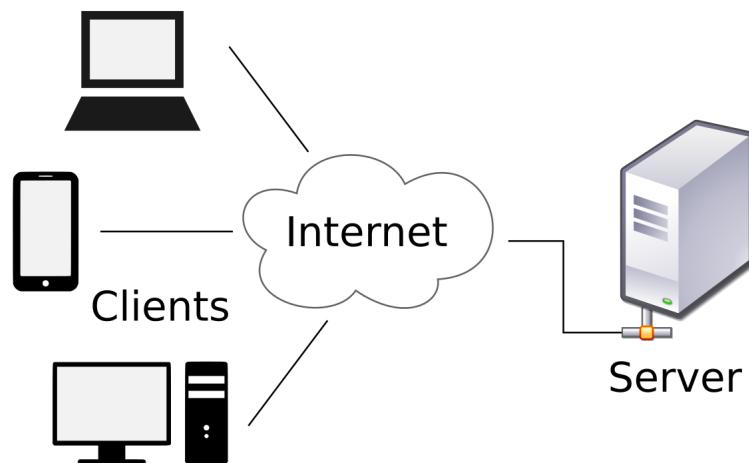
Exercícios de Fixação

1. Anote no bloco de notas, toda a sequência feita e o que precisou complementar para terminar a preparação do ambiente de desenvolvimento no seu computador. Registre quais dificuldades você encontrou e como solucionou.

Cliente X Servidor

O modelo cliente-servidor, ou arquitetura cliente-servidor, é uma estrutura distribuída de aplicativos que divide tarefas entre servidores e clientes. O cliente depende do envio de uma solicitação (request) a outro programa para acessar um serviço disponibilizado por um servidor. O servidor executa um ou mais programas que compartilham recursos e distribuem o trabalho entre os clientes (response).

A comunicação entre cliente-servidor é realizada por um padrão de mensagem de request/response (solicitação/resposta) utilizando um protocolo de comunicação comum, que define formalmente as regras, a linguagem e os padrões de diálogo a serem usados. A comunicação cliente-servidor geralmente segue o conjunto de protocolos **TCP / IP**.



O protocolo TCP mantém uma conexão até que o cliente e o servidor concluam a troca de mensagens. O protocolo TCP determina a melhor maneira de distribuir os dados do aplicativo em pacotes que as redes podem entregar, transfere e recebe pacotes da rede e gerencia o controle de fluxo e a retransmissão de pacotes perdidos ou ilegíveis. IP é um protocolo sem conexão no qual cada pacote transmitido pela Internet é uma unidade independente de dados, não relacionada a nenhuma outra unidade de dados. (IEEE, 2021)

Um servidor pode receber solicitações de muitos clientes distintos em um curto período de tempo. Um computador só pode realizar um número limitado de tarefas a qualquer

momento e depende de um sistema de agendamento para priorizar as solicitações de entrada dos clientes para acomodá-las. Para evitar abusos e maximizar a disponibilidade, o software do servidor pode limitar a disponibilidade para os clientes.

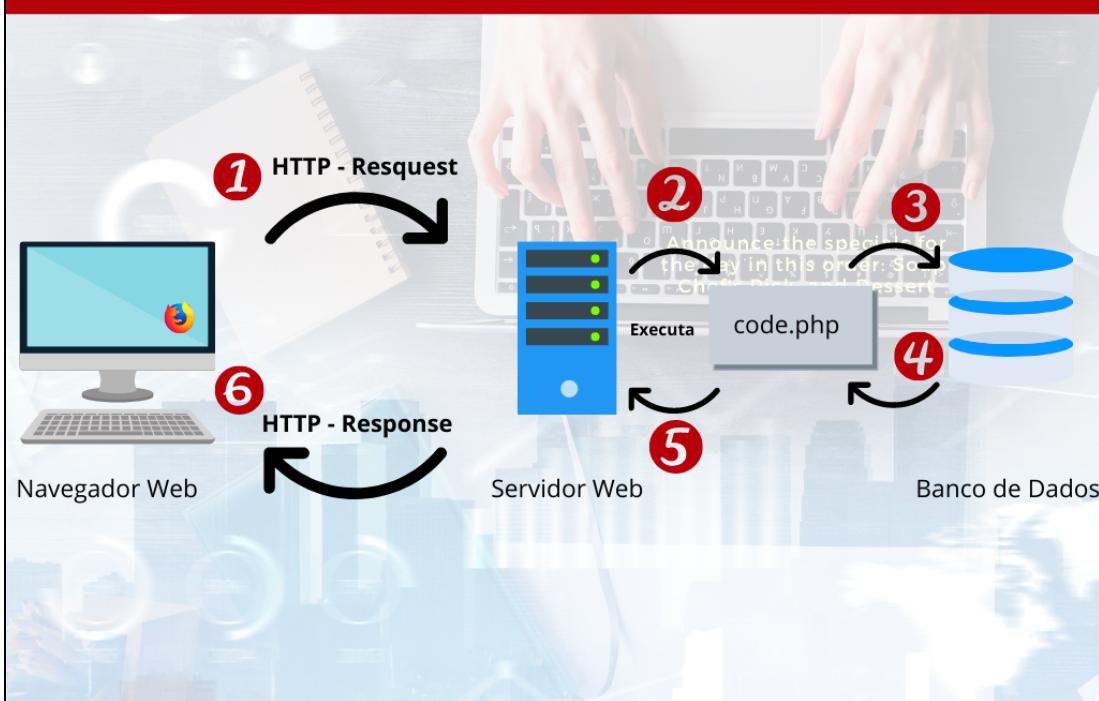
Exemplo de um request



Quando um cliente de um banco acessa serviços on-line com um navegador da web (o cliente), o cliente inicia uma solicitação ao servidor da web do banco (request). As credenciais de login do cliente podem ser armazenadas em um banco de dados e o servidor web acessa o servidor de banco de dados como um cliente. Um servidor de aplicativos interpreta os dados retornados aplicando a lógica de negócios do banco e fornece a saída para o servidor da web. Por fim, o servidor da web retorna o resultado ao navegador da web do cliente para exibição.

Em cada etapa desta sequência de trocas de mensagens cliente-servidor, um computador processa uma solicitação e retorna os dados. Este é o padrão de mensagens de solicitação-resposta. Quando todas as solicitações são atendidas, a sequência é concluída e o navegador da web apresenta os dados ao cliente. O processo de entrega para o cliente web é em HTML, através de um processo de renderização. A próxima figura ilustra todos o processo de solicitação/resposta.

PÁGINA DINÂMICA EM PHP



Processo de Renderização

A função do motor de renderização, também conhecido como motor de layout, é renderizar, ou seja, exibir os conteúdos solicitados na tela do navegador. Esse mecanismo é necessário, pois uma página da web não é uma entidade única que pode ser baixada e exibida na tela um pixel de cada vez. Em vez disso, ela é composta por uma série de instruções escritas em vários tipos de código – HTML, CSS, JavaScript, PHP, entre outras – que dizem ao navegador o que fazer, onde fazer e como fazer.

Cada navegador utiliza um motor de renderização para levar as informações de conteúdo e layout contidas nesses códigos e, em seguida, exibi-las totalmente formatadas e comprehensíveis na tela do seu computador. O grande problema é que cada navegador usa um tipo de motor diferente e as especificações de cada linguagem de

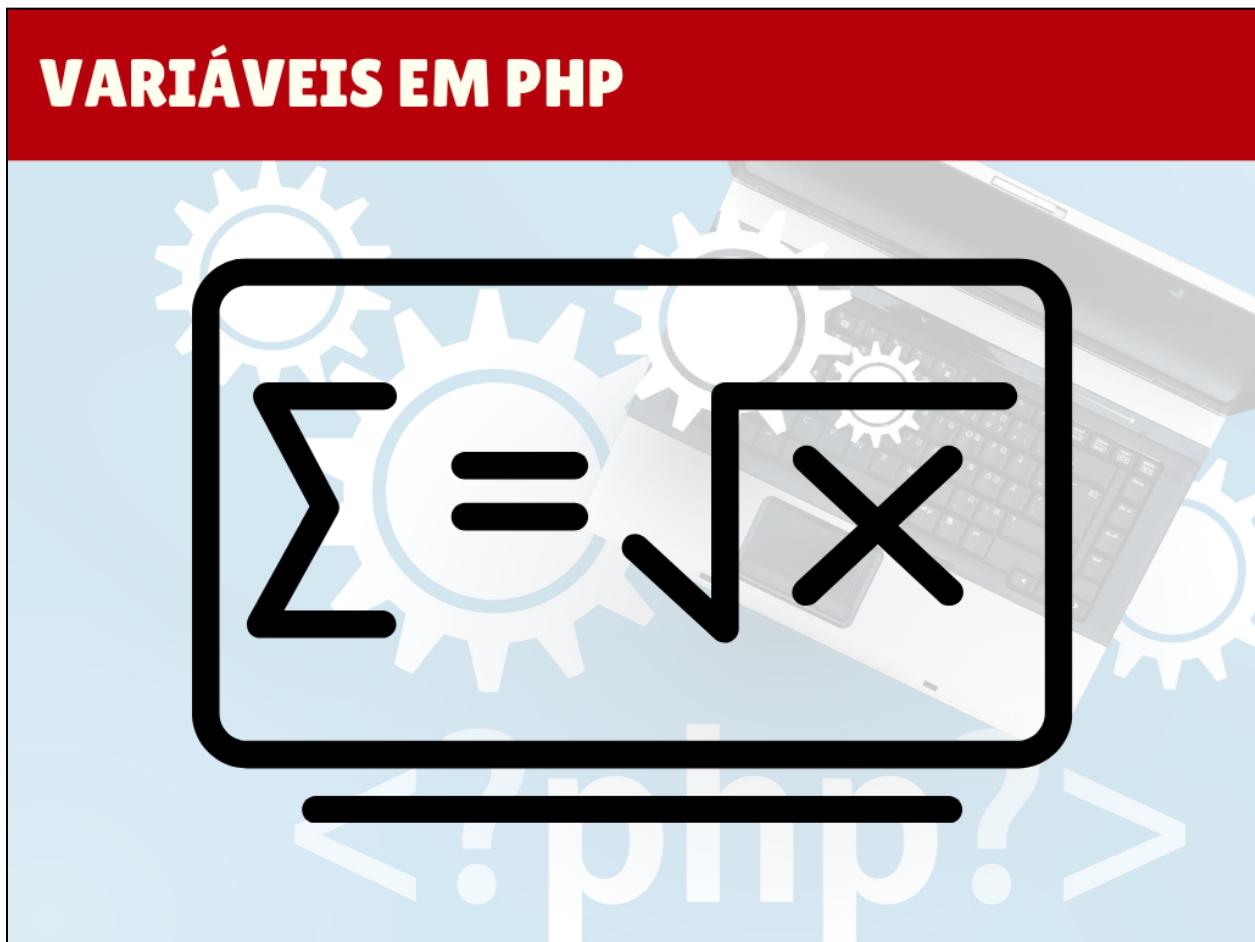
programação são muito detalhadas e cada motor só pode oferecer uma única interpretação desta especificação.

Exercícios de Fixação

1. Descreva com suas palavras o que você entendeu sobre cliente x servidor.
2. O que é o processo de renderização?

Variáveis e Tipos de Dados

Neste capítulo são apresentados os conceitos de tipos de dados, constantes e variáveis. Também apresentamos como realizar a declaração e inicialização de variáveis na linguagem PHP. Também são mostradas as regras de declaração e as convenções utilizadas.



Os Dados

Os dados representam as informações processadas pelo computador. Alguns tipos de dados mais comuns: numéricos (inteiros e reais), caracteres ou literais, lógicos, datas, entre outros, etc.

Os tipos numéricos inteiros são dados numéricos positivos, negativos e não fracionários.

Exemplos: 50, 12, 0, -14.

Já os tipos reais são aqueles também numéricos positivos e negativos, mas, nesse caso, fracionário.

Exemplos: 50.34, 12.14, -14.23.

Os tipos “caracteres” são sequências de letras, números e símbolos especiais, também é conhecido por cadeia de string, literal ou tipo alfanumérico. São representados entre aspas.

Exemplos: “bola”, “Programa”, “19-3806-5555”, etc.

Os tipos lógicos, chamados de booleanos, são dados que representam apenas dois valores: verdadeiro e falso.

Em PHP, é reconhecido pelos termos em inglês, ou seja, true ou false.

Variáveis

Algumas características em PHP:

- As variáveis PHP são representadas por um cifrão (\$) seguido pelo nome do identificador.
- Os nomes de variável no PHP fazem distinção entre maiúsculas e minúsculas, o que chamamos de **case-sensitive**.
- Uma variável pode ter o seu valor modificado durante a execução do código.
- Um nome de variável ou constante válido se inicia com uma letra ou sublinhado, seguido de qualquer número de letras, algarismos ou sublinhados.

Exemplos válidos e inválidos

```
<?php
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";
// exibe "Bob, Joe"
$4site = 'not yet';
// inválido; começa com um número
$_4site = 'not yet';
```

```
// válido; começa com um sublinhado
$täyte = 'mansikka';
// válido; 'ä' é um caracter ASCII (extended) 228
```

Constantes

Constantes são usadas em expressões para representar vários tipos de valores. Existem regras rígidas para determinar como devem ser escritos estes valores. A seguir iremos mostrar as regras para escrever constantes.

Constantes Inteiras

São valores numéricos sem ponto decimal, precedidos ou não por um sinal. Não é possível separar o sinal do valor numérico. Constantes válidas são, por exemplo:

- 1997
- -3
- +5
- 0
- -32000

Alguns exemplos de erros na escrita de constantes inteiras são:

- 1.3 (Não é possível usar ponto decimal).
- - 345 (Não é possível colocar um espaço entre o sinal e o valor numérico).
- 2³ (Não é possível usar notação de expoentes)

Uma constante é um identificador (nome) para um valor único. Como o nome sugere, esse valor não pode mudar durante a execução do script. As constantes são case-sensitive por padrão. Por convenção, identificadores de constantes são sempre em maiúsculas.

Exemplo:

```
<?php
// Nomes de constantes válidos
define("FOO",      "alguma coisa");
define("FOO2",     "alguma outra coisa");
define("FOO_BAR",  "alguma coisa mais");
```

Segundo Exemplo

O próximo exemplo mostra como realizar a declaração de variáveis no PHP. Neste caso o código em php encontra-se dentro de um arquivo no formato HTML5. Lembrando que a extensão do arquivo deve ser .php

Nome do arquivo: codigo02.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exemplo PHP em Arquivo HTML</title>
</head>
<body>
    <?php
        $nome = "Ricardo Antonio dos Santos";
        $cidade = "Campinas";
        $uf = "SP";
        $completo = $cidade . " - " . $uf;
        $num = 10;
        $compra = "15 bolas";
        echo "<br>";
        echo $completo . "<br >";
        echo $num . "<br>";
        echo "Foram compradas $compra <br>";
    ?>
</body>
</html>
```

Salve o arquivo e para executar este arquivo utilizaremos o servidor web a partir do interpretador php na porta 8000, para isso na janela de terminal do Visual Studio Code, digite o comando:

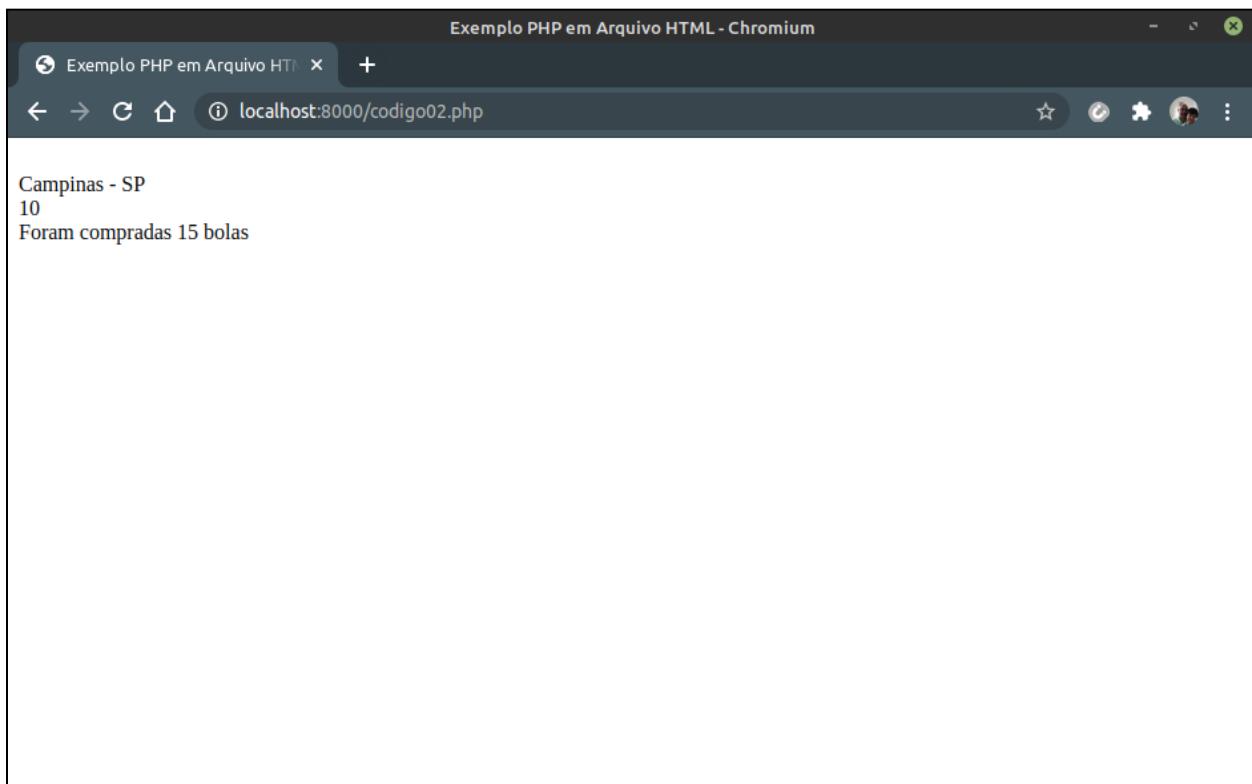
```
$ php -S localhost:8000
```

```
maromo@maromo-Inspiron-5548:~/curso$ php -S localhost:8000
[Sat Apr  3 19:27:17 2021] PHP 7.4.3 Development Server (http://localhost:8000) started
```

Abra o navegador internet de sua preferência e no endereço web digite:

<https://localhost:8000/codigo02.php>

O resultado deve ser uma tela parecida com a seguinte:



Como resultado foi apresentada a impressão da variável `$completo` que mostrou a cidade concatenada (juntada) com a unidade de federação (estado), ou seja, “Campinas - SP”. Depois, foi apresentado o conteúdo da variável numérica `$num` 10 e, por último, a concatenação da frase “Foram compradas “ com a variável `$compra`.

Comentários

Você pode acrescentar comentários para as linhas de código de várias formas, uma delas é a utilização da string “//” antes do comentário. Quando o interpretador PHP encontra essa sequência ele ignora o restante da linha. Exemplo abaixo:

```
<body>
<?php
    //nome completo do cliente
    $nome = "Ricardo Antonio dos Santos";
    $cidade = "Campinas";
```

O PHP suporta comentários do 'C', 'C++' e Unix shell. Por exemplo:

```
<body>
/*
    comentário de mais de uma linha
    script que apresenta os dados de um
    cliente
*/
<?php
    //nome completo do cliente - Estilo C++
    $nome = "Ricardo Antonio dos Santos";
    $cidade = "Campinas";
    $uf = "SP";
    $completo = $cidade . " - " . $uf;
    $num = 10;
    $compra = "15 bolas"; #comentário estilo Unix shell-fim da
linha
```

Terceiro Exemplo

Ainda sobre o assunto tipo de dados, vamos para mais um exemplo. Desta vez utilizaremos duas variáveis que receberão a data e hora atual, respectivamente.

Nome do arquivo: codigo03.php

```

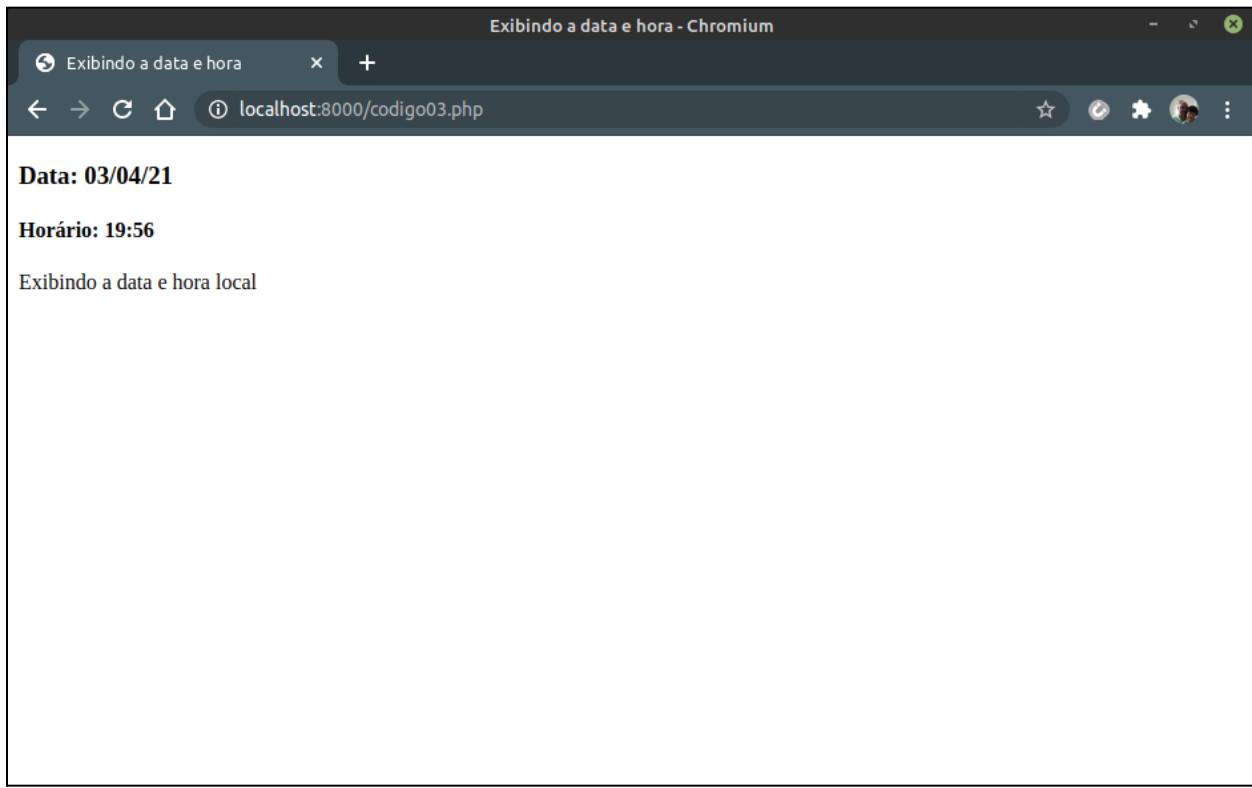
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exibindo a data e hora</title>
</head>
<body>
    <?php
        // Definindo o horário (fuso) para região de São Paulo
        date_default_timezone_set('America/Sao_Paulo');
        $data = date("d/m/y", time());
        $hora = date("H:i", time());
        echo "<h3>Data: $data</h3>";
        echo "<h4>Horário: $hora</h4>";
        echo "<p>Exibindo a data e hora local </p>";
    ?>
</body>
</html>

```

Neste arquivo usamos o comando `date_default_timezone_set` para configurar o fuso horário padrão utilizado por todas as funções de data e hora em um script.

Já a função `date` retorna uma string de acordo com a string de formato informada, usando o inteiro timestamp também informado, ou a hora atual local se nenhum timestamp for informado. Em outras palavras, o parâmetro timestamp é opcional e padronizado para o valor de `time()`.

Resultado da execução no navegador.



Dados Alfanuméricos (Strings)

São sequência de caracteres, que podem ser delimitadas. Delimitadores:

- **Aspas Simples:** delimita qualquer dado alfanumérico;
- **Aspas Duplas:** interpolação de variáveis;
- **Aspas Invertidas:** interpolação de comandos do sistema operacional.

Quarto Exemplo

Neste quarto exemplo vamos explorar as aspas simples.

Nome do arquivo: codigo04.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
```

```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <?php
        echo 'Exemplo de Aspas Simples';
        echo '<br>';
        echo 'Delimita qualquer texto';
        echo '<br>';
        echo 'Em casos especiais quando tivermos uma aspas simples no
meio';
        echo '<br>';
        echo 'Podemos usar duas barras invertidas antes \\';
        echo 'Exemplo de contra barra. Acessou o Maromo\'s site';
    ?>
</body>
</html>

```

Quinto Exemplo

Neste exemplo vamos explorar a string usando a delimitação por aspas duplas.

Se a string for delimitada entre aspas duplas ("), o PHP interpretará a seguinte sequência de escape como caracteres especiais:

Sequências	Significado
\n	fim de linha (LF ou 0x0A (10) em ASCII)
\r	retorno de carro (CR ou 0x0D (13) em ASCII)
\t	TAB horizontal (HT ou 0x09 (9) em ASCII)
\v	TAB vertical (VT ou 0x0B (11) em ASCII) (a partir do PHP 5.2.5)
\e	escape (ESC or 0x1B (27) em ASCII) (a partir do PHP 5.4.4)

\f	form feed (FF ou 0x0C (12) em ASCII) (a partir do PHP 5.2.5)
\\"	contrabarra ou barra invertida
\\$	sinal de cifrão
\"	aspas duplas
\[0-7][1,3]	a sequência de caracteres correspondente a expressão regular é um caractere em notação octal, que silenciosamente é extravasada para caber em um byte (e.g. "\400" === "\000")
\x[0-9A-Fa-f][1,2]	a sequência de caracteres correspondente a expressão regular é um caractere em notação hexadecimal
\u{[0-9A-Fa-f]+}	a sequência de caracteres correspondente a expressão regular é um código Unicode, que será impresso como uma string que representa um código UTF-8 (adicionado no PHP 7.0.0)

Fonte: https://www.php.net/manual/pt_BR/language.types.string.php#language.types.string.syntax.double

Interpretação de variáveis

Quando uma string é especificada dentro de aspas duplas, as variáveis são interpretadas dentro delas.

Há dois tipos de sintaxe: uma simples e uma complexa. A sintaxe simples é a mais comum e conveniente. Provê uma maneira de interpretar uma variável, o valor de um array ou uma propriedade de objeto em uma string com o mínimo de esforço.

A sintaxe complexa pode ser reconhecida pelas chaves envolvendo a expressão.

Se um sinal de cifrão (\$) for encontrado, o interpretador tentará obter tantos identificadores quantos forem possíveis para formar um nome de variável válido. Envolva o nome da variável com chaves para especificar explicitamente o fim do nome.

Vamos ao exemplo.

Nome do arquivo: codigo05.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Exemplo de string com interpolação</title>
</head>
<body>
<?php
$suco = "abacaxi";
echo "Delimita qualquer texto alfanumérico e permite interpolação <br>";
//Veja o exemplo de texto com interpolação de variável
echo "Ele gosta de beber suco de $suco <br>";
?>
</body>
</html>
```

Exercícios de Fixação

1. Desenvolva um script em php que escreva a seguinte frase na tela: MEU PRIMEIRO CÓDIGO EM PHP.
2. Desenvolva um script em PHP que receba três variáveis: nome, sobrenome e idade. E imprima na tela com a seguinte frase: Olá, sou o aluno _____ (mostrar nome e sobrenome), tenho _____ (mostrar idade), e estou cursando o curso Técnico de Desenvolvimento de Sistema.
3. Crie um programa para calcular o estoque médio de uma peça, sendo que: ESTOQUEMÉDIO = (QUANTIDADE MÍNIMA + QUANTIDADE MÁXIMA) /2. Informe os valores das variáveis diretamente no código.
4. Crie um programa para calcular a idade aproximada em anos de uma pessoa. Sendo que IDADE = ANOATUAL – ANONASC. Informe os valores das variáveis diretamente no código.
5. Crie um programa para calcular a seguinte expressão: X:= X ^ 2 + 2. Onde X é informado na declaração da variável.

Operadores

Na programação de computadores, os operadores são construções definidas em linguagens de programação que geralmente se comportam como funções, mas que diferem sintaticamente ou semanticamente.

Exemplos simples incluem aritmética (por exemplo, adição com +), comparação (por exemplo, "maior que" com >) e operações lógicas (por exemplo AND ou && em algumas linguagens). O exemplo de operador mais utilizado é o de atribuição (geralmente = ou :=), já o de acesso a um campo em um registro ou objeto (normalmente .) e o operador de resolução de escopo (frequentemente :: ou .).



Veremos na linguagem de programação PHP os seis tipos de operadores:

- Operadores Aritméticos;
- Operadores Binários;
- Operadores de Comparação;
- Operadores de Atribuição;
- Operadores Lógicos;
- Operadores Ternário.

Operadores Aritméticos

Utilizados para efetuar qualquer operação matemática com dados do tipo numérico: subtrair, multiplicar, etc. Veja a tabela de Operadores:

Operador	Operação
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto

Operandos

O PHP possui também outros operadores aritméticos que atuam sobre um operando.

Veja a tabela:

Operador	Descrição
-a	Inverte o sinal do operando.
++a	Pré-incremento. Primeiro incrementa o valor do operando e depois realiza a operação.
--a	Pré-decremento. Primeiro decremente o valor do operando e depois realiza a operação.

a++	Pós-incremento. Primeiro realiza a operação, depois incrementa o operando.
a--	Pós-decremento. Primeiro realiza a operação, depois decrementa o operando.

Vamos ao exemplo:

Nome do arquivo: codigo06.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exemplo com Operandos</title>
</head>
<body>
    <?php
    $a=10;
    $b=12;
    $c=14;
    $valor1 = ++$a - $b;
    /* Incrementa 1 p/ a = 11 - 12 (valor1) = -1, e A passa p/ 11 */
    $valor2 = $b-- + $a++;
    /* 12 + 12 = 24. Agora decrementa 1 de B ==> 23 e B=11 e A=12 */
    $valor3 = -$a + $c++;
    /* -12 + 14 = 2 Valor3=2 e agora incrementa C = 15 */
    echo "Valor1 => $valor1, <br />Valor2 => $valor2, <br />Valor3 =>
$valor3<br />";
    echo "A => $a, <br>B => $b, <br>C => $c";
    ?>
</body>
</html>
```

Observe os resultados e veja os comentários no código.

Resultados

```
Valor1 => -1,  
Valor2 => 23,  
Valor3 => 2,  
A => 12,  
B => 11,  
C => 15
```

Operadores Binários

Operadores bit-a-bit permitem a avaliação e modificação de bits específicos em um tipo inteiro, na prática permitem manipular o conteúdo das variáveis primitivas de forma binária, ou seja bit a bit. As operações binárias são:

- Complemento de um (^),
- e binário (&), ou binário (|),
- xor (^, ou exclusivo),
- deslocamento a esquerda (<<)
- deslocamento a direita (>>, >>>)

Veja o próximo exemplo comentado.

Nome do arquivo: `codigo07.php`

```
<!DOCTYPE html>  
<html lang="pr-br">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width,  
initial-scale=1.0">  
    <title>Operadores Binários</title>  
</head>  
<body>  
<?php  
$num = 50;
```

```

/* valor 50 em binário = 110010 */
$deslocando1 = $num >> 1; //Desloca 1 bit a direita
/* 110010 deslocar 1 a direita. ==> 11001
Equivale-se a dividir por 2 em decimal */
$deslocando2 = $num << 1; //Desloca 1 bit a esquerda
/* 110010 deslocar 1 a esquerda. ==> 1100100
Equivale-se a multiplicar por 2 em decimal */
$inverte = ~$num; // Inverte os bits
/*Obs.: complemento de dois
~num = -(num + 1) ==> -(50+1)=-51 */
echo "Deslocando1 = $deslocando1<br>";
echo "Deslocando2 = $deslocando2<br>";
echo "Inverte = $inverte<br>";
?>
</body>
</html>

```

Resultados:

Deslocando1 = 25
 Deslocando2 = 100
 Inverte = -51

Operadores de Comparação

Os operadores de comparação são operadores que objetivam analisar os valores de uma expressão, e retornar um valor booleano, ou seja, verdadeiro ou falso.

A tabela a seguir mostra os operadores de comparação.

Operador	Descrição
==	Igual a
!=	Diferente de
<	Menor que
>	Maior que
<=	Menor ou igual a

>=

Maior ou igual a

Analizando se um número é par ou ímpar.

Nome do arquivo: codigo08.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Operadores de Comparaçāo</title>
</head>
<body>
    <?php
        $num = 25;
        if (($num % 2)==0){
            echo "$num é par.<br>";
        }else{
            echo "$num é ímpar.<br>";
        }
    ?>
</body>
</html>
```

Operadores de Atribuição

Atribuição é o termo que usamos quando atribuímos um valor a uma variável. A variável encontra-se à esquerda do operador, e essa recebe o valor gerado pela expressão.

Veja a tabela dos operadores de atribuição e sua descrição.

Operador	Descrição
\$a = \$b	\$a recebe o valor de \$b

<code>\$a += \$b</code>	Equivale a <code>\$a= \$a + \$b</code>
<code>\$a -= \$b</code>	Equivale a <code>\$a= \$a - \$b</code>
<code>\$a *= \$b</code>	Equivale a <code>\$a= \$a * \$b</code>
<code>\$a /= \$b</code>	Equivale a <code>\$a= \$a / \$b</code>
<code>\$a .= \$b</code>	Concatenação => Equivale a <code>\$a= \$a . \$b</code>
<code>\$a %= \$b</code>	Equivale a <code>\$a= \$a % \$b</code>
<code>\$a <= \$b</code>	Equivale a <code>\$a= \$a << \$b</code>
<code>\$a >= \$b</code>	Equivale a <code>\$a= \$a >> \$b</code>
<code>\$a &= \$b</code>	Equivale a <code>\$a= \$a & \$b</code>
<code>\$a = \$b</code>	Equivale a <code>\$a= \$a \$b</code>
<code>\$a ^= \$b</code>	Equivale a <code>\$a= \$a ^ \$b</code>

Vamos a um exemplo.

Optamos a partir deste exemplo em suprimir o conteúdo do HTML e deixar apenas o código em PHP.

Nome do arquivo: codigo09.php

```
<?php
$res = 10;
$a = 10;
$b = 15;
$c = 20;
$res += $a; // $res passa para 10+10 = 20
$res -= $b; // $res passa para 20-15 = 5
$res *= 2; // $res passa para 5*2 = 10
$res .= $a; // $res concatena 10 e 10 = 1010
echo "Valor de res: $res";
```

Operadores Lógicos

São aqueles que retornam o valor verdadeiro ou falso como resultado de uma expressão.

Exemplo	Descrição
\$a and \$b	E (lógico). Retorna verdadeiro somente se \$a e \$b forem verdadeiros.
\$a or \$b	OU (lógico). Retorna verdadeiro se uma das variáveis forem verdadeiras.
\$a xor \$b	OU exclusivo (lógico). Verdadeiro se \$a ou \$b são verdadeiros, mas não ambos.
! \$a	Não (lógico). Inverte o resultado. Se \$a for verdadeiro o resultado será falso. Negação de verdadeiro.
\$a && \$b	E (lógico). Retorna verdadeiro somente se \$a e \$b forem verdadeiros.
\$a \$b	OU (lógico). Retorna verdadeiro se uma das variáveis forem verdadeiras.

Vamos a um exemplo.

Nome do arquivo: `codigo10.php`

```
<?php
$a = 15;
$b = 15;
if(($a==15)&&($b==15)){
    echo "Verdadeiro<br />";
}else{
    echo "Falso <br />";
}
```

Resultado:

- Verdadeiro.

Exercícios de Fixação

1. Resolva as expressões lógicas, determinando se a expressão é verdadeira ou falsa.

- a) $(6 < 8) \text{ || } (3 > 7)$
- b) $(5 \geq 6 \text{ || } 6 < 7) \text{ || } (! (a + 5 - 6 == 8))$ {onde $a == 5$ }
- c) $(34 > 9 \text{ && } 5 + u == 34) \text{ || } (5 == 15/3 \text{ && } 8 > 12) == ((u == 29) \text{ && } 8 > 12)$ {onde $u = 29$ }

2. Para $A = V$, $B = V$ e $C = F$, qual o resultado da avaliação das seguintes expressões:

- a) $(A \text{ && } B) \text{ || } (A \text{ || } B)$
- b) $(A \text{ || } B) \text{ && } (A \text{ e } C)$
- c) $((A \text{ || } C) \text{ e } B \text{ || } A) \text{ && } !(B \text{ != } C)$

3. Considere o seguinte trecho de código:

```
<?php  
$a = 09;  
$b = 04;  
$c = 2021;  
echo ++$a;  
echo $b++;  
echo --$c;
```

Qual será os valores exibidos na variáveis $\$a$, $\$b$ e $\$c$ após a execução do script?

4. Faça um script em php, que:

- Pergunte ao usuário se ele está com fome e tem dinheiro?
- Apenas nos casos em que o usuário esteja com fome e possua dinheiro, indique um local que venda comida.

5. Crie um script em php que pergunte a idade do usuário e mostre a mensagem:

- Se ele tiver menos de 16 anos - Não pode votar.
- Se tiver entre 16 e 18 - O voto é facultativo.
- Se tiver entre 18 e 65 - O voto é obrigatório.
- Se tiver mais de 65 - O voto também é facultativo.

Estruturas de Controle

Até agora entendemos o que é um script em PHP, como ele funciona, de que maneira armazenamos os dados na memória do computador através das variáveis e como realizamos operações de diversos tipos por meio dos operadores que as linguagens disponibilizam.

Neste capítulo iremos estudar instruções para o controle de fluxo: instruções de decisão e de repetição.



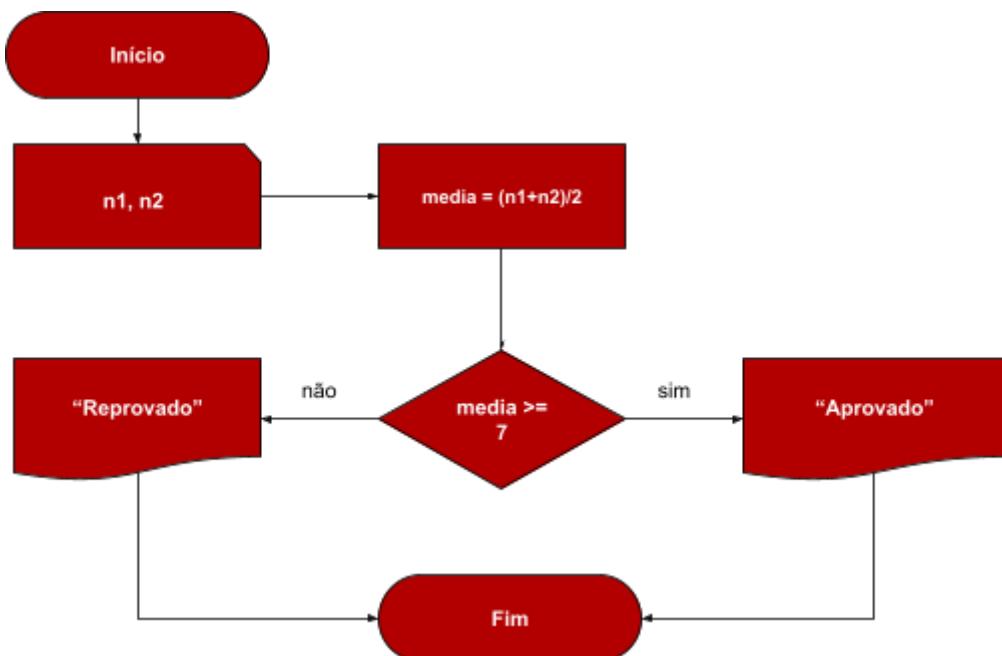
O código de um script só é executado em sequência. Primeiro ele executa uma instrução e, se não houver erro, passa para a próxima instrução, é o que chamamos de estrutura sequencial. Para transferir o controle do programa para um bloco diferente de código se

um certa situação ocorrer, ou durante uma quantidade de vezes repetidas, usamos as Estruturas de Controles.

Estão divididas em dois grupos: **decisão** e **repetição**.

Estruturas de Decisão

Observe o próximo fluxograma para cálculo da média de duas notas de um aluno e que mostra como resultado se o mesmo foi aprovado ou não, dada a seguinte condição: para ser aprovado o aluno deve ter média superior ou igual a sete.



O símbolo de losango deve ser lido como uma pergunta. No caso da figura, a pergunta feita é se a média é maior ou igual a sete, as setas indicam o caminho das respostas possíveis, ou seja, se a resposta for “sim” é exibida a mensagem “Aprovado”, caso contrário, é exibida a mensagem “Reprovado”.

Comando if

Comando que avalia uma expressão e dependendo do resultado, é executado um conjunto diferente de instruções.

Sintaxe:

```
if (exp1){
```

```
bloco 1
} elseif (exp2) {
bloco 2
} else {
bloco 3
}
```

Vamos a um exemplo de uso do comando if.

Nome do arquivo: **codigo11.php**

```
<?php
$n1 = $argv[1]; //receber a nota 1 pela Linha de comando
$n2 = $argv[2]; //receber a nota 2 pela Linha de comando
$media = ($n1 + $n2) / 2;
if($media >= 7){
    echo "Aprovado \n"; //mostra a mensagem e pula uma Linha
}else{
    echo "Reprovado \n"; //mostra a mensagem e pula uma Linha
}
```

No exemplo usamos um array de argumentos passados para o script. O array \$argv contém um todos argumentos passados para o script quando executando através da linha de comando. Ao executarmos na linha de comando a próxima sequência:

```
$ php codigo11.php 6 8
```

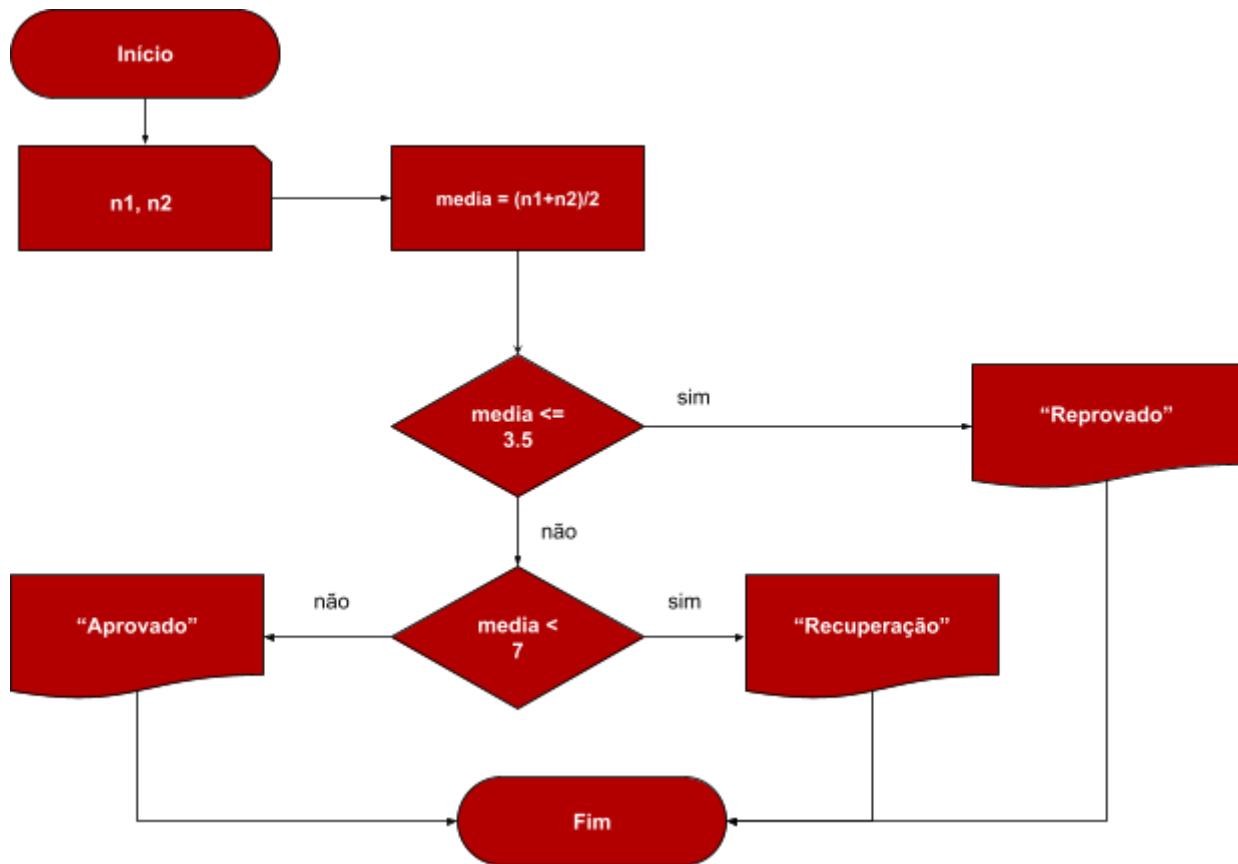
O resultado será a exibição da mensagem “Aprovado” pois a média será 7 e será executado o bloco do comando **if**. Agora caso executemos o comando php, passando os argumentos a seguir:

```
$ php codigo11.php 5 5
```

O resultado será a exibição da mensagem “Reprovado” pois a média será 5. Ou seja, a execução será o bloco do comando **else**.

Podemos também, dentro de uma mesma condição, testar outras condições. Supondo que agora tenhamos uma nova condição, a de **RECUPERAÇÃO**. Dessa forma, se a média for menor ou igual a 3.5, o aluno está “Reprovado”, com média superior a 3.5 e

menor que 7 o aluno está em “Recuperação”, caso seja maior ou igual a 7.0, sua situação será “Aprovado”.



Vamos ao script do fluxograma da nova situação.

Nome do arquivo: codigo12.php

```

<?php
$nota1 = $argv[1]; //receber a nota 1 pela Linha de comando
$nota2 = $argv[2]; //receber a nota 2 pela Linha de comando
$media = ($nota1 + $nota2) / 2;
if($media <= 3.5){
    echo "Reprovado \n"; //mostra a mensagem e pula uma linha
}elseif($media < 7){
    echo "Recuperação \n"; //mostra a mensagem e pula uma linha
}else{
    echo "Aprovado \n"; //mostra a mensagem e pula uma linha
}
  
```

Nota: Teste o código várias vezes de forma a executar cada uma das condições dadas.

Comando switch

O **switch** é parecido com o **if**. Em algumas ocasiões, você tem uma mesma variável a ser testada com valores diferentes, e nesse caso é interessante utilizar o **switch**, que trabalha basicamente com o operador de igualdade.

Sintaxe:

```
switch(operador) {  
    case valor1:  
        <comandos>  
        break;  
    case valor2:  
        <comandos>  
        break;  
    default:  
        <comandos>  
        break;  
}
```

A declaração **switch** é similar a uma série de declarações **if** na mesma expressão. Em muitos casos, se deseja comparar as mesmas variáveis (ou expressões), com diferentes valores, e executar pedaços diferentes de código, dependendo de qual valor ela é igual. Esta é exatamente a serventia da declaração **switch**.

Nome do arquivo: **codigo13.php**

Vamos ao exemplo do comando switch.

```
<?php  
$opcao = $argv[1];  
switch ($opcao) {  
    case 0:  
        echo "Escolha foi 0 \n";  
        break;  
    case 1:  
        echo "Escolha foi 1 \n";
```

```
break;
case 2:
    echo "Escolha foi 2 \n";
    break;
default:
    echo "Escolha foi inválida \n";
}
```

A instrução **break** termina a execução do **switch** e o programa continua a executar na instrução seguinte. O uso do **break** evita testar as demais alternativas de forma desnecessária quando uma opção verdadeira já foi encontrada.

O comando **default** exibe uma mensagem, caso nenhuma das alternativas anteriores seja verdadeira.

Execute pelo terminal passando o argumento referente a opção, exemplo:

```
$ php codigo12.php 2
```

Nota: Teste o código várias vezes de forma a executar cada uma das condições dadas.

Comando if ternário

Ao desenvolver uma aplicação, é comum utilizarmos estruturas condicionais como, por exemplo, **ifs** e **elses** ou **switch case**.

Porém, quando um dos testes que realizamos é simples e que deve retornar apenas um valor para duas possibilidades, o uso do **if** ternário é mais adequado.

Por exemplo. Suponha que tenhamos a necessidade de criar uma funcionalidade para gerar um bônus salarial cuja regra seja esta:

- Se o salário for maior que R\$ 5000, o bônus é de 10%
- Se o salário for menor ou igual a R\$ 5000, o bônus é de 15%

Observe o exemplo com **if** tradicional:

```
<?php
$salario = $argv[1];
if($salario <= 5000.0){
    $bonus = $salario * 0.15;
```

```
}else{
    $bonus = $salario * 0.10;
}
echo "Valor do Bônus: $bonus \n";
```

Veja agora o exemplo usando o if ternário.

Nome do arquivo: codigo13.php

```
<?php
$salario = $argv[1];
$bonus = $salario <= 5000.0 ? $salario * 0.15 : $salario * 0.1;
echo "Valor do Bônus: $bonus \n";
```

Com esse código temos o mesmo resultado de antes.

Mas como funciona esse operador ternário?

A estrutura de um operador ternário é compreendida da seguinte forma:

condição ? valor se for verdadeiro : valor se for falso

No nosso exemplo o valor da variável bônus depende do teste condicional, caso o valor seja menor ou igual a 5000 (condição verdadeira '?') é calculado 15% sobre o valor do salário e atribuído ao bônus, caso contrário (condição falsa ':'), é calculado 10% sobre o valor do salário e atribuído ao bônus.

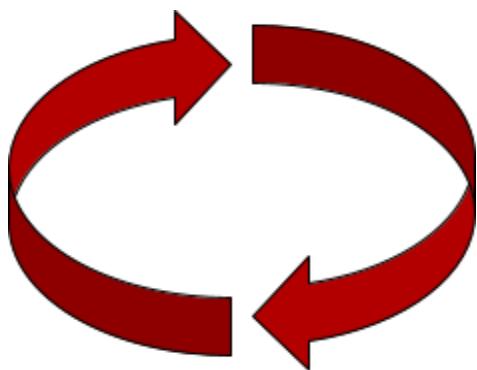
Exercícios de fixação

1. Crie um script que receba dois valores numéricos: a e b. Imprima na tela a mensagem “É divisível.” quando a for divisível por b ou a mensagem “Não é divisível.”, no caso contrário.
2. Crie um script que contenha uma variável: horaDoDia. Essa variável deverá conter a hora do dia e três mensagens deverão ser impressas na tela de acordo com a hora: “Bom dia”, “Boa tarde” ou “Boa noite”. Não se preocupe em capturar a hora do sistema, receba a hora manualmente pela linha de comando.
3. Elabore um script que receba um número. Se positivo armazene-o na variável A e se for negativo em B.

-
4. Construa um script que receba um número e verifique se ele é par ou ímpar. Quando for “par” guarde esse valor em P e quando for “ímpar” guarde em I.
 5. Construa um script que receba uma variável numérica N e imprimi-la, somente se, a mesma for maior que 100, caso contrário imprimir o número 0.
 6. Tendo como dados de entrada a altura e o sexo de uma pessoa. Construa um script que calcule seu peso ideal, utilizando as seguintes fórmulas:
 - a. Para homens: $(72.7 \cdot h) - 58$
 - b. Para mulheres: $(62.1 \cdot h) - 44.7$ (h = altura)
 7. Escreva um programa que receba três números e imprima o maior deles.
 8. Uma empresa paga R\$29,00 por hora normal trabalhada e R\$ 35,00 pela hora extra. Escreva um programa que receba o total de horas normais e o total de horas extras trabalhadas por um empregado em um determinado ano, e calcule o salário a receber por este trabalhador. Caso o salário seja maior que R\$ 1000,00, além do salário, você deve pagar um bônus de R\$ 100,00.

Estruturas de repetição

Uma estrutura de repetição é uma estrutura de desvio do fluxo de controle presente em PHP que realiza e repete diferentes instruções ou ações dependendo se uma condição é verdadeira ou falsa, em que a expressão é processada e transformada em um valor booleano. Estão associados a uma estrutura de repetição uma condição (também chamada "expressão de controle" ou "condição de parada") e um bloco de código: verifica-se a condição, e caso seja verdadeira, o bloco é executado. Após o final da execução do bloco, a condição é verificada novamente, e caso ela ainda seja verdadeira, o código é executado até que a condição não seja mais verdadeira.



Deve-se observar que, caso o bloco de código nunca modifique o estado da condição, a estrutura será executada para sempre, uma situação chamada **laço infinito** (loop infinito). Da mesma forma, é possível especificar uma estrutura em que o bloco de código modifica o estado da condição, mas esta é sempre verdadeira.

Algumas linguagens de programação especificam ainda uma palavra reservada para sair da estrutura de repetição de dentro do bloco de código (**return**), "quebrando" a estrutura, que é o caso do PHP. Também é oferecida por algumas linguagens uma palavra reservada para terminar uma iteração específica do bloco de código, forçando nova verificação da condição (**break**), ou até mesmo voltando ao início do laço para uma próxima iteração (**continue**).

Tipos ou modelos de repetição

1. Repetição pré-testada;
2. Repetição pós-testada;
3. Repetição com variável de controle e;
4. Iteração de coleção.

Comando While

A construção "enquanto" (também chamada "repetição pré-testada") é a mais difundida estrutura de repetição, e sua estrutura básica é a seguinte:

Enquanto (condição) Faça

(bloco de código)

Fim Enquanto

Sintaxe em PHP:

```
<?php  
while(condição){  
    //bloco de comandos  
}
```

O comando avalia a expressão, e enquanto essa expressão for verdadeira executa o bloco de comandos em questão.

Vamos ao exemplo:

Nome do arquivo: **codigo14.php**

```
<?php  
$num = $argv[1];  
while ($num < 30) {  
    echo "Valor atual de num => $num \n";  
    $num++;  
}  
echo "Terminou ... \n ";
```

Teste o código passando pela linha de comando um valor menor do que 30. Teste algumas vezes e verifique o resultado. Enquanto o valor da variável \$num for menor do que 30 o bloco de código é executado, e a cada passagem pelo bloco acrescemos um (1) ao valor da variável, até que ela assuma o valor 30, e encerre o bloco dando continuidade ao script escrito fora do laço.

Comando do...while

A construção "faça {bloco} enquanto" (também chamada "repetição pós-testada") é uma variação da construção anterior, e difere, pois a verificação da condição é feita após uma execução do bloco. Sua estrutura básica é a seguinte:

```
<?php  
do {  
    //bloco de comandos  
}while(condição)
```

O bloco de comandos é executado ao menos uma vez, e enquanto essa expressão for verdadeira executa-se o bloco de comandos novamente.

Vamos ao exemplo:

Nome do arquivo: **codigo15.php**

```
<?php  
$num = $argv[1];  
do {  
    echo "Valor atual de num => $num \n";
```

```
$num++;
}while ($num < 30);
echo "Terminou ... \n ";
```

A diferença entre o código **do..while** em relação ao código **while** visto, reside no fato que se executarmos com valor maior ou igual a 30, ele executa o laço uma vez, condição que não ocorria no comando **while**.

Comando **for**

A construção "for" (ou "repetição com variável de controle") é uma estrutura de repetição que designa uma variável de controle para cada iteração do bloco, e uma operação de passo a cada iteração. Sua estrutura básica é a seguinte:

```
for (inicialização; condição; operação pós execução) {
    //bloco de comandos
}
```

Na construção anterior declara-se uma variável de controle (estado inicial). Em seguida verifica-se a condição a ser testada, caso esta condição seja verdadeira o bloco de comandos é executado, e ao final, volta-se a operação de pós execução, que normalmente afeta a variável dada na inicialização incrementando ou decrementando valores, repetindo a execução do bloco de comandos até que condição se torne falsa.

Vamos ao exemplo:

Nome do arquivo: codigo16.php

```
<?php
//Tabuada de um número passado como parâmetro pelo usuário
$num = $argv[1];
for ($i = 1; $i <= 10; $i++) {
    $tabu = $num * $i;
    echo "$num X $i => $tabu \n";
}
```

No código anterior, script `codigo16.php`, ao executá-lo é criada uma tabuada do valor passado pela linha de comando. Uma variável (`$i`) é inicializada com o valor 1, e em

seguida, testa-se se o valor é menor ou igual a 10. Como é verdadeira a condição, executa-se o bloco de comandos e após a execução é incrementado +1 ao valor da variável de controle (`$i`), e novamente testa o bloco novamente, e assim sucessivamente, até que a condição seja falsa.

Teste com valores diferentes na linha de comando.

Resultado da execução do comando:

```
$ php codigo16.php 7
```

```
7 X 1 => 7
7 X 2 => 14
7 X 3 => 21
7 X 4 => 28
7 X 5 => 35
7 X 6 => 42
7 X 7 => 49
7 X 8 => 56
7 X 9 => 63
7 X 10 => 70
```

Aninhamento de laços

Isso é muito útil quando queremos, por exemplo, apresentar na tela todos os elementos de um array bidimensional. Suponha que temos um array bidimensional de dimensões 2 x 2. Poderíamos imprimir todos os elementos desse array usando duas vezes o comando `for` de forma aninhada, como o próximo exemplo:

Nome do arquivo: codigo17.php

```
<?php
$vetor[0][0] = "elemento 0.0.";
$vetor[0][1] = "elemento 0.1.";
$vetor[1][0] = "elemento 1.0.";
$vetor[1][1] = "elemento 1.1.";
for ($i = 0; $i < 2; $i++) {
    for ($j = 0; $j < 2; $j++) {
        echo "O elemento da posição $i, $j é: ";
        echo $vetor[$i][$j];
        echo "\n";
}
```

```
}
```

```
echo "Encerrou \n";
```

Resultado da execução:

O elemento da posição 0, 0 é: elemento 0.0.
O elemento da posição 0, 1 é: elemento 0.1.
O elemento da posição 1, 0 é: elemento 1.0.
O elemento da posição 1, 1 é: elemento 1.1.
Encerrou

Comando **foreach**

A estrutura "for each" é usada para iterar itens de uma coleção, sendo uma especialização da estrutura "para". Menos flexível que a estrutura "para", esta estrutura torna implícita a atribuição inicial e o incremento do passo, e determina que a condição de parada é somente a situação na qual todos os elementos do conjunto já foram iterados.

Possui duas sintaxes:

```
foreach ($array as $value) {  
    //comandos  
}
```

A primeira forma, itera sobre arrays informados no \$array. A cada iteração, o valor do elemento atual é atribuído a \$value e o ponteiro interno do array avança uma posição (então, na próxima iteração, apontará para o próximo elemento).

```
foreach ($array as $key => $value) {  
    //comandos  
}
```

A segunda forma, adicionalmente, atribui a chave do elemento corrente a variável \$key, a cada iteração.

Vamos ao exemplo:

Nome do arquivo: codigo18.php

```
<?php  
$meses = [  
    'janeiro',  
    'fevereiro',  
    'março',  
    'abril',  
    'maio',  
    'junho',  
    'julho',  
    'agosto',  
    'setembro',  
    'outubro',  
    'novembro',  
    'dezembro',  
];  
foreach ($meses as $mes) {  
    echo "Mês: $mes \n";  
}
```

Ao executar o script anterior, é exibida uma lista de todos os meses do ano, no comando **foreach** cada item é atribuído a variável **\$mes** em cada uma das passagens no laço.

Resultado:

```
Mês: janeiro  
Mês: fevereiro  
Mês: março  
Mês: abril  
Mês: maio  
Mês: junho  
Mês: julho  
Mês: agosto  
Mês: setembro  
Mês: outubro  
Mês: novembro  
Mês: dezembro
```

Os arrays associativos são estruturas onde cada elemento é identificado por uma chave única, então, é indiferente se estamos trabalhando com Arrays Indexados ou Arrays Multidimensionais, ambos serão por definição, estruturas associativas. Veja o próximo exemplo:

Nome do arquivo: codigo19.php

```
<?php
$brasileirao = [
    "Flamengo" => 71,
    "Internacional" => 70,
    "Atlético MG" => 68,
    "São Paulo" => 66,
    "Fluminense" => 64,
    "Grêmio" => 59,
    "Palmeiras" => 58,
    "Santos" => 54,
];
$posicao = 0;
foreach ($brasileirao as $time => $pontos) {
    $posicao++;
    echo "Posição: $posicao - Time: $time : $pontos \n";
}
```

Ao executarmos o código, no bloco do foreach além de apresentar a chave (`$time`) também apresentamos o valor (`$pontos`). A variável `$posicao` apresenta a posição final do time no campeonato.

Resultado da execução:

```
Posição: 1 - Time: Flamengo : 71
Posição: 2 - Time: Internacional : 70
Posição: 3 - Time: Atlético MG : 68
Posição: 4 - Time: São Paulo : 66
Posição: 5 - Time: Fluminense : 64
Posição: 6 - Time: Grêmio : 59
Posição: 7 - Time: Palmeiras : 58
Posição: 8 - Time: Santos : 54
```

Controlando o Fluxo da execução

Para controlar o fluxo de execução dos programas além das estruturas vistas, ainda podemos usar os comandos **break** e **continue**.

O comando **break** é usado para interromper a execução de um dos laços de interação vistos acima ou de um comando **switch**. Este comando é comumente utilizado para produzir a parada de um laço mediante a ocorrência de alguma condição específica, antes da chegada do final natural do laço. Exemplo:

```
$v = [2, 4, 6, 8, 10];
for ($i = 0; $i < 5; $i++) {
    if ($v[$i] < 0) {
        break;
    }
    if ($i == 4) {
        echo "elemento negativo não encontrado. \n";
    }
}
```

No exemplo acima é percorrido todo o vetor à procura de um valor negativo. Como não há elemento negativo no vetor, ao chegar na última posição é apresentada a mensagem “elemento negativo não encontrado”. O que aconteceria se o segundo elemento do vetor fosse negativo? Resposta: ao ler o valor e compará-lo se o valor é menor que zero, ao executar o comando **break** o laço deixaria de ser executado.

Já o comando **continue** é utilizado dentro dos blocos de comandos de repetição para ignorar o restante das instruções pertencentes ao laço corrente, e ir para a próxima iteração (voltando ao início do laço). Veja o exemplo a seguir:

```
<?php
$vetor = [1,1,3,5,8,13,21,34,55,89,144];
for($i=0; $i<sizeof($vetor); $i++) {
    if($vetor[$i] % 2 != 0) { //é ímpar
        continue;
    }
    $num_par = $vetor[$i];
    echo "O número $num_par é par. <br>" ;
```

```
}
```

```
?>
```

Ao executar o bloco de comandos ao encontrar um valor ímpar no vetor é desviado o fluxo para a pós instrução, e assim dando sequência no teste e execução do bloco, desprezando as linhas abaixo do comando continue.

Exercícios de fixação

1. Faça um programa que determine o maior entre N números (declare manualmente).
2. Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos, indagou à rainha se o pagamento poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez, de tal forma que o primeiro quadro deveria conter apenas um grão e os quadros subsequentes, o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Tarefa: Faça um programa para calcular o número de grãos que o monge esperava receber.
3. Faça um programa que conte de 1 a 100 e a cada número divisível exato por 10 emita uma mensagem: “É divisível exato por 10”;
4. Elabore um programa que gere e escreva os números ímpares entre 100 e 200.
5. Construa um programa que gere 500 valores aleatórios. (Pesquise sobre randômico).
 - a. Encontre o maior valor
 - b. Encontre o menor valor
 - c. Calcule a média dos números lidos
6. Faça um programa que dados 05 números como argumentos pelo usuário, mostre quantos são divisíveis exatos por 5.
7. Faça um programa que receba um número pelo usuário, e mostre se o mesmo é um número primo ou não.
8. Faça um programa que mostre o número de números pares (entre 1 e 168) que são divisíveis exatos por 6. Deve-se listar cada um deles no resultado em tela.
9. Elabore um programa que apresente o valor de uma potência de uma base qualquer elevada a um expoente qualquer, ou seja, de b^e .

-
10. Elabore um programa que apresente no final, o somatório dos valores pares existentes entre 10 e 20.
 11. Elabore um programa que apresente as potências de 2, variando de 0 a 10, ou seja $2^0, 2^1, \dots 2^{10}$.

Escopo de Variáveis

Mais adiante, veremos como funcionam as funções em PHP. Porém você já deve conhecer um pouco sobre o assunto.



Quando uma variável é declarada dentro de uma função dizemos que seu escopo é local, seu valor é conhecido apenas dentro da função. Pode-se ter outra função com uma variável definida com o mesmo nome que ocupa posições de memória diferentes.

O escopo de uma variável é o contexto onde ela foi definida. A maioria das variáveis do PHP tem somente escopo local. Por exemplo:

```
<?php  
$a = 1; /* escopo global */
```

```
function Teste() {  
    echo $a; /* referencia uma variável do escopo local (não  
definida) */  
}  
  
Teste();  
?>
```

Ao executar este código a chamada da função Teste ao executar o comando **echo** apresentará a seguinte mensagem PHP Notice: Undefined variable: a teste.php on line 5. Ou seja, a variável (\$a) não foi definida no contexto da função Teste, e sim dentro do programa que está sendo executado. No PHP, as variáveis globais precisam ser declaradas como globais dentro de uma função, se for utilizada em uma. Para que mostrasse o valor 1, o código deveria estar escrito desta forma.

```
<?php  
$a = 1; /* escopo global */  
  
function Teste() {  
    global $a;  
    echo " $a \n " ;  
}  
  
Teste();  
?>
```

Variáveis estáticas em PHP

Variáveis estáticas podem ser declaradas. A partir do PHP 5.6 é possível atribuir valores a essas variáveis, que são resultados de expressões, porém não é possível usar nenhuma função, o que causará um erro de interpretação.

```
<?php  
function estatico(){
```

```
static $inteiro = 0;
static $inteiro = 1+2;
$inteiro++;
echo "Valor do inteiro: $inteiro \n";
}
estatico();
estatico();
?>
```

Ao executar teremos o valor **4** como resultado na primeira chamada e **5** na segunda chamada, ou seja os valores são mantidos a cada execução.

Conversão de variáveis

Se tivermos uma **string** contendo somente números, o PHP somará normalmente esse valor com outra variável numérica. Se houver textos e números em uma string ele usa somente a parte que possui números para efetuar operações aritméticas.

Veja o exemplo:

```
<?php
$num=6;
$string="6";
$texto="6 produtos em estoque";
echo $num+$string . "\n";
echo $string*$texto . "\n";
```

Ao executarmos o código anterior, teremos como resultado:

```
12
PHP Notice: A non well formed numeric value encountered in teste.php
on line 6
36
```

O resultado **12** refere-se a soma de (**\$num**) (variável numérica) com (**\$string**) (variável implicitamente convertida para numérica). Já a mensagem informada sobre a linha 6 indica que encontrava-se em um formato não numérico, apesar disto como o primeiro caractere do texto era o número 6 foi realizada uma multiplicação, cujo resultado foi **36**.

Tabela de conversão (cast)

A tabela abaixo indica os tipos de conversores e a descrição:

Conversor	Descrição
(int), (integer)	converte para inteiro
(bool), (boolean)	converte para booleano
(float), (double), (real)	converte para número de ponto flutuante
(string)	converte para string
(array)	converte para array
(object)	converte para objeto
(unset)	converte para NULL

Observe o próximo exemplo:

```
<?php
    $a = 11.9;
    $b = 12.7;
    $c = 7.3;

    $soma = (int)$a + $b + $c;
    echo "A soma dos inteiros é: $soma \n";
```

A variável (`$a`) que é do tipo float foi convertida para inteiro. Se quiséssemos converter o resultado de toda uma expressão para inteiro bastaria colocar entre parênteses para delimitar a expressão cujo resultado deve ser convertido:

```
$soma = (int) ($a + $b + $c);
```

Execute e veja os resultados.

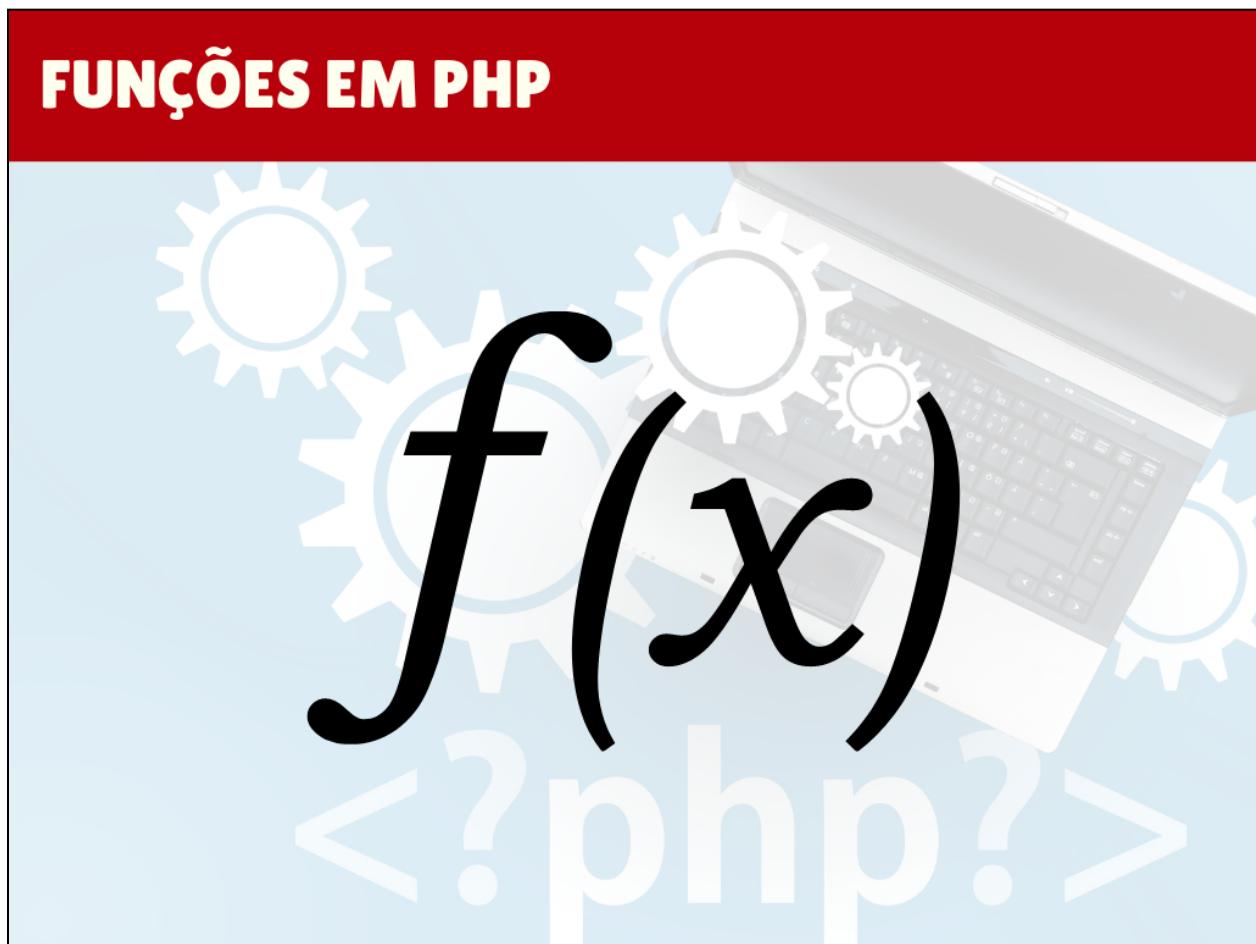
Exercícios de Fixação

1. Faça um programa que dados dois valores como argumentos pelo usuário, concatenando os dois valores como string. Exemplo: 10 10, resulta em 1010.

-
- 2. Faça um programa que receba um salário de uma pessoa com casa decimal, passado como argumento na chamada do programa, exiba o salário como texto.
 - 3. Repita o programa anterior, mas agora apresente o valor do salário sem casas decimais.
 - 4. Dados três argumentos numéricos pelo usuário, converta cada um deles para um número do tipo float e apresente na tela a soma de todos os valores acrescendo 63.88, ao final.

Funções

Uma função em PHP é uma parte ou bloco de código que executa uma ação específica e que pode ser reutilizada. Ela recebe a entrada na forma de parâmetros, executa os comandos e fornece uma saída. As funções podem retornar valores quando chamadas ou podem simplesmente executar uma operação sem retornar nenhum valor. O PHP possui centenas de funções integradas que realizam diferentes tarefas.



Porque usamos funções ?

- **Para melhorar a organização de código** - as funções do PHP nos permitem agrupar blocos de código relacionados que realizam uma tarefa específica.
- **Reutilização** - uma vez definida, uma função pode ser chamada por vários scripts em nossos arquivos PHP.

- **Facilitar a manutenção do código** - as atualizações do sistema só precisam ser feitas em um lugar.

Funções integradas

Funções integradas são funções predefinidas em PHP que existem no pacote de instalação. Essas funções internas do PHP é o que o torna uma linguagem de script muito eficiente e produtiva. As funções internas do PHP podem ser classificadas em muitas categorias.

Abaixo está a lista das categorias.

- **Funções de manipulação de String:** Estas são funções que manipulam dados de string (cadeia de caracteres), consulte o artigo sobre strings para exemplos de implementação de funções de string.
- **Funções numéricas:** As funções numéricas em PHP são as funções que retornam resultados numéricos. Podem ser usadas para formatar números, retornar constantes, realizar cálculos matemáticos, etc.
- **Função de manipulação de datas:** A função de data é usada para formatar a data e hora do Unix em um formato legível por humanos.
- Funções que manipulam banco de dados: existem uma série de banco de dados suportados, para saber sobre cada um deles, recomenda-se consultar a documentação no site oficial: php.net.
- **Outras:** funções que manipulam arquivos, arrays, etc...

Por que usar funções definidas pelo usuário?

As funções definidas pelo usuário são úteis quando:

- Você tem tarefas de rotina em seu aplicativo, como adicionar dados ao banco de dados;
- Realizar verificações de validação nos dados;
- Autenticação de usuários no sistema;
- Entre outras...

Antes de criarmos nossa primeira função definida pelo usuário, vamos examinar as regras que devemos seguir ao criar nossas próprias funções:

- Os nomes das funções devem começar com uma letra ou sublinhado, mas não com um número.
- O nome da função deve ser único.
- O nome da função não deve conter espaços.
- É considerado uma boa prática usar nomes de função descritivos.
- As funções podem, opcionalmente, aceitar parâmetros e também retornar valores.

Vamos agora criar nossa primeira função. Criaremos uma função básica que ilustra os principais componentes de uma função em PHP.

Nome do arquivo: **codigo20.php**

```
<?php
// define uma função que soma dois valores
function somarValores(){
    echo 1 + 2;
}
somarValores();
```

Onde:

- somarValores é o nome da função, que não recebe argumentos de fora;
- o bloco da função começa no { e termina em };
- somarValores() é a chamada da função.

Vejamos agora um exemplo mais complexo que aceita um parâmetro e exibe uma mensagem.

Suponha que queremos escrever uma função que imprima o nome do usuário na tela, podemos escrever uma função personalizada que aceita o nome do usuário e o exibe na tela.

Nome do arquivo: **codigo21.php**

```
<?php
function mostrarNome($nome = ""){
    echo "Olá $nome \n";
}
mostrarNome("João Santos");
```

Resultado:

Olá João Santos

Onde:

... (\$ Name) {...” é o parâmetro da função chamado nome e é inicializado como sem nome. Se nenhum parâmetro for passado para a função, será exibido como “”. Isso é útil se não fornecer nenhum parâmetro à função pode resultar em erros inesperados.

Vejamos agora uma função que aceita um parâmetro e retorna um valor. Vamos criar uma função que converte quilômetros em milhas. Os quilômetros serão passados como parâmetro. A função retornará as milhas equivalentes aos quilômetros passados. O código abaixo mostra a implementação.

Nome do arquivo: codigo22.php

```
<?php  
$kms = $argv[1];  
function converterKmParaMilhas($kms){  
    $milhas = 0.62;  
    return $kms / $milhas;  
}  
echo converterKmParaMilhas($kms) . "\n";
```

Ao executar o script, passe 55 como argumento.

```
$ php codigo22.php 55
```

Resultado:

88.709

Usamos o return (opcional) em duas situações:

1. Quando queremos atribuir o valor retornado a uma variável;
2. Quando queremos testar (comandos condicionais) o valor do retorno de uma função.

A função pode simplesmente executar uma tarefa e não retornar nenhum valor.

Resumo

- Função é um bloco de código que executa tarefas específicas.
- A função interna no PHP é uma função que vem instalada com o PHP
- Funções de string manipulam dados de string
- Funções numéricas manipulam dados numéricos
- Funções de data manipulam dados de data
- Funções definidas pelo usuário são funções que você mesmo pode criar em PHP.

Exercícios de fixação

1. Faça um programa para imprimir a sequência abaixo para um (**n**) informado pelo usuário. Use uma função que receba um valor (**n**) e imprima até a n-ésima linha.

```
1
2  2
3  3  3
.....
n  n  n  n  n  ... n
```

2. Faça um programa para imprimir a sequência abaixo para um (**n**) informado pelo usuário. Use uma função que receba um valor (**n**) inteiro imprima até a n-ésima linha.

```
1
1  2
1  2  3
.....
1  2  3  ... n
```

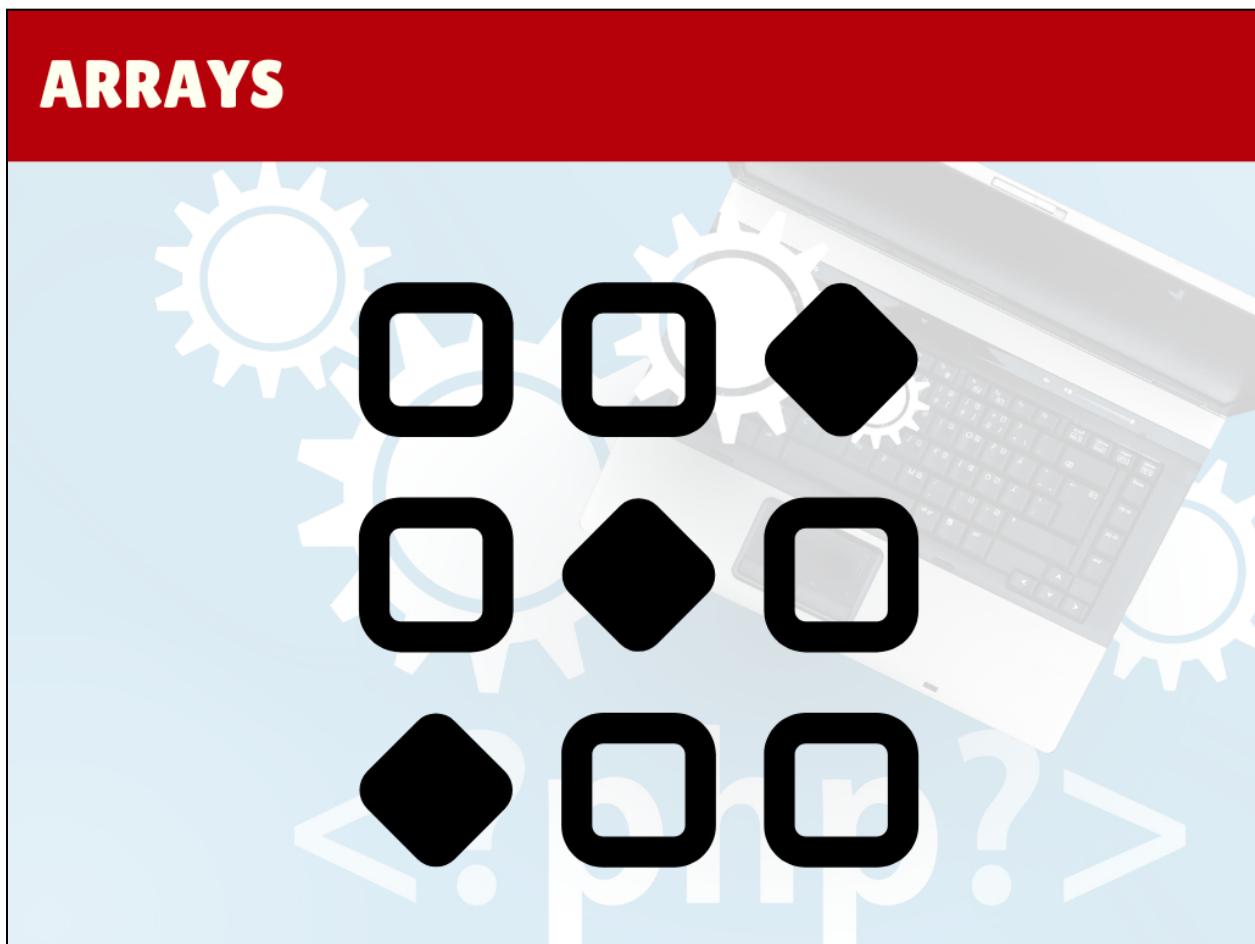
3. Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos.
4. Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.
5. Faça um programa com uma função chamada **somalImposto**. A função possui dois parâmetros formais: **taxalImposto**, que é a quantia de imposto sobre vendas

expressa em porcentagem e **custo**, que é o custo de um item antes do imposto. A função “altera” o valor de custo para incluir o imposto sobre vendas.

6. Faça um programa que converta da notação de 24 horas para a notação de 12 horas. Por exemplo, o programa deve converter 14:25 em 2:25 P.M. A entrada é dada em dois inteiros. Deve haver pelo menos duas funções: uma para fazer a conversão e uma para a saída. Registre a informação A.M./P.M. como um valor ‘A’ para A.M. e ‘P’ para P.M. Assim, a função para efetuar as conversões terá um parâmetro formal para registrar se é A.M. ou P.M.
7. Faça um programa que use a função **valorPagamento** para determinar o valor a ser pago por uma prestação de uma conta. O programa receberá como argumento o valor da prestação e o número de dias em atraso e passar estes valores para a função **valorPagamento**, que calculará o valor a ser pago e devolverá este valor ao programa que a chamou. O programa deverá então exibir o valor a ser pago na tela.
8. Faça uma função que informe a quantidade de dígitos de um determinado número inteiro informado.
9. Faça uma função que apresente o reverso do número informado. Por exemplo: 123 -> 321.
10. Jogo de Craps. Faça um programa de implemente uma primeira jogada do jogo de Craps. O jogador lança um par de dados, obtendo um valor entre 2 e 12. Se, na primeira jogada, você tirar 7 ou 11, você ganhou. Se você tirar 2, 3 ou 12 na primeira jogada, isto é chamado de "craps" e você perdeu. O programa deve retornar a mensagem correspondente a situação ocorrida. Outros valores retornam a mensagem: jogue novamente.

Array em PHP

Um array no PHP é na verdade um mapa ordenado. Um mapa é um tipo que relaciona valores a chaves. Este tipo é otimizado para vários usos diferentes: ele pode ser tratado como um array, uma lista (vetor), hash table (que é uma implementação de mapa), dicionário, coleção, pilha, fila, etc. Assim como existe a possibilidade dos valores do array serem outros arrays, árvores e arrays multidimensionais.



A explicação dessas estruturas está além do escopo deste material, mas veremos alguns exemplos de uso.

Observe o exemplo:

Nome do arquivo: codigo23.php

```

<?php
function paulistas(){
    $clubes = [
        "São Paulo",
        "Palmeiras",
        "Santos",
        "Corinthians",
    ];
    return $clubes;
}
$nomesDosClubes = paulistas();
echo "\nGrandes clubes paulistas que integram a primeira divisão do
Brasileirão:\n";
for ($i = 0; $i < sizeof($nomesDosClubes); $i++) {
    $ordem = $i + 1;
    echo "$ordem) $nomesDosClubes[$i] \n";
}

```

No exemplo anterior foi criada uma função de nome `paulistas`, que não recebeu argumentos em sua passagem. Dentro dessa função foi declarado um vetor com o nome (`$clubes`) e atribuído a esse vetor o nome de alguns times paulistas que integram o campeonato brasileiro. No final do bloco da função, foi retornada esta variável com o nome das equipes.

A variável (`$nomesDosClubes`) recebeu o resultado da chamada da função `paulistas` que retornou o nome dos clubes. Trata-se de um vetor local com os dados retornados.

Por fim, usou-se o comando (`for`) para percorrer cada item do vetor (`$nomesDosClubes`) e apresentá-los um a um na tela.

Agora vamos a outro exemplo, usando chave e valor na declaração.

Neste programa teremos um vetor associativo com dados de vários clientes, cada cliente com o seu código de cliente, seu nome e o valor do seu crédito.

Nome do arquivo: codigo24.php

```

<?php
function listarClientes(){

```

```

$clientes = [
    ["codigo" => 1, "nome" => "Ricardo Santos", "credito" =>
1000.0],
    ["codigo" => 2, "nome" => "Ana Santos", "credito" => 2000.0],
    ["codigo" => 3, "nome" => "Mariana Santos", "credito" =>
3000.0],
    ["codigo" => 4, "nome" => "Rodrigo Santos", "credito" =>
2000.0],
    ["codigo" => 5, "nome" => "Ulisses Santos", "credito" =>
1000.0],
];
return $clientes;
}
$lista = listarClientes();
foreach ($lista as $cliente) {
    echo "Cliente: \n";
    foreach ($cliente as $campo=>$valor) {
        echo "$campo = $valor \n";
    }
    echo "-----\n";
}

```

Resultado da execução:

Cliente:
 codigo = 1
 nome = Ricardo Santos
 credito = 1000

Cliente:
 codigo = 2
 nome = Ana Santos
 credito = 2000

Cliente:
 codigo = 3
 nome = Mariana Santos
 credito = 3000

Cliente:

```
codigo = 4
nome = Rodrigo Santos
credito = 2000
```

```
Cliente:
codigo = 5
nome = Ulisses Santos
credito = 1000
```

Vamos criar uma variação na chamada da função. Suponhamos que queiramos apenas os dados do cliente cujo código seja igual a 4. Para isso, crie o próximo script,

Nome do arquivo: codigo25.php

```
<?php
function listarCliente($codProc){
    $clientes = [
        ["codigo" => 1, "nome" => "Ricardo Santos", "credito" =>
1000.0],
        ["codigo" => 2, "nome" => "Ana Santos", "credito" => 2000.0],
        ["codigo" => 3, "nome" => "Mariana Santos", "credito" =>
3000.0],
        ["codigo" => 4, "nome" => "Rodrigo Santos", "credito" =>
2000.0],
        ["codigo" => 5, "nome" => "Ulisses Santos", "credito" =>
1000.0],
    ];
    foreach ($clientes as $cliente) {
        foreach ($cliente as $campo => $valor) {
            if ($campo == "codigo" && $valor == $codProc) {
                return $cliente;
            }
        }
    }
}
$codProc = 1;
$cliente = listarCliente($codProc);
foreach ($cliente as $campo => $valor) {
    echo "$campo = $valor \n";
```

```
}
```

Resultado da execução:

```
codigo = 1
nome = Ricardo Santos
credito = 1000
```

Vamos a mais um exemplo. Usando vetor numérico. O Próximo programa recebe cinco números como argumentos pela linha de comando, coloca-os em um vetor e mostra-os.

Nome do arquivo: **codigo26.php**

```
<?php
$valores = [];
//Percorre os argumentos passados pela Linha de comando até cinco (1 a 5);
for ($i=1; $i <= 5; $i++) {
    $valores[] = $argv[$i];
}
//O vetor de valores tem suas posições de 0 a 4 preenchidas
//percorremos para exibir cada valor.
for ($i=0; $i < sizeof($valores); $i++) {
    echo "Valor: $valores[$i] \n";
}
```

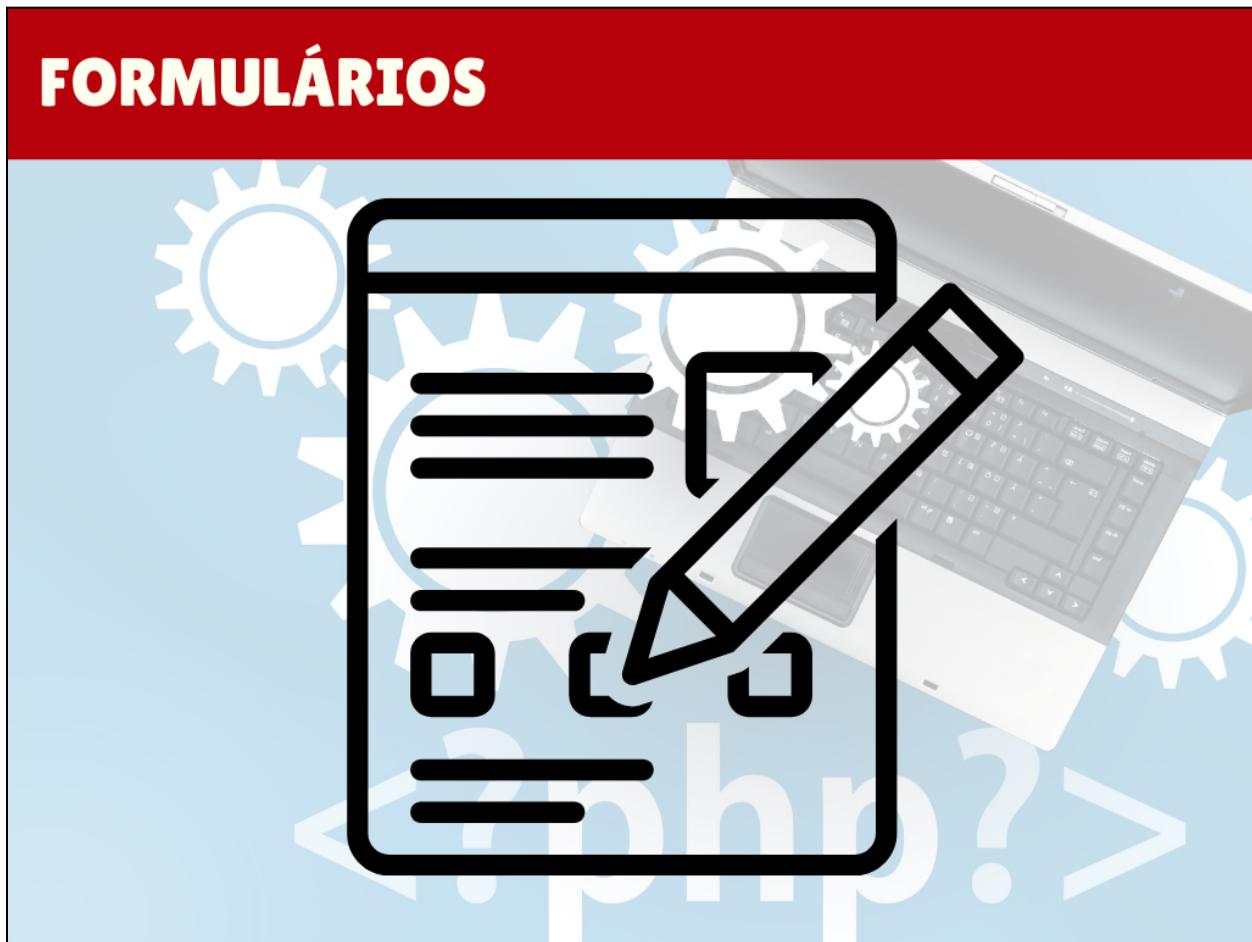
Exercícios de Fixação

1. Faça um Programa que receba um vetor de 5 números reais e mostre-os na ordem inversa.
2. Faça um Programa que receba 4 notas, mostre as notas e a média na tela.
3. Faça um Programa que receba um vetor de 5 caracteres, e diga quantas consoantes foram lidas. Imprima as consoantes.

-
4. Faça um Programa que receba 5 números inteiros e armazene-os num vetor. Armazene os números pares no vetor PAR e os números ÍMPARES no vetor impar. Imprima os três vetores.
 5. Faça um Programa que receba as quatro notas de 2 alunos, calcule e armazene num vetor a média de cada aluno.
 6. Faça um Programa que receba um vetor de 5 números inteiros, mostre a soma, a multiplicação e os números.
 7. Faça um Programa que receba a idade e a altura de 3 pessoas, armazene cada informação no seu respectivo vetor. Imprima a idade e a altura na ordem inversa a ordem lida.
 8. Faça um Programa que receba um vetor A com 5 números inteiros, calcule e mostre a soma dos quadrados dos elementos do vetor.
 9. Foram anotadas as idades de 5 alunos. Faça um Programa que determine quantos alunos têm mais de 13 anos.
 10. Faça um programa que tenha declarado a temperatura média de cada mês do ano e armazene-as uma lista (vetor associativo). Após isto, calcule a média anual das temperaturas e mostre todas as temperaturas acima da média anual, e em que mês elas ocorreram (mostrar o mês por extenso: 1 – Janeiro, 2 – Fevereiro, . . .).

Formulários HTML

Um formulário HTML é usado para coletar a entrada do usuário. A entrada do usuário geralmente é enviada a um servidor para processamento.



O elemento HTML `<form>` é usado para criar um formulário HTML para entrada do usuário, trata-se de um contêiner para diferentes tipos de entrada, como: campos de texto, caixas de seleção, botões de opção, botões de envio, etc. Sintaxe:

```
<form>
.... elementos do formulário
</form>
```

Elementos de um formulário

Em HTML um formulário pode conter os seguintes elementos:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

O elemento <input>

Um dos elementos mais utilizados. O <**input**> pode ser exibido de várias maneiras, dependendo do atributo **type** definido.

```
<label for="nome">Nome completo: </label>
<input type="text" id="nome" name="nome">
```

Nome do arquivo: formulario01.php

```
<!DOCTYPE html>
<html>
<body>

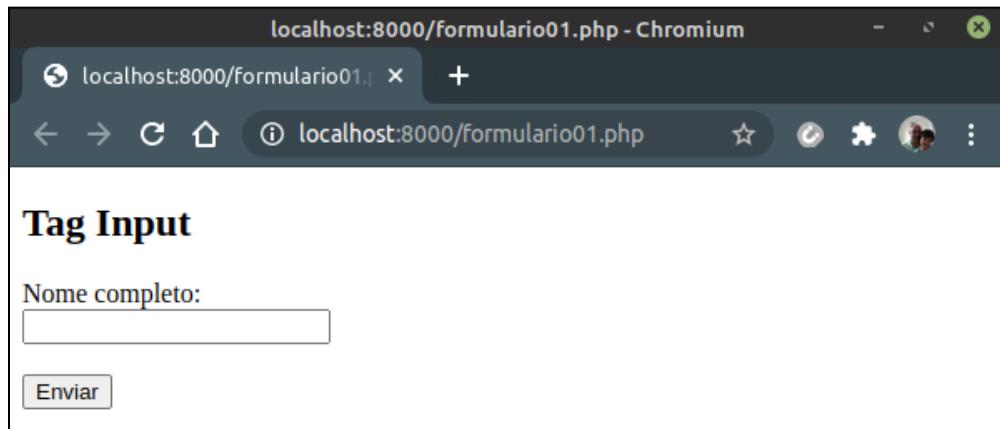
<h2>Tag Input</h2>

<form action="/action_page.php">
  <label for="nome">Nome completo:</label><br>
  <input type="text" id="nome" name="nome"><br><br>
  <input type="submit" value="Enviar">
</form>
```

```
</body>  
</html>
```

Exibindo no navegador

```
$ php -S localhost:8000/formulario01.php
```



No exemplo acima ainda temos o elemento `label`, que definimos em seguida.

O elemento <label>

Este elemento define um rótulo para vários elementos do formulário. O `<label>` é útil para usuários de leitores de tela, porque o leitor de tela lerá o rótulo em voz alta quando o usuário focar no elemento de entrada. Também ajuda os usuários que têm dificuldade em clicar em regiões muito pequenas (como botões de rádio ou caixas de seleção) - porque quando o usuário clica no texto dentro do `<label>`, ele alterna o botão de rádio / caixa de seleção.

O atributo `(for)` da tag `<label>` deve ser igual ao id do `<input>` para uni-los.

O elemento <select>

O elemento `<select>` define uma lista do tipo drop-down. Veja o exemplo.

Nome do arquivo: formulario02.php.

```
<!DOCTYPE html>  
<html>
```

```
<body>

<h2>A Tag Select</h2>

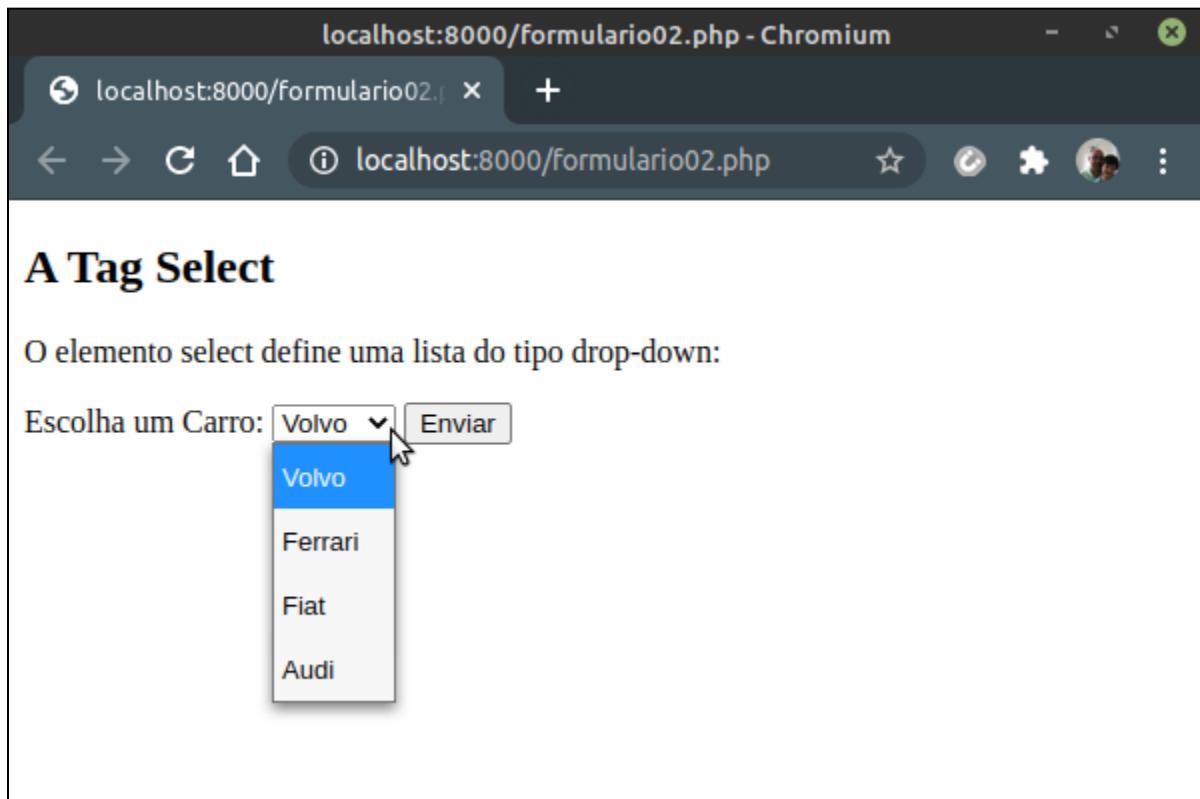
<p>O elemento select define uma lista do tipo drop-down:</p>

<form action="/action_page.php">
  <label for="cars">Escolha um Carro:</label>
  <select id="cars" name="cars">
    <option value="volvo">Volvo</option>
    <option value="ferrari">Ferrari</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <input type="submit" value="Enviar">
</form>

</body>
</html>
```

Exibindo no navegador

```
$ php -S localhost:8000/formulario02.php
```



Permitir várias seleções:

Use o atributo `multiple` permite que o usuário selecione mais de um valor, veja a alteração no código:

```
<!DOCTYPE html>
<html>
<body>

<h2>A Tag Select</h2>

<p>O elemento select define uma lista do tipo drop-down:</p>

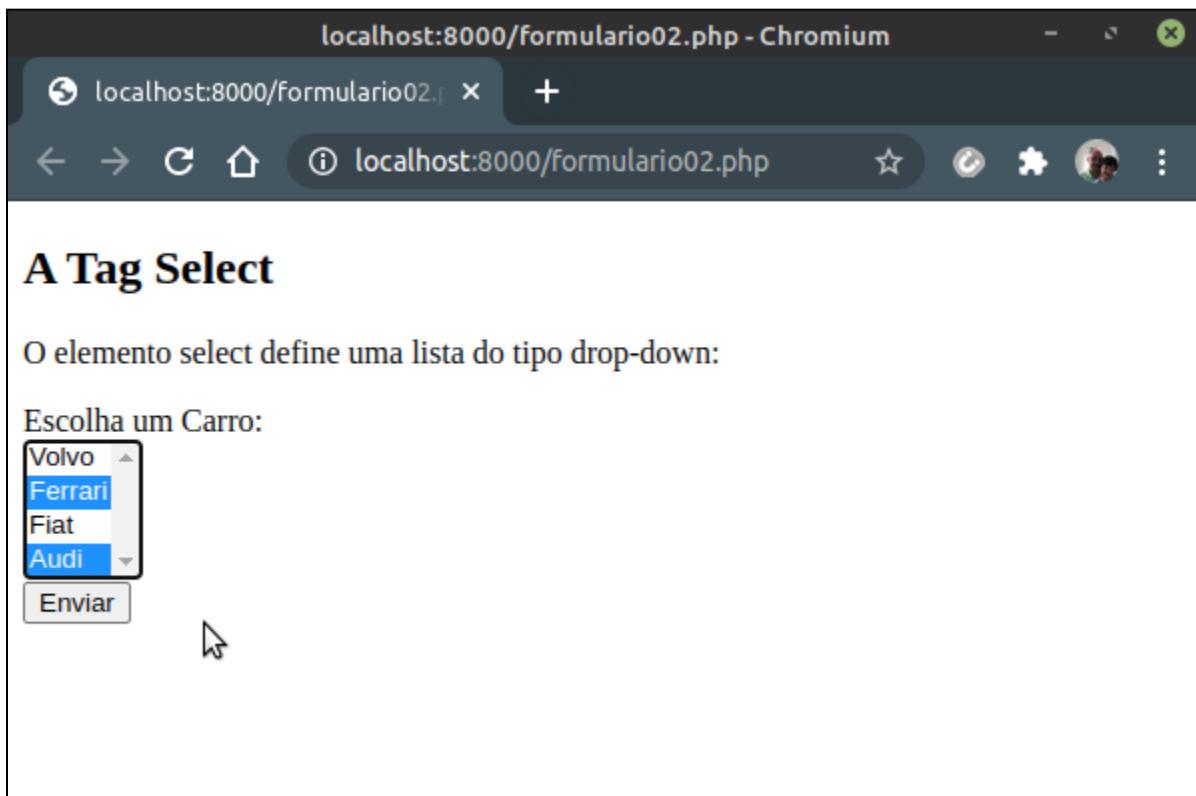
<form action="/action_page.php">
  <label for="cars">Escolha um Carro:</label>
  <br>
  <select id="cars" name="cars" multiple>
    <option value="volvo">Volvo</option>
    <option value="ferrari">Ferrari</option>
```

```
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
<br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Exibindo no navegador

```
$ php -S localhost:8000/formulario02.php
```



O elemento <textarea>

O elemento select define um campo de texto que pode conter várias linhas.

Nome do arquivo: formulario03.php.

```
<!DOCTYPE html>
<html>
<body>
    <h2>Textarea</h2>
    <p>A Tag textarea define um campo texto de várias linhas.</p>

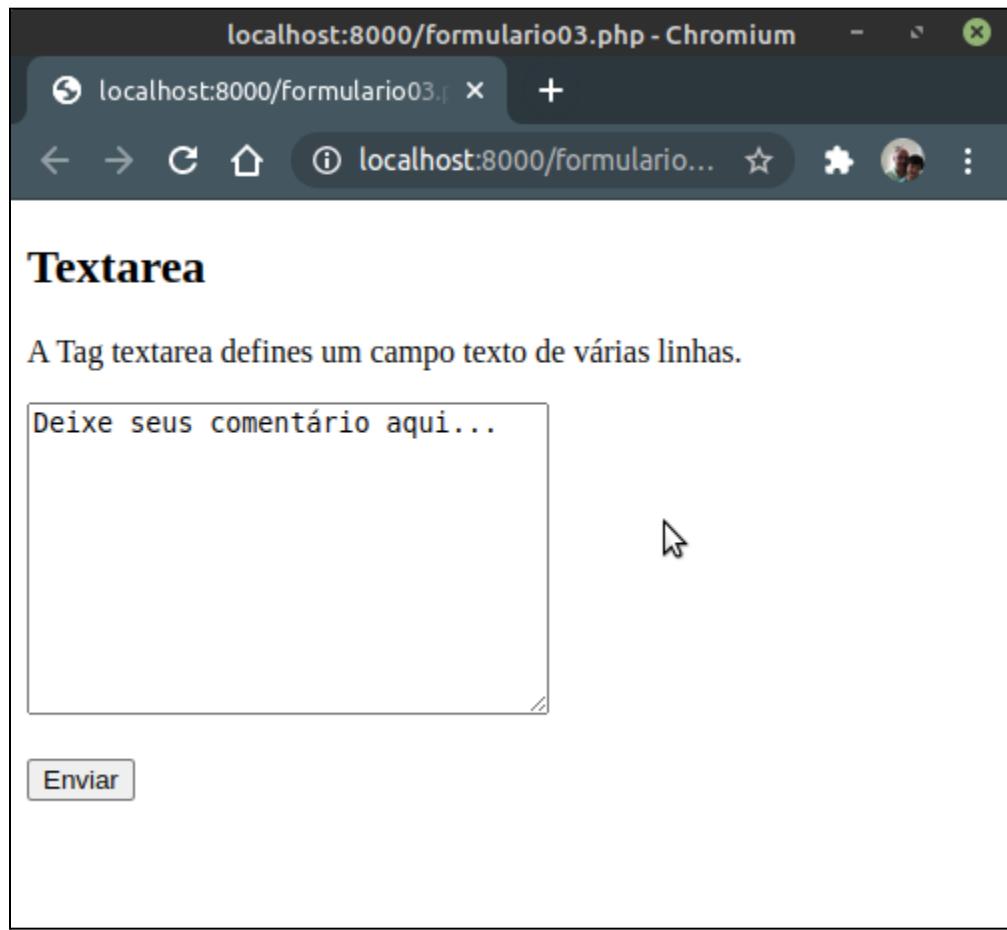
    <form action="/action_page.php">
        <textarea name="message" rows="10" cols="30">
            Deixe seus comentário aqui...
        </textarea>
        <br><br>
        <input type="submit" value="Enviar">
    </form>

</body>

</html>
```

Exibindo no navegador

```
$ php -S localhost:8000/formulario03.php
```



Enviando dados para um arquivo php

Vamos criar um formulário que objetiva buscar impressões do visitante sobre um site e seus serviços. Após o usuário preencher os campos as informações serão enviadas para um arquivo com script php que tratará o recebimento dos dados.

Para isso, crie um novo arquivo.

Nome do arquivo: formulario04.php

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sua opinião é importante para nós</title>
```

```

</head>
<body>
    <h2>Sua opinião</h2>
    <form action="recebe.php" method="post">
        <h3>O que você achou dos nossos serviços? </h3>
        <p>
            <input type="radio" name="opcao" value="Muito Bom" checked>Muito
            Bom
            <input type="radio" name="opcao" value="Bom">Bom
            <input type="radio" name="opcao" value="Regular">Regular
            <input type="radio" name="opcao" value="Ruim">Ruim
        </p>
        <h3>Qual de nossas áreas utiliza com frequencia:</h3>
        <select size="1" name="tipo">
            <option value="sauna">Sauna</option>
            <option value="futebol">Futebol</option>
            <option value="bocha">Bocha</option>
            <option value="outros">Outros</option>
        </select>
        <h3>Digite seus comentários abaixo:</h3>
        <textarea name="comentarios" rows="5" cols="50"></textarea>
        <input type="submit" value="Enviar!" name="enviar">
    </form>
</body>
</html>

```

Note a linha:

```
<form action="recebe.php" method="post">
```

Nesta linha, indicamos ao php que o arquivo destino dos dados é o arquivo de nome `recebe.php`, este arquivo deve ser processado no servidor e efetuar o processamento dos dados enviados, como por exemplo, a gravação para o banco de dados e retornar uma renderização em html ao cliente informando sobre a ação. No nosso caso, optamos por apresentar os dados enviados na tela do cliente no formato HTML.

Abaixo código do arquivo `recebe.php`

```

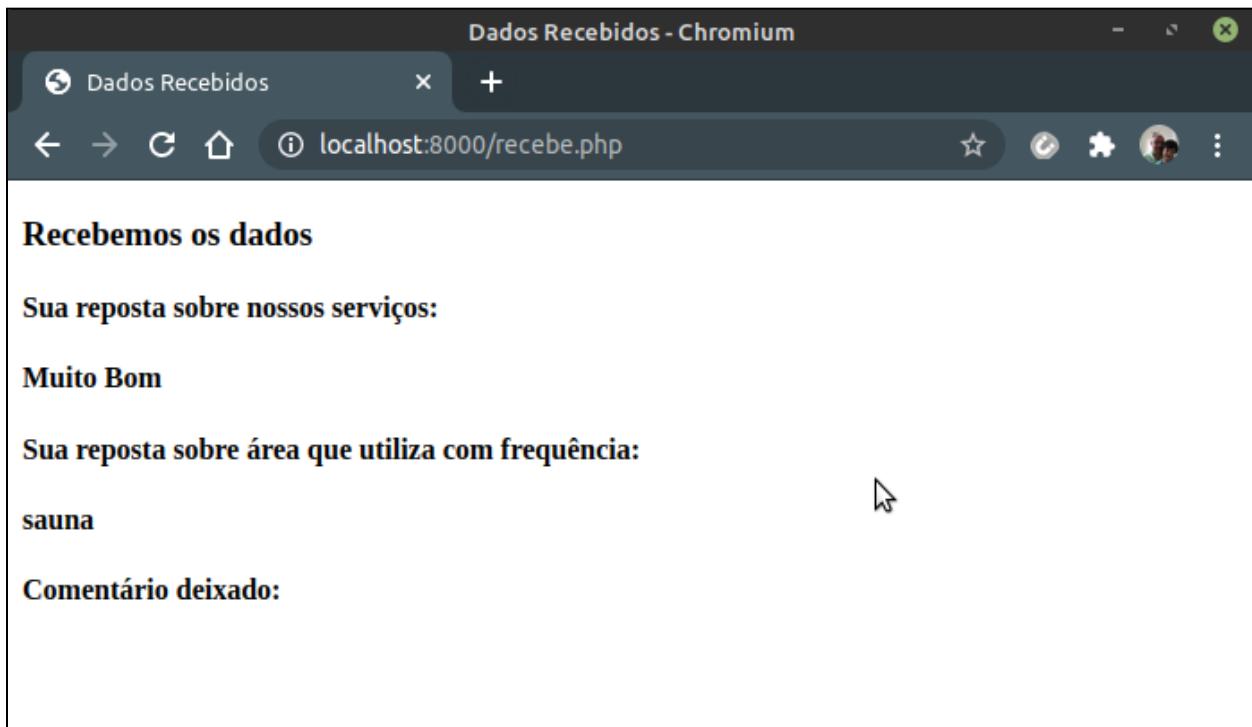
<?php
//Usando variáveis para receber os dados pelo método POST
$opcao = $_POST["opcao"];

```

```
$tipo = $_POST["tipo"];
$comentarios = $_POST["comentarios"];
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Dados Recebidos</title>
</head>
<body>
    <h3>Recebemos os dados</h3>
    <h4>Sua resposta sobre nossos serviços:</h4>
    <p><?=$opcao?></p>
    <h4>Sua resposta sobre área que utiliza com frequência:</h4>
    <p><?=$tipo?></p>
    <h4>Comentário deixado:</h4>
    <p><?=$comentarios?></p>
</body>
</html>
```

Abra o formulário “**formulario04.php**” no navegador, preencha os dados, e ao final clique no botão enviar. Os dados enviados são processados pelo arquivo “recebe.php” que renderiza o resultado em HTML na sua tela.

Resultado:



Uma boa prática é validar os dados enviados ao servidor antes de efetuar os demais processamentos, como gravação dos dados em um banco de dados. A ideia é abortar ou parar o processamento enviando uma mensagem ao usuário que determinado campo não foi devidamente preenchido.

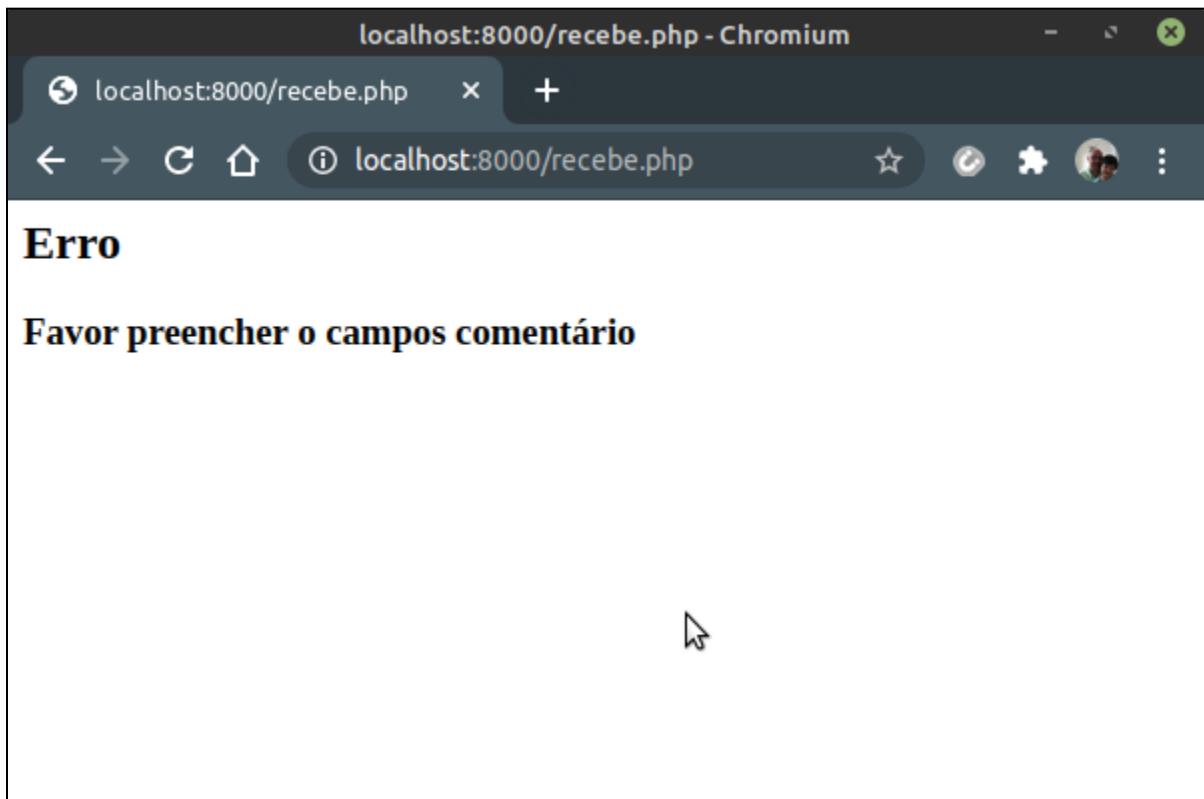
Altere o exemplo do arquivo `recebe.php` para o código disposto a seguir:

```
<?php
//Usando variáveis para receber os dados pelo método POST
$opcao = $_POST["opcao"];
$tipo = $_POST["tipo"];

if(empty($_POST["comentarios"])){
    echo "<h2>Erro</h2>";
    echo "<h3>Favor preencher o campos comentário</h3>";
    return;
} else{
    $comentarios = $_POST["comentarios"];
}
?>
```

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Dados Recebidos</title>
</head>
<body>
    <h3>Recebemos os dados</h3>
    <h4>Sua resposta sobre nossos serviços:<h4>
    <p><?=$opcao?></p>
    <h4>Sua resposta sobre área que utiliza com frequência:<h4>
    <p><?=$tipo?></p>
    <h4>Comentário deixado:</h4>
    <p><?=$comentarios?></p>
</body>
</html>
```

Execute o formulário e deixe o campo comentário sem preencher. O resultado da execução deve ser este:



Recomenda-se que se pratique os demais elementos. Uma boa fonte para pesquisa é o site da Organização Mozilla.

<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/form>

The screenshot shows the MDN Web Docs page for the `<form>` element. The page includes a navigation bar with links to Technologies, References & Guides, Feedback, and a search bar. Below the navigation, there's a breadcrumb trail: Tecnologia Web para desenvolvedores > HTML: Linguagem de Marcação de Hipertexto > Elementos HTML > `<form>`. There are also links to Change language and View in English.

Table of contents

- Resumo
- Atributos
- Exemplos
- Especificações
- Compatibilidade do navegador
- Veja também

<form>

Resumo

O elemento HTML `<form>` representa uma seção de um documento que contém controles interativos que permitem ao usuário submeter informação a um determinado servidor web.

É possível utilizar as pseudo-classes de CSS `:valid` e `:invalid` para aplicar estilo a um elemento `<form>`.

Related Topics

- ▶ HTML Elements

Content categories Flow content, palpable content.
Permitted content Flow content, but with no contained `<form>` elements.
Tag omission None, both the starting and ending tag are mandatory.
Permitted parent elements Any element that accepts flow content.
DOM interface [HTMLFormElement](#) (en-US)

Nota:

Existem dois métodos para a transmissão / envio de dados por um formulário, são eles o POST e o GET. Vamos conhecê-los melhor.

Método GET

O método GET, é o método padrão para envio de dados. Utiliza os caractere “?” que representa o início de uma cadeia de variáveis e o símbolo “&” que identifica o início de uma nova variável.

Inconveniente:

Limite de caracteres que pode ser enviado desta forma: cerca de 2000.

Vantagem:

Esse método pode ser usado para a passagem de parâmetro por meio de links.

Método POST

Envia os dados do formulário através do corpo da mensagem encaminhada ao servidor.

Vantagem:

- Não há limites de tamanho de dados a ser enviado.
- Pode-se enviar outros tipos de dados como imagens ou arquivos. Esses tipos não podem ser enviados via GET.

É recomendado para formulários que possuem grande número de informações, ou para o envio dos tipos acima (imagens ou arquivos).

Exercícios de fixação

1. Faça um programa para o cálculo de uma folha de pagamento, sabendo que os descontos são do Imposto de Renda, que depende do salário bruto (conforme tabela abaixo) e 3% para o Sindicato e que o FGTS corresponde a 11% do Salário Bruto, mas não é descontado (é a empresa que deposita). O Salário Líquido corresponde ao Salário Bruto menos os descontos. Você deve criar um formulário

em HTML para que o usuário informe o valor da sua hora e a quantidade de horas trabalhadas no mês. Calcule o valor do Salário Bruto usando a fórmula: Salário Bruto = valor da hora * quantidade de horas. E em seguida calcule o desconto do IR de acordo com a tabela.

Faixa	Desconto IR
Salário Bruto até 900 (inclusive)	isento
-Salário Bruto até 1500 (inclusive)	desconto de 5%
Salário Bruto até 2500 (inclusive)	desconto de 10%
Salário Bruto acima de 2500	desconto de 20%

Imprima na tela as informações, dispostas conforme o exemplo abaixo.

No exemplo o valor da hora é 5 e a quantidade de hora é 220.

Salário Bruto:	(5 * 220)	:	R\$ 1100,00
(-) IR (5%)		:	R\$ 55,00
(-) INSS (10%)		:	R\$ 110,00
FGTS (11%)		:	R\$ 121,00
Total de descontos		:	R\$ 165,00
Salário Liquido		:	R\$ 935,00

Você deve enviar os dados do formulário criado, para um arquivo em PHP que recebe e trata os todos dados, e ao final, exibe as informações como mostradas no exemplo.

- Crie um formulário para entrada de duas notas de um aluno. Depois faça um programa que receba e trate as duas notas parciais obtidas por um aluno numa disciplina ao longo de um semestre, e calcule a sua média. Ao final, mostre como resultado a média e o conceito do aluno. A atribuição de conceitos obedece à tabela abaixo:

Média de Aproveitamento	Conceito
Entre 9.0 e 10.0	A
Entre 7.5 e 9.0	B
Entre 6.0 e 7.5	C
Entre 4.0 e 6.0	D
Entre 4.0 e zero	E

Sessions

Uma sessão é uma forma de armazenar informações (em variáveis) a serem usadas em várias páginas. Ao contrário de um cookie, a informação não é armazenada no computador do usuário.

O que é uma sessão PHP?

Ao trabalhar com um aplicativo, você o abre, faz algumas alterações e, em seguida, fecha-o. Isso é muito parecido com uma sessão. O computador sabe quem você é. Ele sabe quando você inicia e quando você termina o aplicativo. Mas na internet há um problema: o servidor web não sabe quem você é ou o que você faz, porque o endereço HTTP não mantém o estado.

As variáveis de sessão resolvem esse problema armazenando informações do usuário para serem usadas em várias páginas (por exemplo, nome de usuário, suas preferências, etc.). Por padrão, as variáveis de sessão duram até que o usuário feche o navegador.



Iniciar uma sessão PHP

Uma sessão é iniciada com a função `session_start()`.

Variáveis de sessão são definidas com a variável global PHP: `$_SESSION`.

Agora, vamos criar uma nova página. Nesta página, iniciamos uma nova sessão PHP e definimos algumas variáveis de sessão.

Nome do arquivo: codigo27.php

```
<?php
// Iniciando a sessão
session_start();
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exemplo com sessões</title>
</head>
<body>
<?php
    // Definindo as variáveis de seção
    $_SESSION["fav-color"] = "verde";
    $_SESSION["fav-animal"] = "gato";
    echo "Variáveis de sessão foram criadas. <br>";
?>
<a href="codigo28.php">Ver Página</a>
</body>
</html>
```

Obter valores de variáveis de sessão PHP

Em seguida, criamos outra página. A partir desta página, acessamos as informações da sessão que definimos na primeira página ("codigo27..php").

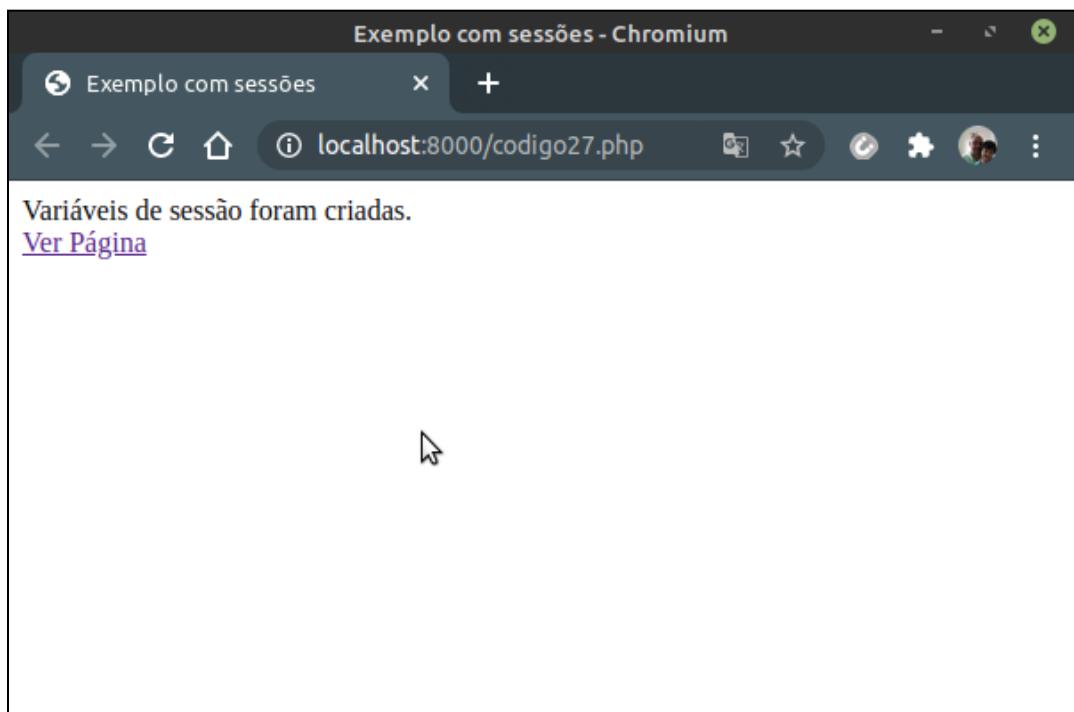
Observe que as variáveis de sessão não são passadas individualmente para cada nova página, em vez disso, são recuperadas da sessão que abrimos no início de cada página (`session_start()`).

Observe também que todos os valores das variáveis de sessão são armazenados na variável global `$_SESSION`:

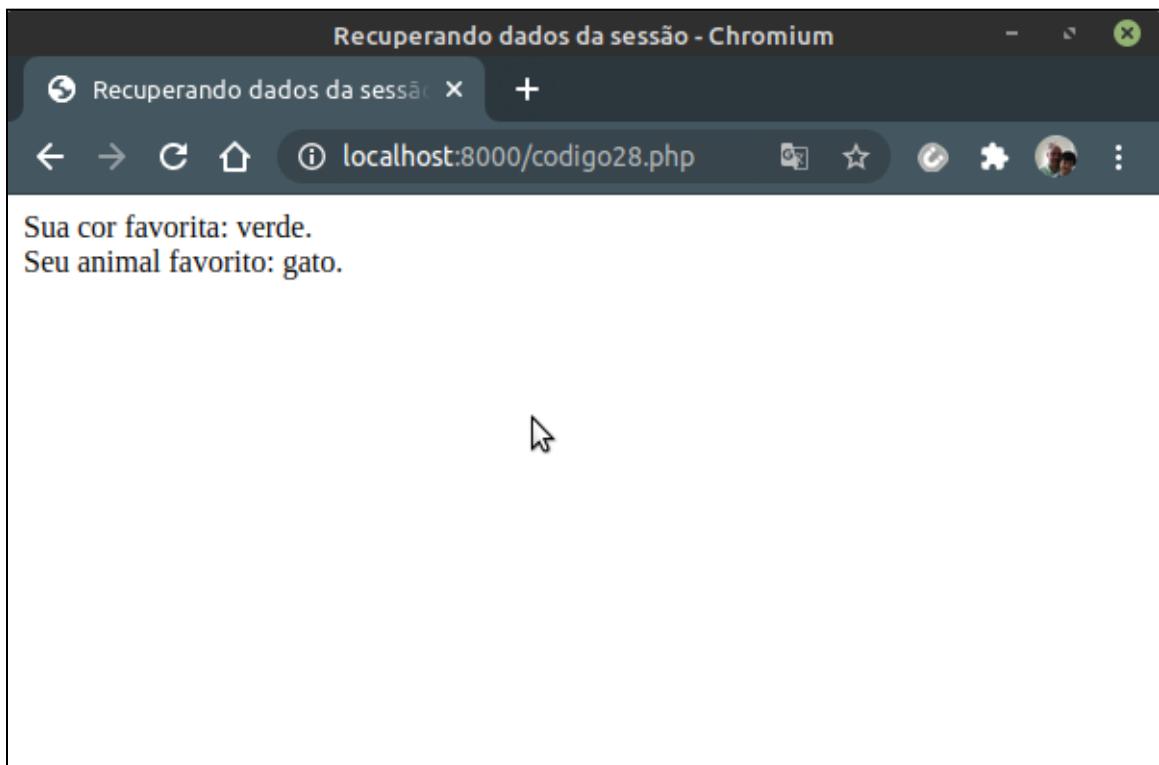
```
<?php
    session_start();
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Recuperando dados da sessão</title>
</head>
<body>
<?php
    // Imprimir as variáveis de sessão que foram definidas
    anteriormente
    echo "Sua cor favorita: " . $_SESSION["fav-color"] . ".<br>";
    echo "Seu animal favorito: " . $_SESSION["fav-animal"] . ".";
?>
</body>
</html>
```

Execute a primeira página:

```
$ php -S localhost:8000/codigo27.php
```



Clicando no link, teremos:



Modificando uma variável de sessão

Para mudar uma variável de sessão, basta sobrescrever o seu valor.

```
<?php
// Iniciando a sessão
session_start();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exemplo com sessões</title>
</head>
<body>
<?php
    // Alterando as variáveis de sessão
    $_SESSION["fav-animal"] = "cão";
    echo "Uma variável de sessão foi alterada. <br>";
?>
<a href="codigo28.php">Ver Página</a>
</body>
</html>
```

Destruindo uma sessão PHP

Para remover todas as variáveis globais de sessão e destruir a sessão, devemos invocar as funções `session_unset()` e `session_destroy()`. Crie uma nova página.

Nome do arquivo: codigo29.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html lang="en">
```

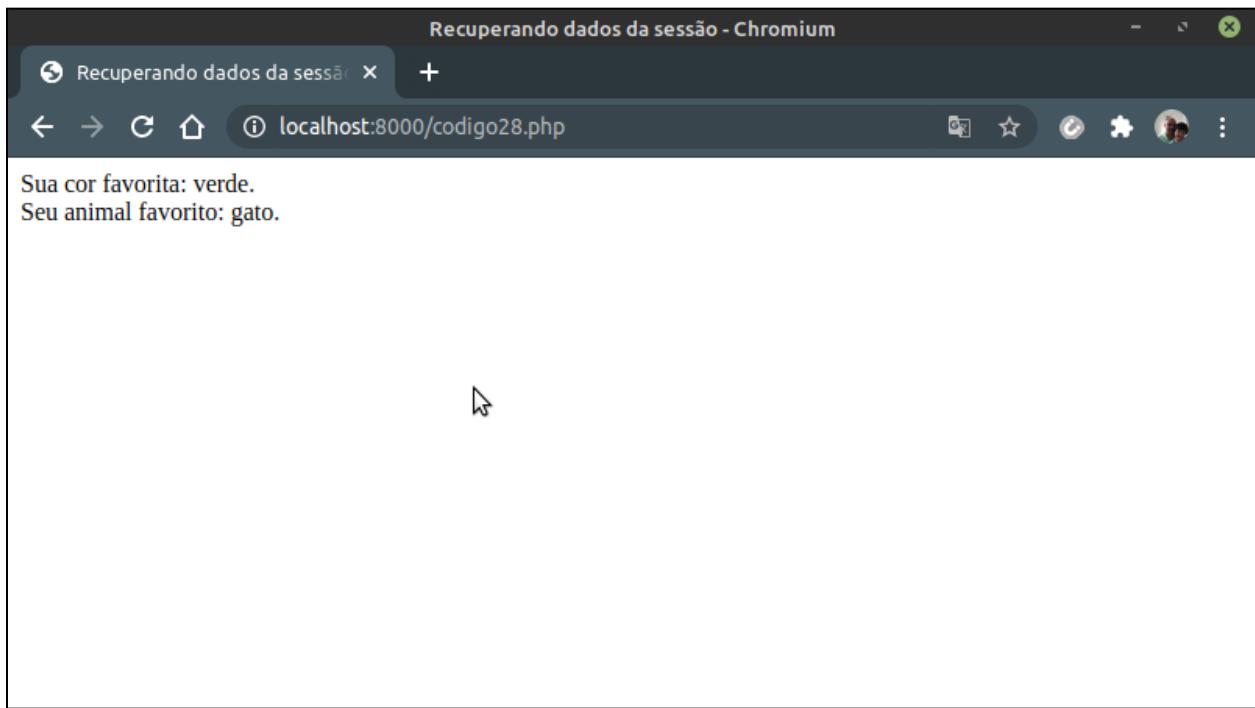
```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Destruindo uma sessão</title>
</head>

<body>
    <?php
        // remove todas as variáveis de sessão
        session_unset();

        // destrói a sessão
        session_destroy();
        echo "Sessão destruída. <br>";
    ?>
    <a href="codigo28.php">Verificar variáveis novamente</a>
</body>

</html>
```

Ao clicar sobre o link, veremos que os valores não serão mais apresentados, pois a sessão foi destruída.



Exercícios de Fixação

1. Seu desafio é criar um formulário para entrada de dados do usuário, como o apresentado a seguir:

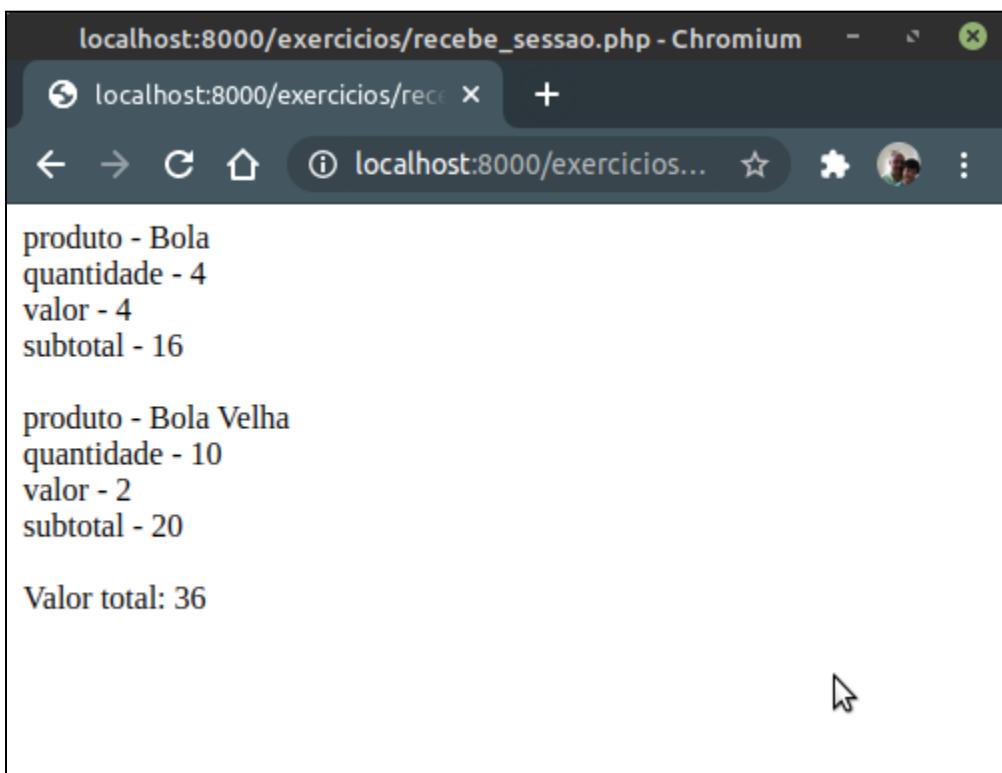
Dados do Compra

Nome do Produto:

Quantidade comprada:

Valor do produto:

Nele temos três elementos <input> que representam a entrada de dados de um nome de produto, a sua quantidade comprada e o valor unitário do programa. Este arquivo deve iniciar uma sessão apenas. Ao clicar no botão “enviar”, outro arquivo deve receber as informações e exibi-las na tela, até aqui nada demais, já vimos como fazer isto na aula de formulários. O desafio é guardar em uma sessão todos os produtos (uma lista), de maneiras que, se o usuário voltar na tela principal e cadastrar mais produtos, eles devem ser incluídos na lista. Ao final, você deve apresentar - além da lista do que foi comprado, o valor total da compra. Como mostra a tela a seguir:



O código deste desafio está disponível no repositório Github do curso, no final deste material.

PHP com MySQL



Antes de iniciarmos o gerenciamento de dados utilizando o PHP com o MySQL, apresentamos a seguir uma visão geral sobre o MySQL. Estas informações estão disponíveis no site do fabricante.

- MySQL, é o sistema de gerenciamento de banco de dados SQL Open Source mais popular do mercado, é desenvolvido, distribuído e tem suporte Oracle.
- O web site do MySQL (<http://www.mysql.com/>) fornece informações mais recentes sobre o programa MySQL e a Oracle.
- O MySQL é um sistema de gerenciamento de bancos de dados.Um banco de dados é uma coleção de dados estruturados. Ele pode ser qualquer coisa, desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede corporativa. Para adicionar, acessar, e processar dados

armazenados em um banco de dados de um computador, você necessita de um sistema de gerenciamento de bancos de dados como o Servidor MySQL.

- O MySQL é um sistema de gerenciamento de bancos de dados relacional. Um banco de dados relacional armazena dados em tabelas separadas em vez de colocar todos os dados num só local. A parte SQL do “MySQL” atende pela “Structured Query Language - Linguagem Estruturada de Consultas”. SQL é a linguagem padrão mais comum usada para acessar banco de dados e é definida pelo padrão ANSI/ISO SQL. (O padrão SQL está evoluindo desde 1986 e existem diversas versões).
- O MySQL é um software Open Source. Open Source significa que é possível para qualquer um usar e modificar o programa.

No curso técnico de desenvolvimento para sistemas ou de informática para internet há dois módulos de disciplinas que tratam sobre banco de dados¹.

Instalando o SGBD MySQL no Windows



Como nosso objetivo é o uso do PHP e não a preparação de um SGBD (Sistema Gerenciador de Banco de Dados), vamos instalar um software que facilita a instalação do MySQL, na verdade é um conjunto de soluções preparada pela Organização Apache Friends. O programa é o XAMPP.

O objetivo do XAMPP é construir uma distribuição fácil de instalar para desenvolvedores entrem no mundo do Apache. Para torná-lo conveniente para os desenvolvedores, o XAMPP é configurado com todos os recursos ativados.

A licença do XAMPP

XAMPP é uma compilação de softwares livres (comparável a uma distribuição Linux), é gratuito e é livre para ser copiado sob os termos da GNU General Public Licence. Mas é apenas a compilação do XAMPP que é publicada sob a licença GPL.

Para baixar o XAMPP você primeiramente deve acessar o link:

https://www.apachefriends.org/pt_br/download.html

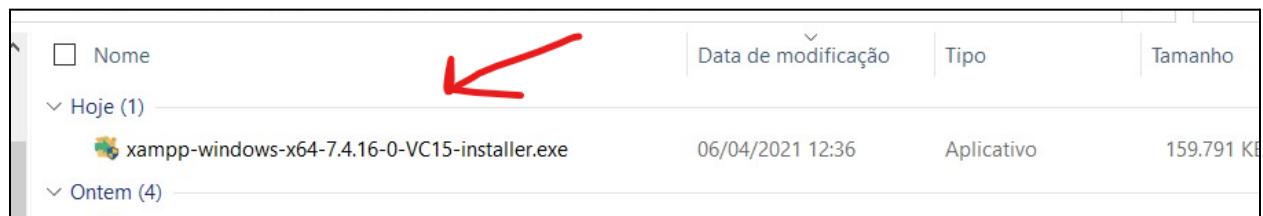
¹ São as disciplinas Modelagem e Desenvolvimento de Banco de Dados, Banco de Dados I e II que fazem parte das grades dos cursos técnicos em desenvolvimento de sistemas ou de informática para internet nas Escolas Técnicas Estaduais (Etecs) do Centro Paula Souza, autarquia do estado de São Paulo.

Escolha a versão 7.4.16

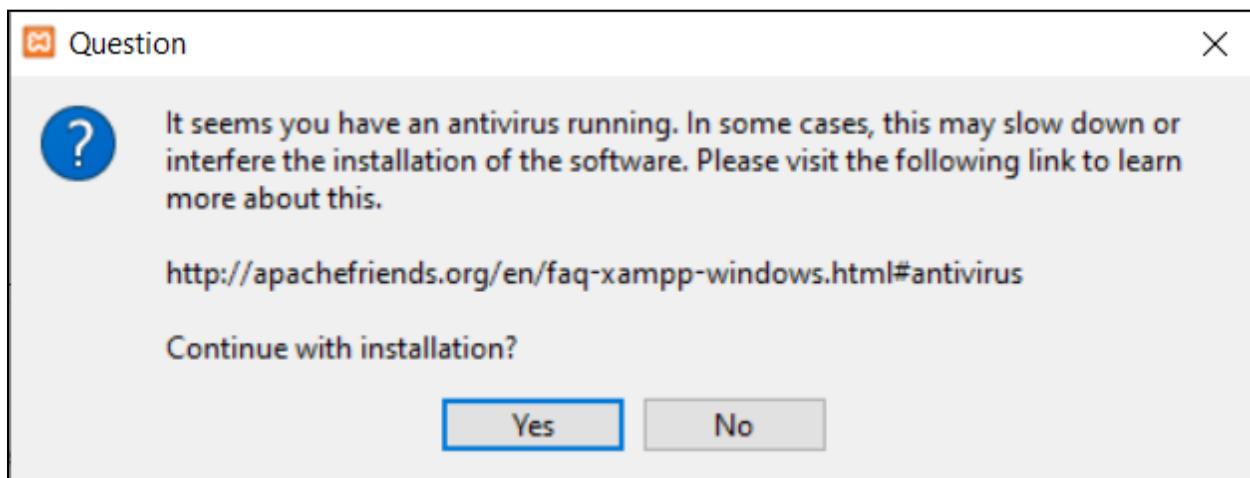
Versão	Soma de verificação	Tamanho
7.3.27 / PHP 7.3.27	O que está incluído? md5 sha1	Baixar (64 bit) 154 Mb
<u>7.4.16 / PHP 7.4.16</u>	O que está incluído? md5 sha1	<u>Baixar (64 bit)</u> <u>156 Mb</u>
8.0.3 / PHP 8.0.3	O que está incluído? md5 sha1	Baixar (64 bit) 156 Mb

Requisitos Extensões Mais Downloads »

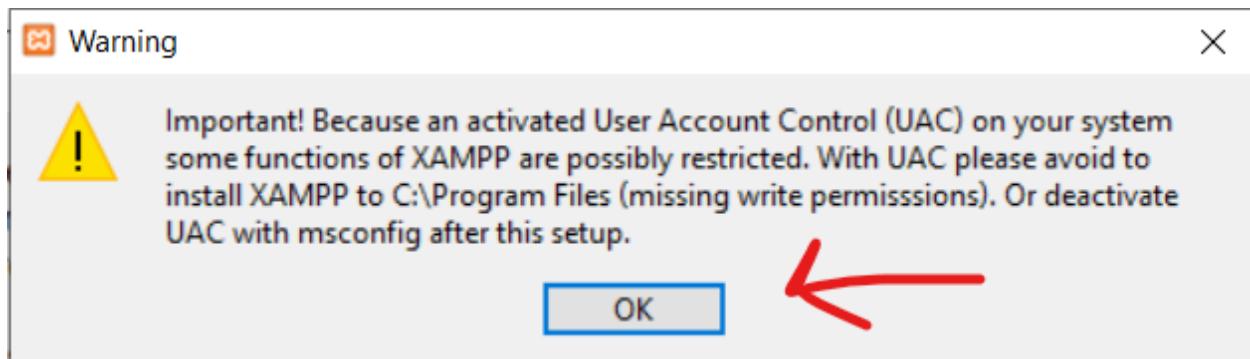
Execute o arquivo instalador do programa, como administrador do sistema, clique com botão direito do mouse para selecionar esta opção.



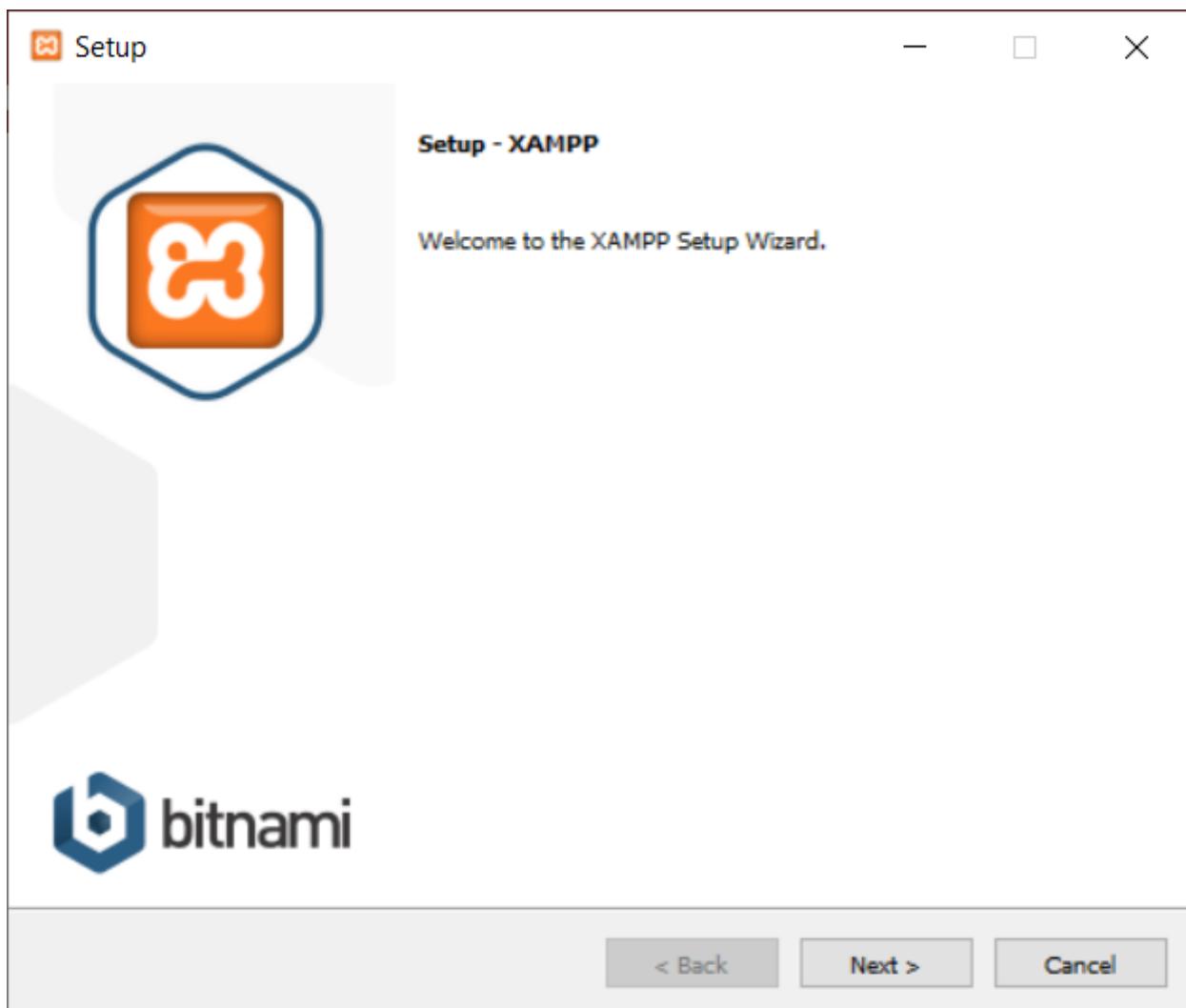
Ao ser questionado sobre a presença de um antivírus, é interessante pausar a execução do aplicativo para depois continuar a instalação clicando sobre o botão sim (yes).



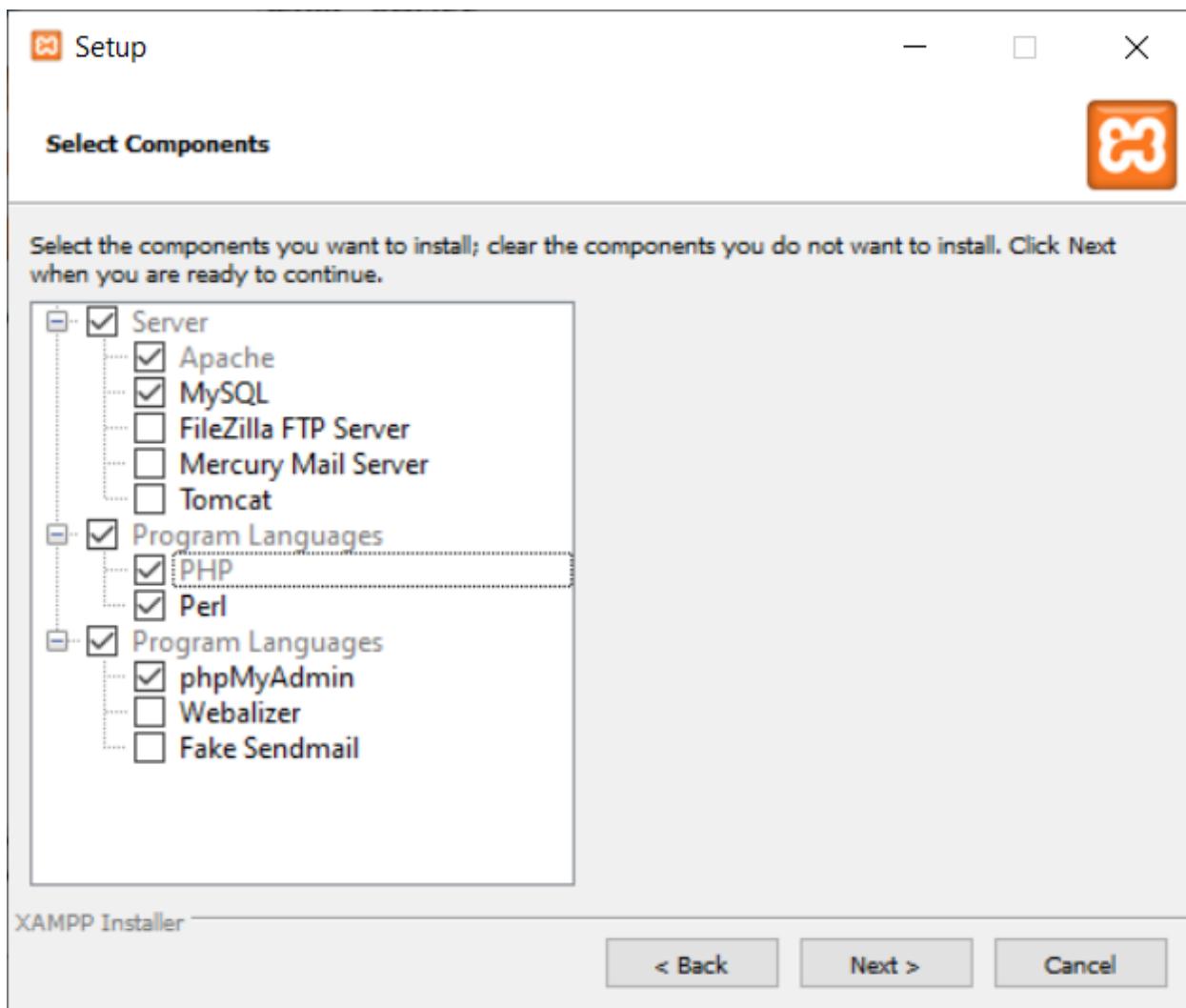
Se você iniciou como administrador é só confirmar na próxima tela.



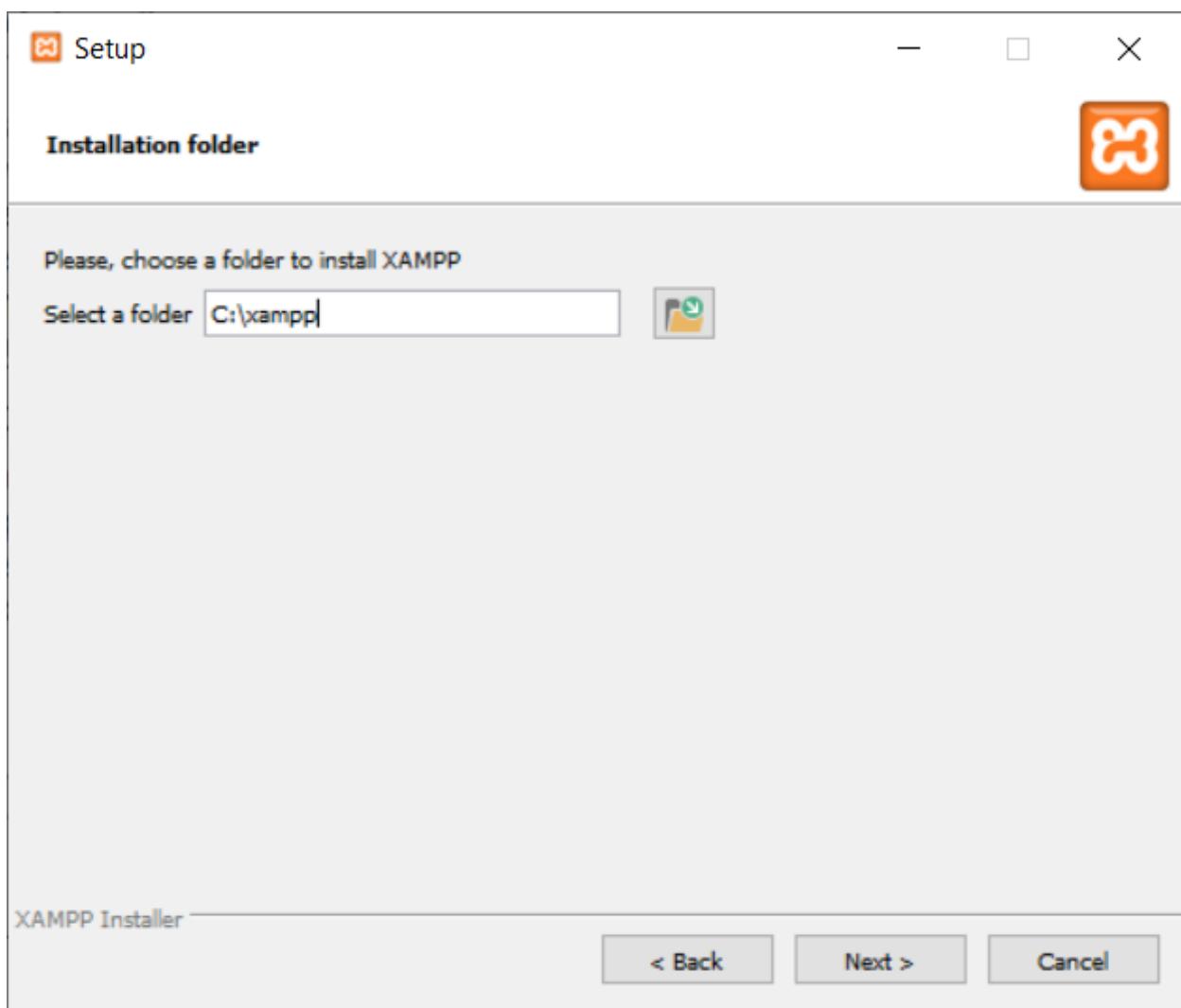
E depois em Next.



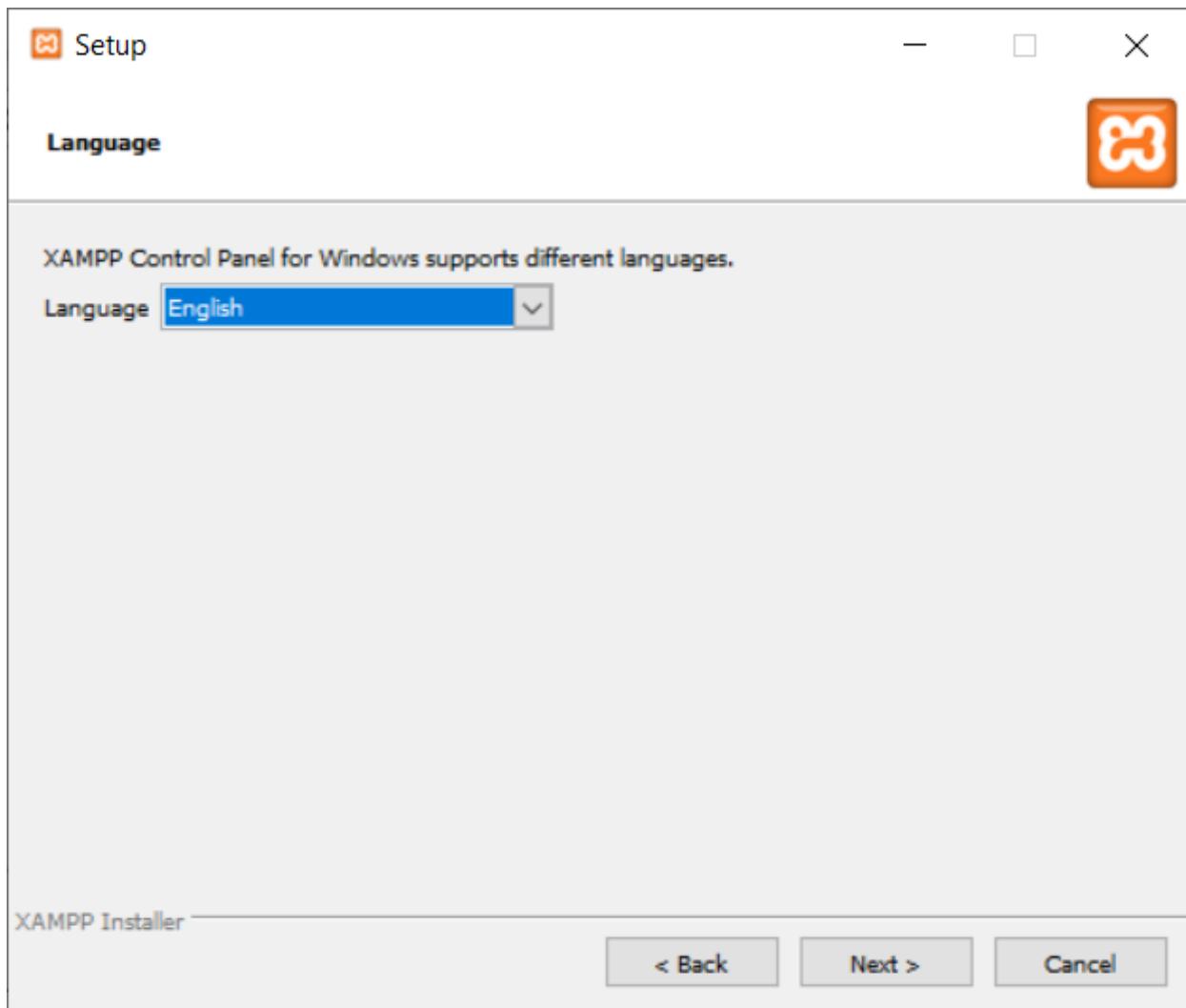
Deixe apenas marcadas as opções informadas na próxima tela.



Clique em Next para continuar. Confirme o diretório para instalação.

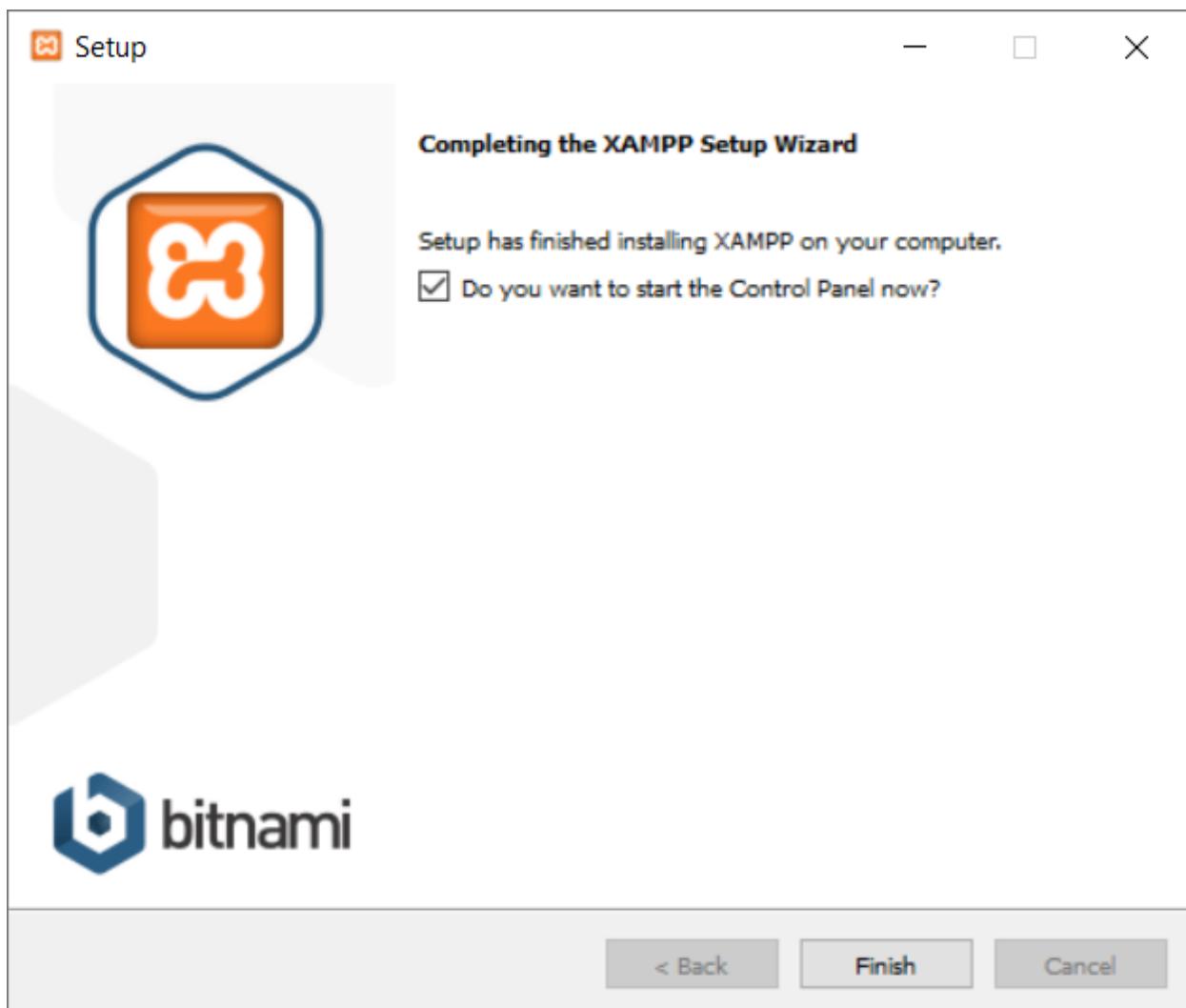


Clique em Next. Escolha o idioma na próxima tela.



Clique em Next, e novamente, até finalizar a instalação. Este processo pode ser um pouco demorado. Uma boa pausa para um café. :-)

Pronto.



Deixe marcada a opção para carregar o painel de controle neste momento. Clique em Finish.

O painel de controle deve aparecer na barra de tarefas, clique nele para ativá-lo. No nosso caso, como temos o PHP instalado, basta clicar sobre o botão Start do MySQL para iniciar o Gerenciador de Banco de Dados.

The screenshot shows the XAMPP Control Panel interface. At the top, it displays 'XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]'. Below the title, there's a table with columns: Service, Module, PID(s), Port(s), and Actions. The table lists the following services:

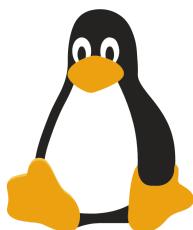
Service	Module	PID(s)	Port(s)	Actions
	Apache	7088 11016	80, 443	Stop Admin Config Logs
	MySQL	6176	3306	Stop Admin Config Logs
	FileZilla			Start Admin Config Logs
	Mercury			Start Admin Config Logs
	Tomcat			Start Admin Config Logs

On the right side of the panel, there are several icons with labels: Config, Netstat, Shell, Explorer, Services, Help, and Quit. Below the table, a log window displays the following entries:

```
15:43:58 [main] The Mercury module is disabled
15:43:58 [main] The Tomcat module is disabled
15:43:58 [main] Starting Check-Timer
15:43:58 [main] Control Panel Ready
15:44:01 [Apache] Attempting to start Apache app...
15:44:01 [Apache] Status change detected: running
15:44:08 [mysql] Attempting to start MySQL app...
15:44:09 [mysql] Status change detected: running
```

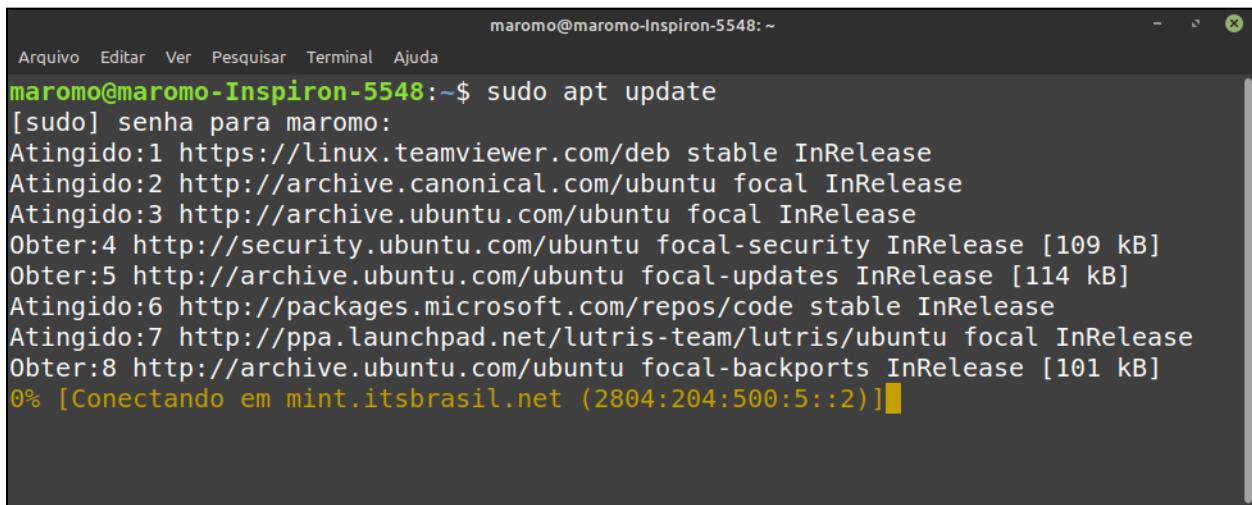
Pronto. O Sistema Gerenciador de Banco de Dados está instalado.

Instalando o SGBD MySQL no Linux



Abra o terminal e atualize o cache do repositório apt antes de instalar o MySQL:

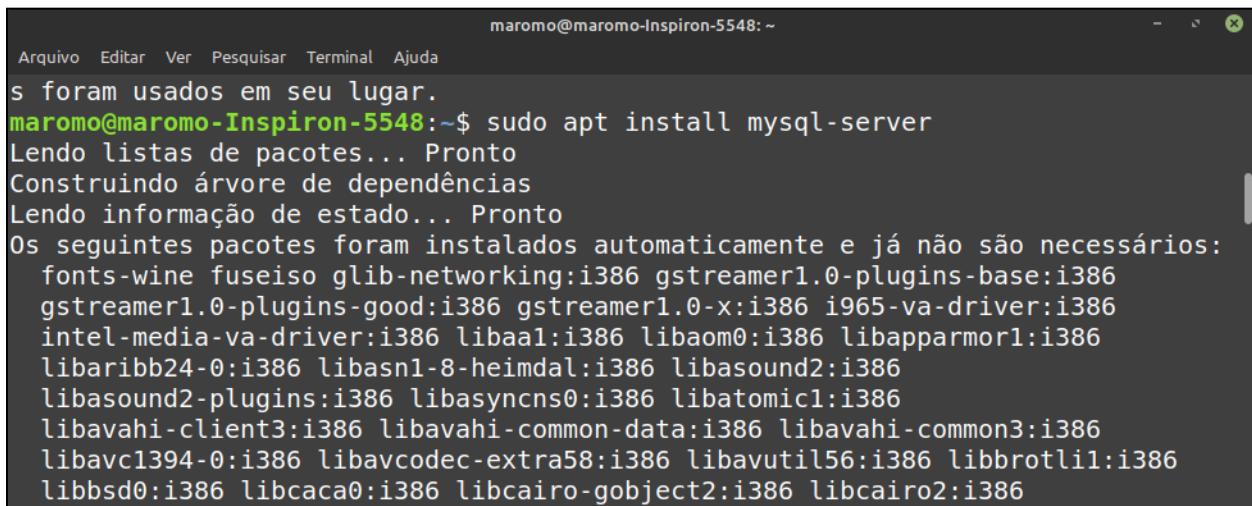
```
$ sudo apt update
```



```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
maromo@maromo-Inspiron-5548:~$ sudo apt update
[sudo] senha para maromo:
Atingido:1 https://linux.teamviewer.com/deb stable InRelease
Atingido:2 http://archive.canonical.com/ubuntu focal InRelease
Atingido:3 http://archive.ubuntu.com/ubuntu focal InRelease
Obter:4 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Obter:5 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Atingido:6 http://packages.microsoft.com/repos/code stable InRelease
Atingido:7 http://ppa.launchpad.net/lutris-team/lutris/ubuntu focal InRelease
Obter:8 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
0% [Conectando em mint.itsbrasil.net (2804:204:500:5::2)]
```

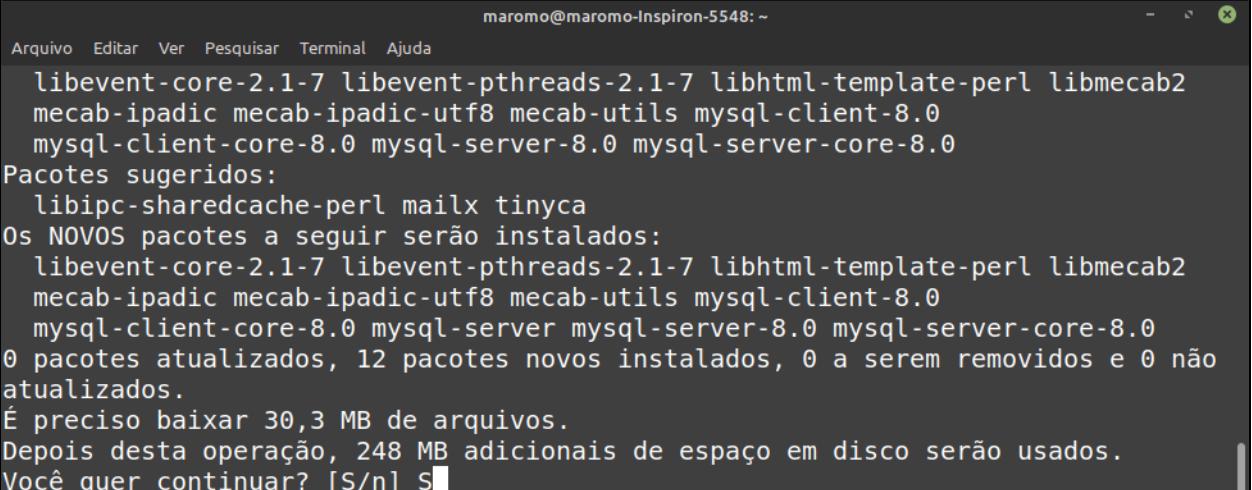
Em seguida, instale o servidor MySQL com o comando:

```
$ sudo apt install mysql-server
```



```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
s foram usados em seu lugar.
maromo@maromo-Inspiron-5548:~$ sudo apt install mysql-server
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  fonts-wine fuseiso glib-networking:i386 gstreamer1.0-plugins-base:i386
  gstreamer1.0-plugins-good:i386 gstreamer1.0-x:i386 i965-va-driver:i386
  intel-media-va-driver:i386 libaa1:i386 libao0:i386 libapparmor1:i386
  libaribb24-0:i386 libasn1-8-heimdal:i386 libasound2:i386
  libasound2-plugins:i386 libasyncns0:i386 libatomic1:i386
  libavahi-client3:i386 libavahi-common-data:i386 libavahi-common3:i386
  libavc1394-0:i386 libavcodec-extra58:i386 libavutil56:i386 libbrotli1:i386
  libbsd0:i386 libcaca0:i386 libcairo-gobject2:i386 libcairo2:i386
```

Tecle S para continuar:



A screenshot of a terminal window titled "maromo@maromo-Inspiron-5548: ~". The window shows the output of a package manager (likely apt) during the MySQL installation. It lists several packages being installed, including libevent-core, libevent-pthreads, libhtml-template-perl, libmecab2, mecab-ipadic, mecab-ipadic-utf8, mecab-utils, mysql-client, mysql-client-core, mysql-server, mysql-server-8.0, and mysql-server-core-8.0. It also shows suggested packages like libipc-sharedcache-perl, mailx, and tinyca. The message indicates 0 new packages to install, 0 updated packages, 12 new packages to install, 0 packages to remove, and 0 packages not upgraded. It asks if the user wants to continue with "S/n" and the user has typed "S".

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
libevent-core-2.1-7 libevent-pthreads-2.1-7 libhtml-template-perl libmecab2
mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0
mysql-client-core-8.0 mysql-server-8.0 mysql-server-core-8.0
Pacotes sugeridos:
  libipc-sharedcache-perl mailx tinyca
0s NOVOS pacotes a seguir serão instalados:
libevent-core-2.1-7 libevent-pthreads-2.1-7 libhtml-template-perl libmecab2
mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0
mysql-client-core-8.0 mysql-server mysql-server-8.0 mysql-server-core-8.0
0 pacotes atualizados, 12 pacotes novos instalados, 0 a serem removidos e 0 não
atualizados.
É preciso baixar 30,3 MB de arquivos.
Depois desta operação, 248 MB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] S
```

Assim que o servidor MySQL for instalado com sucesso, verifique a versão instalada e verifique a instalação usando o comando:

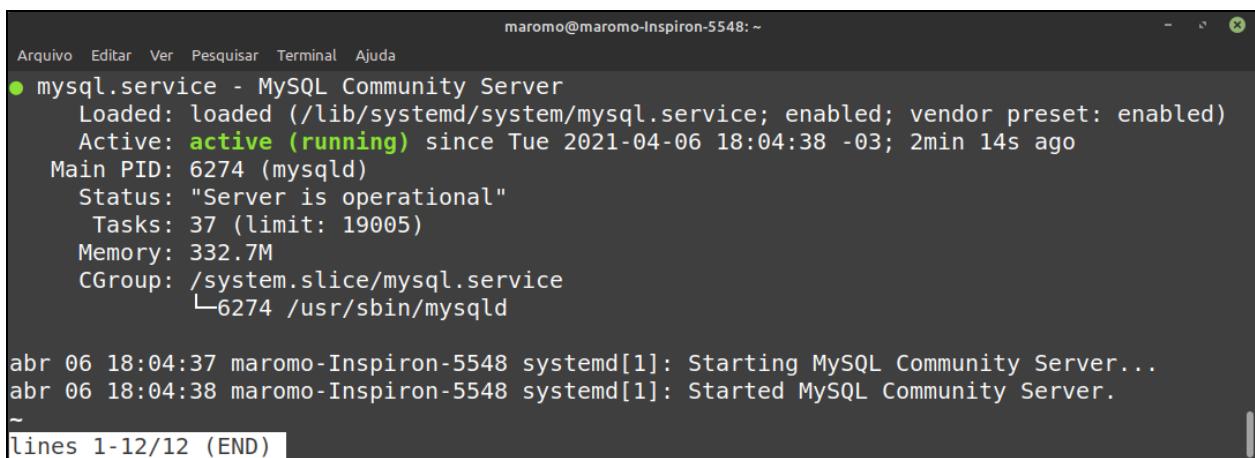
```
$ mysql --version
```

No momento da criação deste livro a versão atual do MySQL era:

```
mysql Ver 8.0.23-0ubuntu0.20.04.1 for Linux on x86_64 ((Ubuntu))
```

Após a instalação bem-sucedida, o serviço MySQL será iniciado automaticamente. Para verificar o status do servidor MySQL, execute o comando:

```
$ sudo systemctl status mysql
```



A screenshot of a terminal window titled "Terminal". The window shows the output of the command "systemctl status mysql.service". The output indicates that the MySQL Community Server is active and running, with a Main PID of 6274 and a Status of "Server is operational". It also shows the start logs for the MySQL server.

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-04-06 18:04:38 -03; 2min 14s ago
     Main PID: 6274 (mysqld)
       Status: "Server is operational"
         Tasks: 37 (limit: 19005)
        Memory: 332.7M
       CGroup: /system.slice/mysql.service
               └─6274 /usr/sbin/mysqld

abr 06 18:04:37 maromo-Inspiron-5548 systemd[1]: Starting MySQL Community Server...
abr 06 18:04:38 maromo-Inspiron-5548 systemd[1]: Started MySQL Community Server.
~
```

A tela mostra que o serviço MySQL está ativo e em execução.

Configurando o MySQL no Linux

O script `mysql_secure_installation`, que vem por padrão com a instalação do MySQL, nos permite proteger a segurança do MySQL. Execute o script `mysql_secure_installation` com o comando:

```
$ sudo mysql_secure_installation
```

Durante a execução do script `mysql_secure_installation`, você verá vários prompts. Primeiro, você verá três níveis de uma política de validação de senha. Você deve pressionar 'y' para definir uma senha. Além disso, selecione o nível de senha inserindo o respectivo dígito numérico.

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
maromo@maromo-Inspiron-5548:~$ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: [
```

Além disso, insira a senha e a reinsira para confirmação. A linha de comando exibirá a força estimada da senha.

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
ary          file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 0
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes,
any other key for No) : [
```

Na sequência, será solicitado a remover os usuários anônimos, testar os bancos de dados e recarregar a tabela de privilégios. Pressione 'y' ou 'n' de acordo com sua escolha e a configuração será realizada com sucesso. Sugiro que pressione 'y'.

Desabilite o login remoto dos usuários. Pressionando 'y'. Repita a operação para remover banco de dados de teste, ou seja, pressione 'y'.

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No)
: y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key
y for No) : y
```

Finalmente, pressione ‘y’ para recarregar a tabela de privilégios agora.

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda

Reload privilege tables now? (Press y|Y for Yes, any other key for No)
: y
Success.

All done!
maromo@maromo-Inspiron-5548:~$
```

Ainda no linux, vamos criar um novo usuário.

Para iniciar com o servidor MySQL a partir da linha de comando, use o comando:

```
$ sudo mysql
```

O shell interativo do MySQL será iniciado. Uma vez que o servidor MySQL está configurado, o MySQL cria um usuário root que pode gerenciar os bancos de dados e realizar várias ações administrativas.

```
create user aluno@localhost IDENTIFIED BY 'aluno1234';
```

```
maromo@maromo-Inspiron-5548: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
mysql> create user aluno@localhost IDENTIFIED BY 'aluno1234';
Query OK, 0 rows affected (0.02 sec)
```

Depois de criar o novo usuário com sucesso, você pode conceder os privilégios ao usuário da seguinte maneira:

```
GRANT ALL PRIVILEGES ON *.* TO aluno@localhost;
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO aluno@localhost;
Query OK, 0 rows affected (0.01 sec)
```

O *.* Concederá todos os tipos de privilégios ao usuário recém-criado. É aconselhável liberar os privilégios. Isso irá liberar a memória extra que o servidor armazena em cache ao criar um novo usuário.

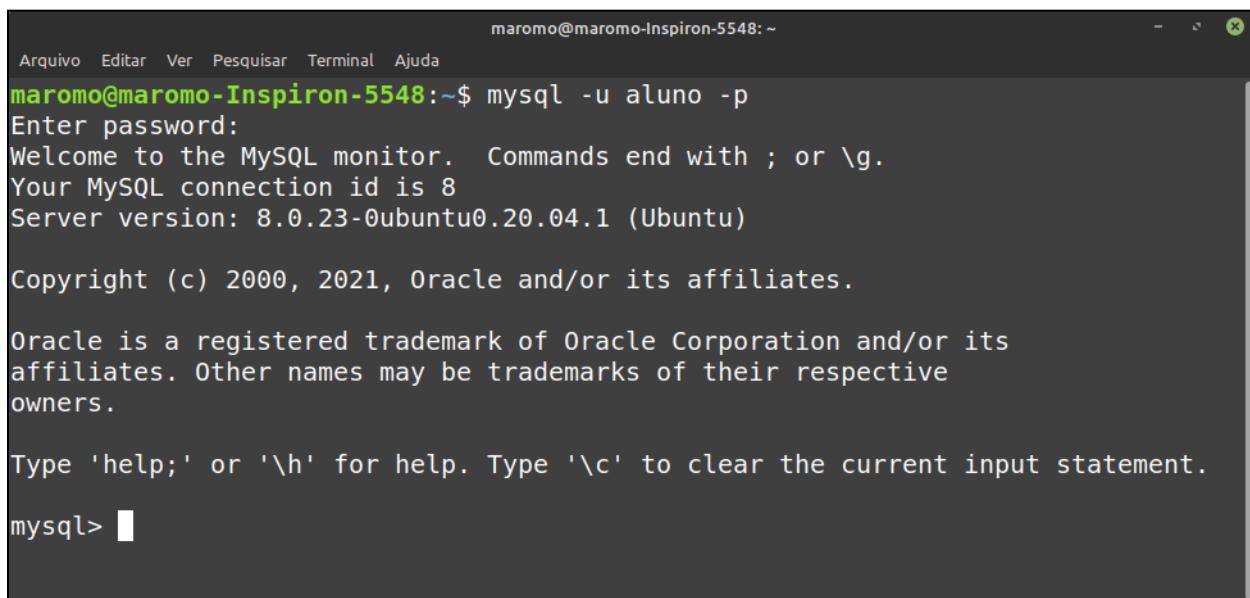
```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Fazer login no servidor MySQL e criar um banco de dados

Agora vamos fazer o login no servidor MySQL e criar um novo banco de dados. Para realizar um login, use o comando. Atenção o comando deve ser dado no prompt do shell.

```
$ mysql -u aluno -p
```

Digite a senha que você definiu anteriormente para este usuário e o shell interativo do MySQL será iniciado.



The screenshot shows a terminal window titled 'maromo@maromo-Inspiron-5548: ~'. The window contains the MySQL monitor output:

```
Arquivo Editar Ver Pesquisar Terminal Ajuda
maromo@maromo-Inspiron-5548:~$ mysql -u aluno -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.23-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

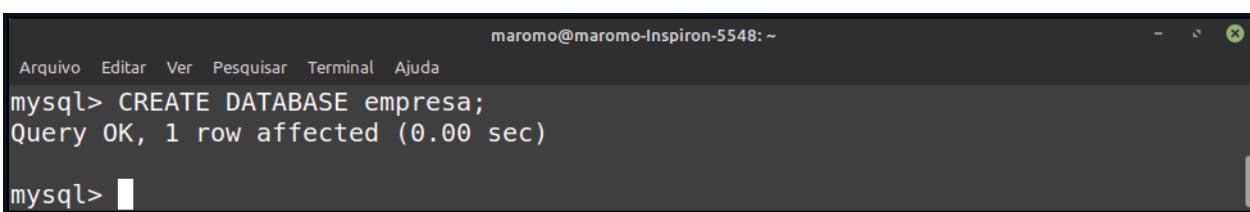
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

Para criar um novo banco de dados, execute o comando:

```
mysql> CREATE DATABASE empresa;
```



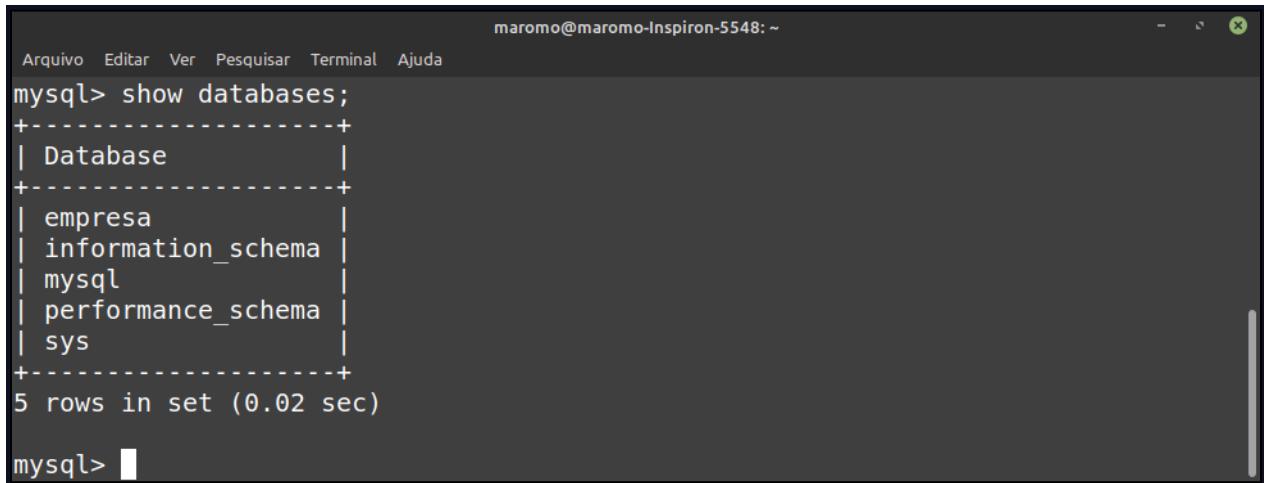
The screenshot shows a terminal window titled 'maromo@maromo-Inspiron-5548: ~'. The window contains the MySQL monitor output:

```
Arquivo Editar Ver Pesquisar Terminal Ajuda
mysql> CREATE DATABASE empresa;
Query OK, 1 row affected (0.00 sec)

mysql> █
```

Para ver a lista de todos os bancos de dados disponíveis, digite o comando:

```
mysql> show databases;
```



A screenshot of a terminal window titled "maromo@maromo-Inspiron-5548: ~". The window shows the MySQL command "show databases;" being run. The output lists five databases: "empresa", "information_schema", "mysql", "performance_schema", and "sys". Below the list, it says "5 rows in set (0.02 sec)". The MySQL prompt "mysql>" is visible at the bottom.

```
Arquivo Editar Ver Pesquisar Terminal Ajuda
mysql> show databases;
+-----+
| Database           |
+-----+
| empresa            |
| information_schema |
| mysql               |
| performance_schema |
| sys                |
+-----+
5 rows in set (0.02 sec)

mysql> |
```

Pronto!!! Temos um ambiente preparado e um banco criado. O que pode ajudar o trabalho do aluno, já que nesta disciplina o foco não é em Banco de Dados, é instalar um aplicativo com interface gráfica para facilitar a interface com o banco. Sugerimos o aplicativo MySQL Workbench do mesmo fabricante, que possui versões tanto para Windows como para Linux.

Conhecendo o PDO

PHP Data Objects ou simplesmente (PDO) é uma interface leve e consistente, que tem por objetivo acessar sistemas gerenciadores de bancos de dados em PHP. Cada driver de banco de dados que implementa a interface PDO permite acesso a recursos específicos do banco de dados, como o MySQL, por exemplo.

Você não pode executar nenhuma função de um banco de dados usando a extensão PDO sozinha; você deve usar um driver PDO específico do banco de dados (instalar) para acessar um servidor de banco de dados (como é o caso do MySQL).

O PDO fornece uma camada de abstração de acesso a dados, independentemente de qual banco de dados você está usando, você usa as mesmas funções para emitir consultas e buscar dados. O PDO está disponível desde a versão 5.1 do PHP . O PDO requer os novos recursos OO no kernel do PHP 5 e, portanto, não será executado com versões anteriores do PHP.

Nota sobre as versões de APIs para Banco de Dados

Existem três API's de conexão com o banco de dados em PHP, são elas:

- mysql: Pacote de funções para acesso ao MySQL, foi descontinuado no PHP7.
- mysqli: Extensão da API mysql com suporte a funcionalidades adicionadas a versões posteriores ao MySQL 4.1 - <http://www.mysql.com/>
- PDO - PHP Data Objects: Interface para acesso a dados do PHP.

Como vimos, por tratar-se da mais recente, o foco do nosso estudo será a biblioteca PDO - PHP Data Objects.

PDO é Robusta

O PDO é uma biblioteca robusta que pode ser usada, independentemente do driver que você estiver usando. Muitos ficam um pouco intimidados em usar o PDO de início, não por se tratar de uma API difícil de usar, muito pelo contrário, pois é muito fácil de usar, mas sim porque a API mysql é muito fácil de usar e as pessoas acabam se acostumando com ela.

Mas vamos ver como realmente é fácil de usá-la.

Conexão com Banco de Dados

Primeiro vamos ver como se conectar usando o PDO. Nós vamos criar uma nova instância de classe e especificar o driver que vamos usar, no caso o mysql, o nome do banco de dados, nome de usuário e senha.

```
<?php  
$conn = new PDO( 'mysql:host=localhost;dbname=meuBancoDeDados' ,  
                 $username,  
                 $password);
```

Como vimos no código anterior, também é muito simples se conectar usando o PDO, mas como em toda conexão, é preciso tratar os erros, para que se por ventura aconteça algum durante a conexão, mostrar o erro ao usuário.

```
<?php  
try {
```

```

$conn = new PDO('mysql:host=localhost;dbname=meuBancoDeDados',
    $username,
    $password);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}

```

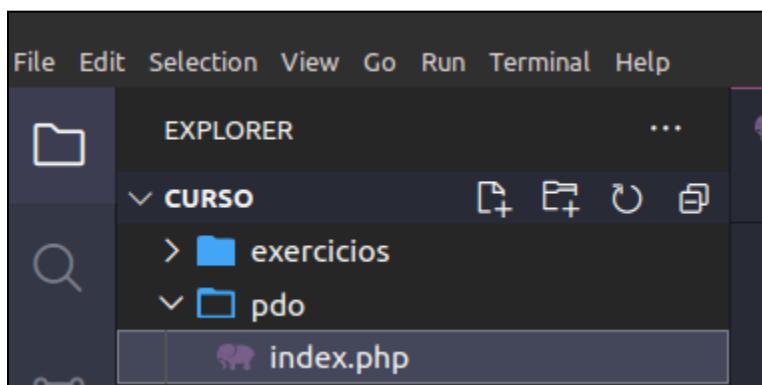
Sobre os Erros

O erro padrão do PDO é o PDO::ERRMODE_SILENT, mas no nosso código usamos o PDO::ERRMODE_EXCEPTION. Abaixo listamos as opções que temos:

- PDO::ERRMODE_SILENT
- PDO::ERRMODE_WARNING
- PDO::ERRMODE_EXCEPTION

Criando um Banco de Dados

Para efeito prático, vamos criar um novo banco de dados, chamado **Escola** no MySQL e no Visual Studio Code vamos criar um diretório chamado **pdo**. Neste diretório teremos apenas um arquivo chamado index.php neste momento. Veja a próxima figura:



Vamos configurar o aplicativo Workbench caso ainda não o tenha feito em outras disciplinas.

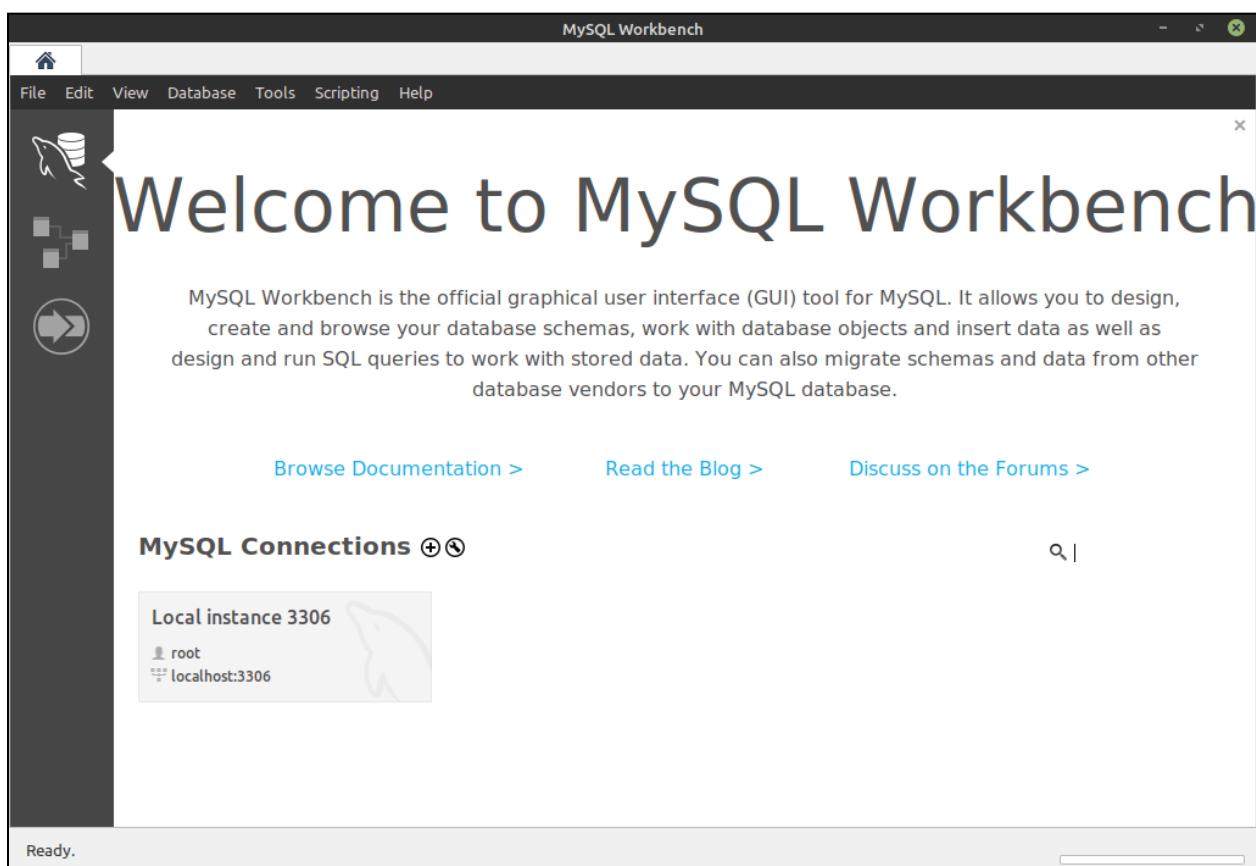
Aplicativo Workbench

MySQL Workbench é uma ferramenta com interface gráfica com o usuário, própria para arquitetos de banco de dados, desenvolvedores e DBAs. Com o MySQL Workbench é possível realizar a modelagem de dados, o desenvolvimento de SQL e além de contar com várias ferramentas de administração para configuração de servidor, administração de usuário, backup entre outros. O MySQL Workbench está disponível no Windows, Linux e Mac OS X.

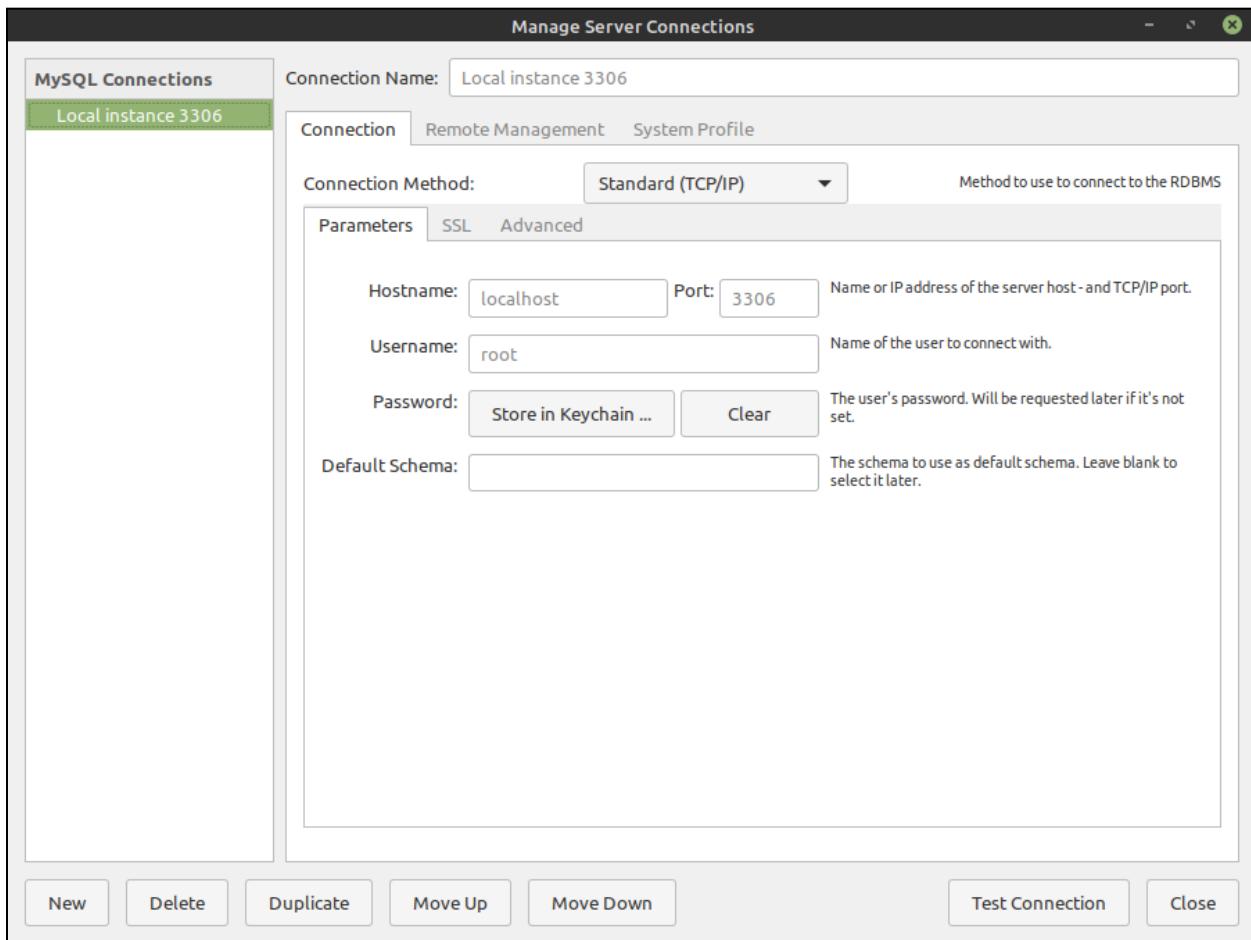
Baixe o aplicativo para Windows ou Linux no site do fabricante:

<https://dev.mysql.com/downloads/workbench/>.

Depois de instalado, execute o aplicativo.

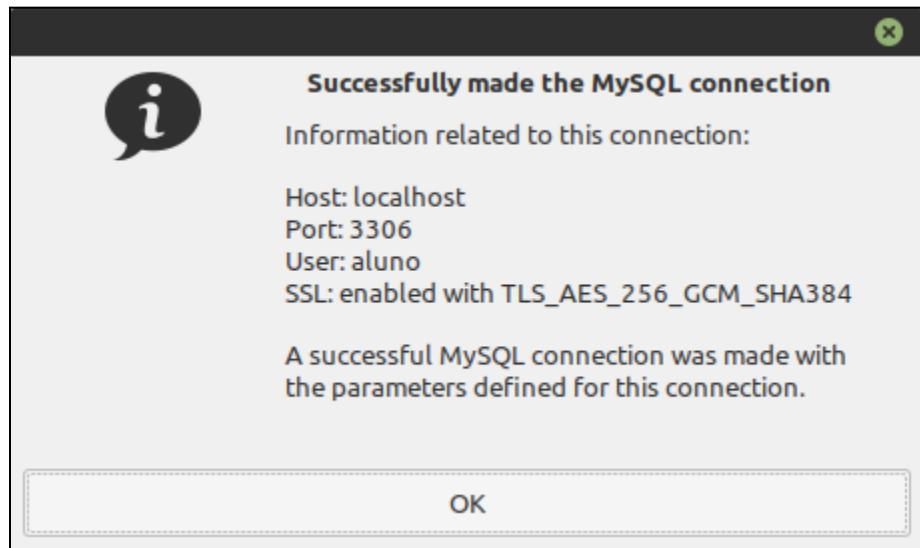


Na janela inicial do programa em MySQL Connections clique no símbolo de chave de fenda (configuração). Será aberta a janela a seguir.



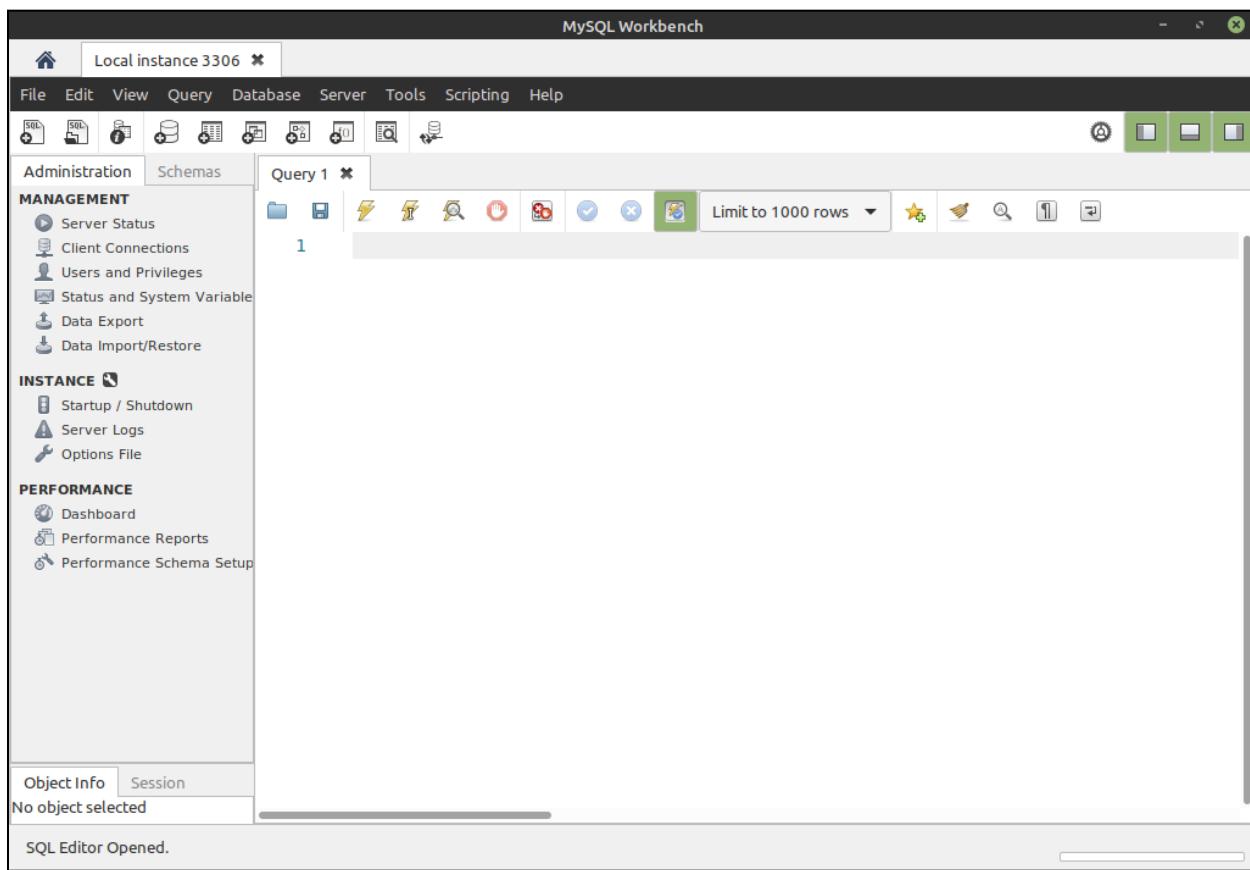
Na janela Manage Server Connections, observe que o Username é “root”, troque para “aluno”. Depois clique em “Store in Keychain..” (Armazenar senha) e digite a senha “aluno1234” sem as aspas.

Nesta mesma janela, agora clique no botão localizado na parte inferior dela, chamado Test Connection. Se tudo ocorreu bem, você verá a mensagem de sucesso na conexão.



Feche a janela da mensagem (OK) e depois feche a janela da configuração. Assim, voltamos à janela inicial do programa.

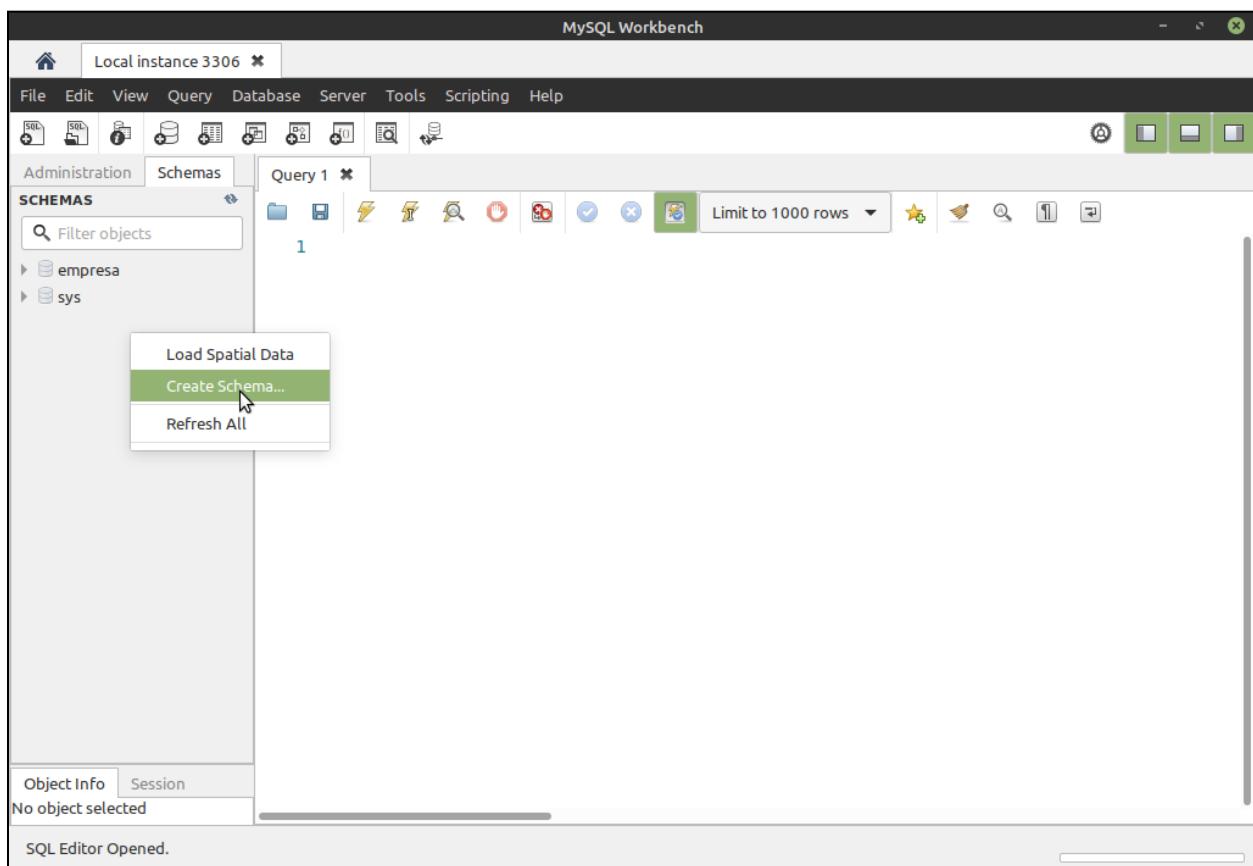
Agora é só clicar sobre a instância local 3306. Você será conduzido a próxima janela.



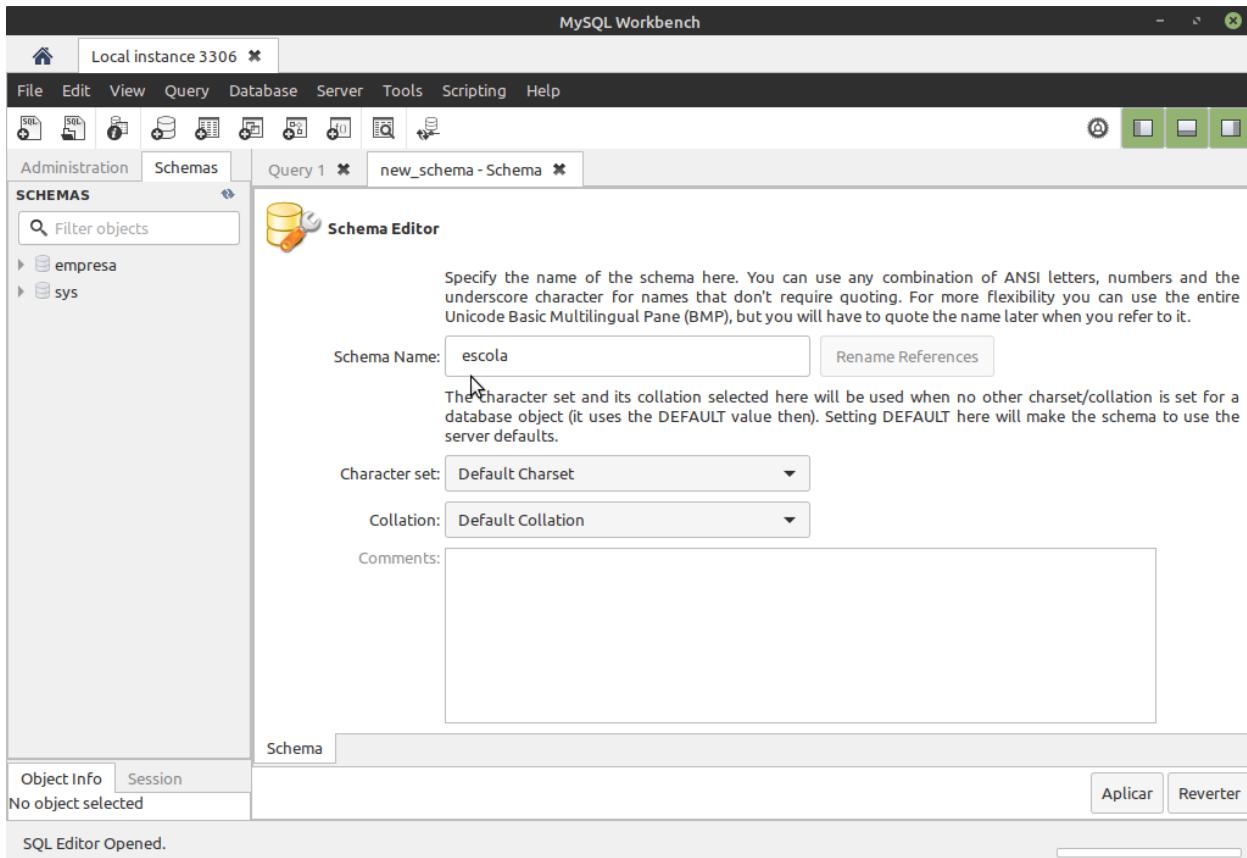
Olhando para a janela, na parte esquerda superior há duas abas a saber: Administration e Schemas. Clique em Schemas para acessar aos bancos de dados disponíveis. Se você está usando o ambiente Windows verá os bancos criados por padrão durante a instalação, se estiver usando o ambiente Linux, além das atuais verá o banco criado durante a configuração do mysql no Linux, ou seja, o banco de dados: empresa. Não há problemas aqui, se você não o tiver, pois criaremos um novo agora.

Para criar um novo banco de dados chamado **Escola**. Faça assim:

1. Com a janela schemas ativada, clique com o botão direito do mouse na parte vazia e selecione a opção Create Schema... Veja na próxima figura.



Na janela que se abrirá em “Schema Name” (nome do banco) digite: **escola**. Depois clique em Aplicar, Aply (Aplicar) novamente e Close para fechar a janela.



Pronto, agora temos o nosso banco de dados escola criado.

O nome do Banco de Dados já deve aparecer do lado esquerdo da janela principal. Dê um duplo clique nele para ativá-lo.

Agora copie o código (script) abaixo para a janela Query1.

```
CREATE TABLE `tb_alunos` (
  `rm` int(11) NOT NULL,
  `nome` varchar(60) DEFAULT NULL,
  `email` varchar(60) DEFAULT NULL,
  PRIMARY KEY (`rm`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Clique no segundo botão com desenho de um raio. Veja na figura.

The screenshot shows the MySQL Workbench interface. In the top left, it says "Local instance 3306". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. On the left, there's a sidebar titled "Administration" with "Schemas" selected. It lists three schemas: "empresa", "escola", and "sys". The "escola" schema is currently active, as indicated by the selected icon. The main area is a "Query 1" editor with the following SQL code:

```
3   `nome` varchar(60) DEFAULT NULL,
4   `email` varchar(60) DEFAULT NULL,
5   PRIMARY KEY (`rm`)
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

A yellow box highlights the "Execute" button in the toolbar, which has a lightning bolt icon. A tooltip above the button says "Execute the statement under the keyboard cursor".

Pronto, sua tabela foi criada.

Para ver o resultado, clique com o botão direito do mouse sobre o nome do banco de dados, escola. E escolha a opção **Refresh All**. Assim o banco de dados será atualizado e a tabela aparecerá, conforme mostra a próxima figura.

The screenshot shows the MySQL Workbench interface. In the left sidebar under 'Schemas', the 'escola' schema is selected. In the 'Tables' section of the schema tree, there is a table named 'tb_alunos'. The 'Columns' section shows three columns: 'rm', 'nome', and 'email'. A context menu is open over the 'tb_alunos' table, with the 'Select Rows - Limit 1000' option highlighted. The main query editor window contains the following SQL code:

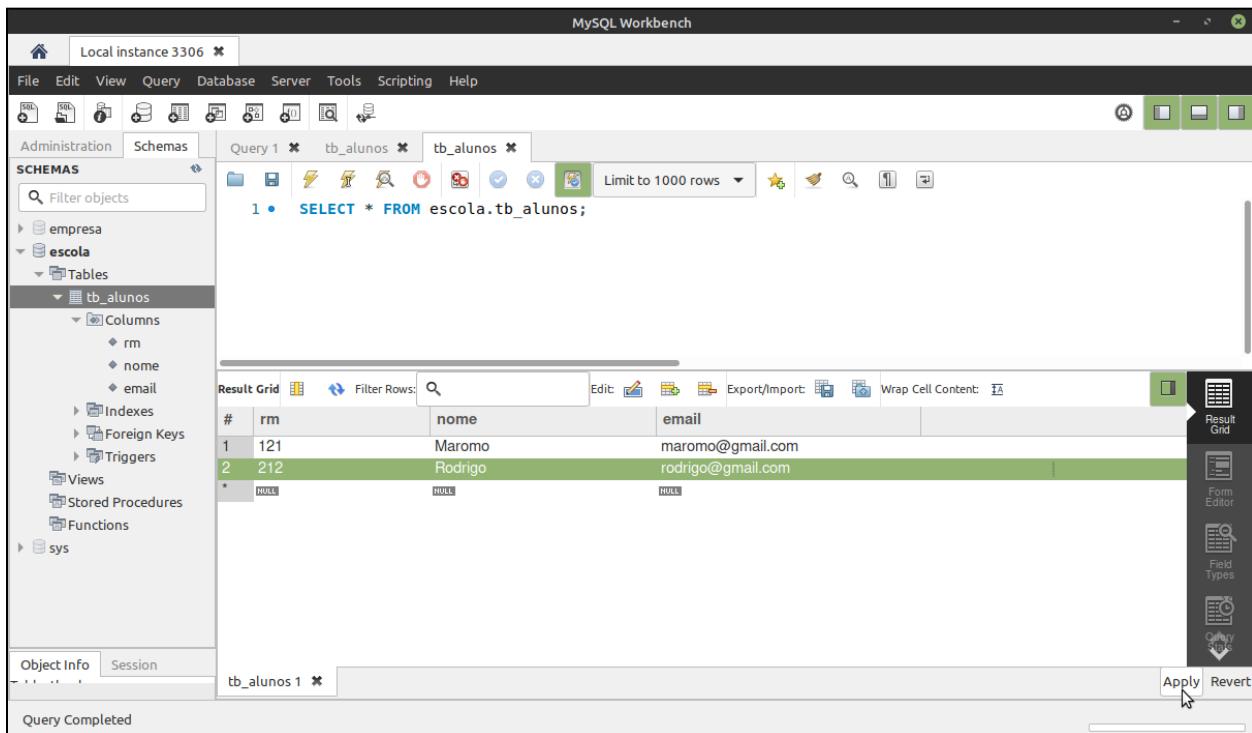
```
1 • CREATE TABLE `tb_alunos` (
2     `rm` int(11) NOT NULL,
3     `nome` varchar(60) DEFAULT NULL,
4     `email` varchar(60) DEFAULT NULL,
5     PRIMARY KEY (`rm`)
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Popule a tabela com alguns exemplos, para isso clique com o botão direito do mouse sobre o nome da tabela, ou seja, **tb_alunos** e escolha a opção **Select Rows - Limit 1000**.

The screenshot shows the MySQL Workbench interface with the 'tb_alunos' table selected. A context menu is open over the table, and the 'Select Rows - Limit 1000' option is highlighted. The main query editor window contains the following SQL code:

```
1 • SELECT * FROM escola.tb_alunos;
```

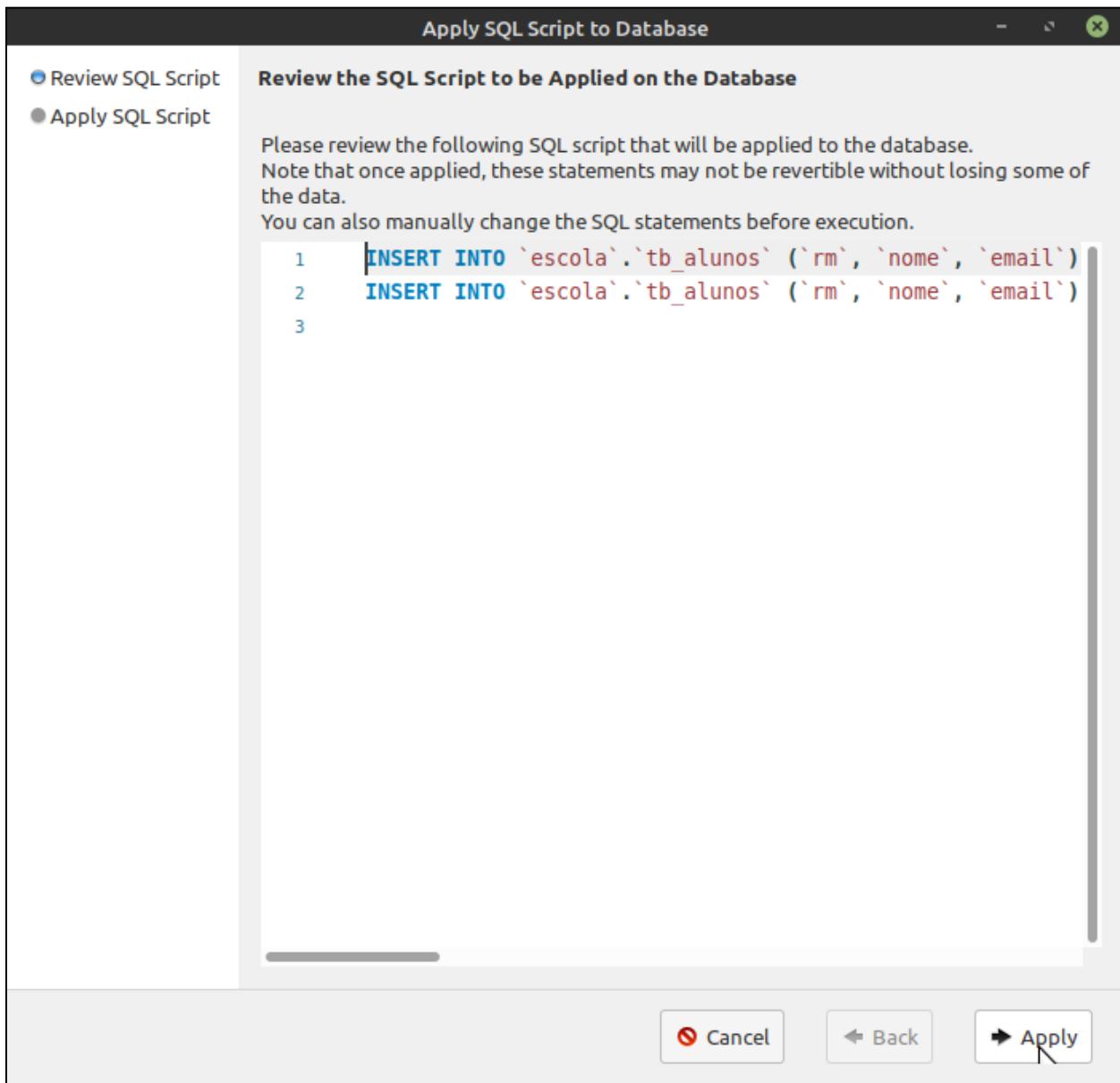
Agora digite alguns dados de exemplo e depois clique no botão Apply (Aplicar) e confirme o script criado, clicando sobre Apply novamente. Estas instruções Insert Into, são as cláusulas mysql para inserir novos registros no Banco.



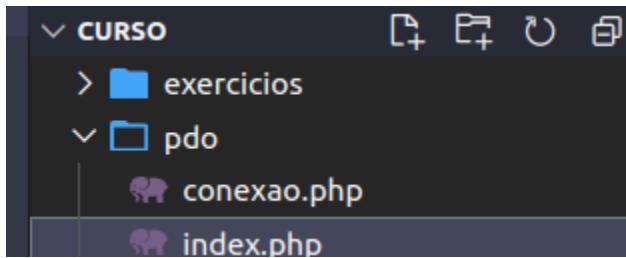
The screenshot shows the MySQL Workbench interface. On the left, the Schemas panel displays the 'escola' schema with its tables, including 'tb_alunos'. The 'tb_alunos' table is selected, showing columns: rm, nome, and email. A 'Result Grid' window is open, displaying the following data:

#	rm	nome	email
1	121	Maromo	maromo@gmail.com
2	212	Rodrigo	rodrigo@gmail.com
*	NULL	NULL	NULL

At the bottom right of the Result Grid window, there are 'Apply' and 'Revert' buttons. The 'Apply' button is highlighted with a mouse cursor.



Agora no Visual Studio Code na pasta **pdo**, que já possuímos um arquivo **index.php** vazio. Crie um novo arquivo chamado **conexao.php**.



Volte ao arquivo **index.php** e defina como abaixo:

```
<?php
require_once "conexao.php";
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Lista de Alunos</title>
</head>

<body>
    <h2>Lista de Alunos</h2>
    <?php
        //definir a codificação para UTF-8 (evitar problemas com acentos,
cedilhas)
        $conn->exec("set names utf8");
        //preparando a consulta
        $sql = "Select * from tb_alunos;";
        $result = $conn->query($sql);
        $rows = $result->fetchAll(PDO::FETCH_ASSOC);
    ?>
    <table style="width:33%; border-collapse: collapse;">
        <thead>
            <tr style="border: solid 1px">
                <th>RM</th>
                <th>Nome</th>
```

```

        <th>Email</th>
    </tr>
</thead>
<tbody>
<?php
//exibindo um array associativo
foreach ($rows as $row) {
?
<tr style="border: solid 1px">
    <td><?=$row['rm']?></td>
    <td><?=$row['nome']?></td>
    <td><?=$row['email']?></td>
</tr>
<?php
}
?
</tbody>
</table>
</body>
</html>

```

Tendo a instância da classe PDO na variável \$conn, podemos realizar um simples SELECT para realizar a busca no banco de dados todas as linhas referentes aos alunos. Vamos usar um array associativo para isto.

Arquivo:conexao.php

```

<?php
$username = 'aluno';
$password = 'aluno1234';
try{
    $conn = new PDO(
        'mysql:host=localhost;dbname=escola',
        $username,
        $password
    );
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}catch(PDOException $e){

```

```
echo "Aconteceu um erro ao conectar com o MySQL:  
$e->getMessage()";  
}
```

Exemplo de Resposta do arquivo index.php:

```
$ php -S localhost:8000/pdo
```

The screenshot shows a browser window titled "Lista de Alunos - Chromium". The address bar contains the URL "localhost:8000/pdo". The main content area displays a table with the following data:

RM	Nome	Email
121	Maromo	maromo@gmail.com
212	Rodrigo	rodrigo@gmail.com

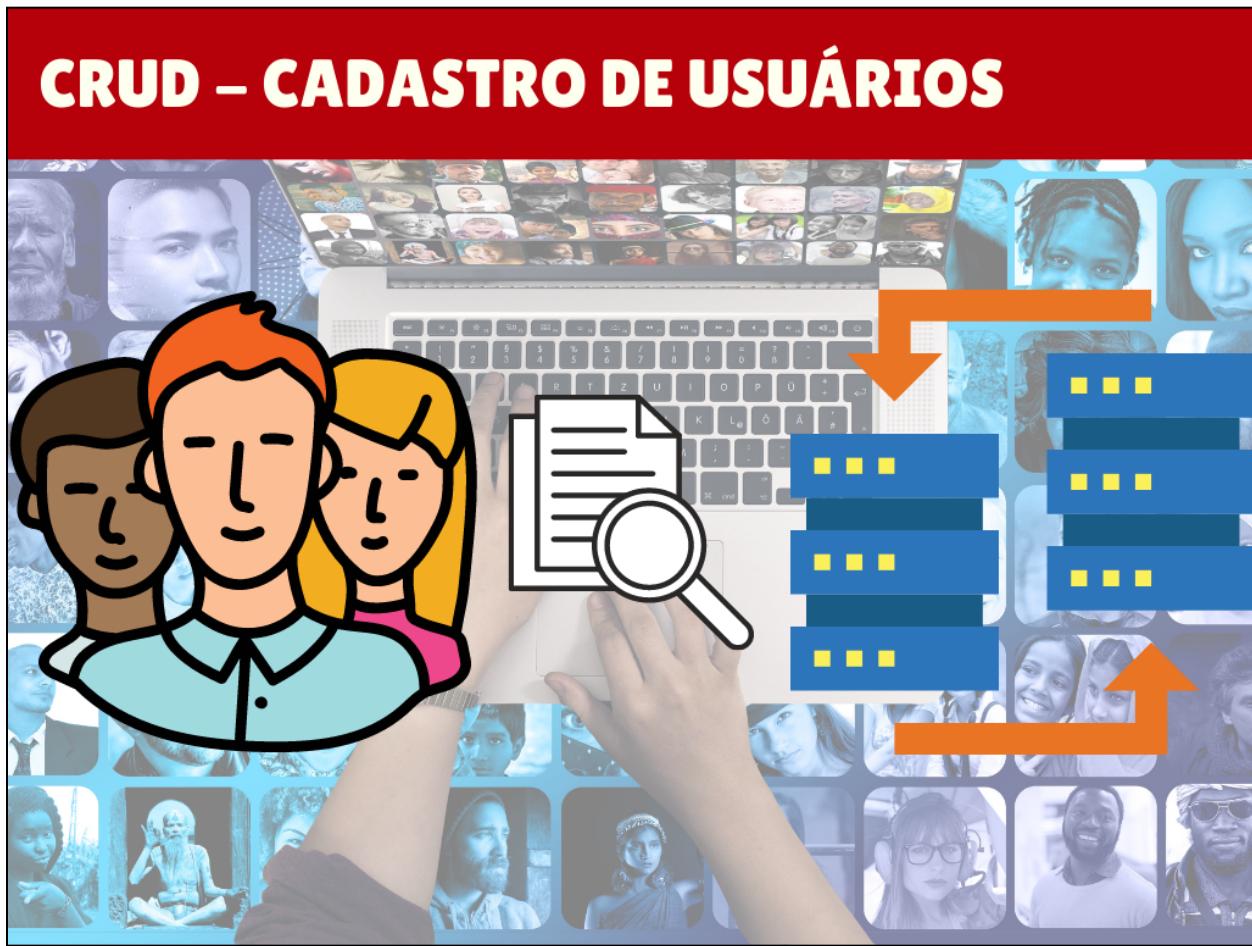
Exercícios de Fixação

1. O que é um SGBD ? E o MySQL é uma boa opção ? Responda e Justifique.
2. Prepare o ambiente da sua máquina conforme os passos necessários para o seu sistema operacional.
3. Crie um novo Banco de Dados chamado Livraria.
4. Crie dentro deste Banco de Dados uma Tabela chamada livros com os seguintes campos: ISBN, nome, ano de publicação, editora e autor.

Aplicativo CRUD

Em programação de computador, criar, ler, atualizar e excluir (CRUD) é um estilo de arquitetura de software relacionado às quatro operações básicas de persistência em armazenamento de dados.

Às vezes, palavras alternativas são usadas ao definir as quatro operações básicas do CRUD, (Create, Read, Update, Delete), como construir em vez de criar, recuperar em vez de ler ou destruir ou vez de excluir.



Imagine um sistema de catálogo de usuários, sem ao menos essas quatro operações, um software não pode ser considerado completo. Como essas operações são tão fundamentais, muitas vezes são documentadas e descritas sob um título abrangente,

como "gerenciamento de usuários", "gerenciamento de contatos" ou "manutenção de usuários".

Neste material vamos desenvolver um simples sistema de cadastro, usando PHP e MySQL, utilizando a biblioteca PDO.

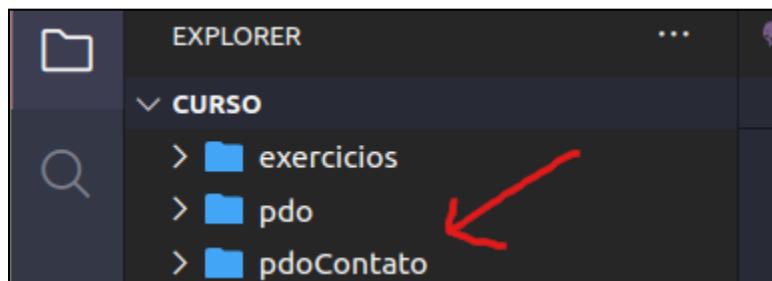
Vamos fazer um cadastro simples de usuários, com os seguintes dados:

- nome,
- email,
- gênero (sexo) e
- data de nascimento.

Teremos uma tela de listagem de usuários, uma de cadastro, uma de edição e outra de remoção. O famoso CRUD.

Estrutura do Projeto em PHP

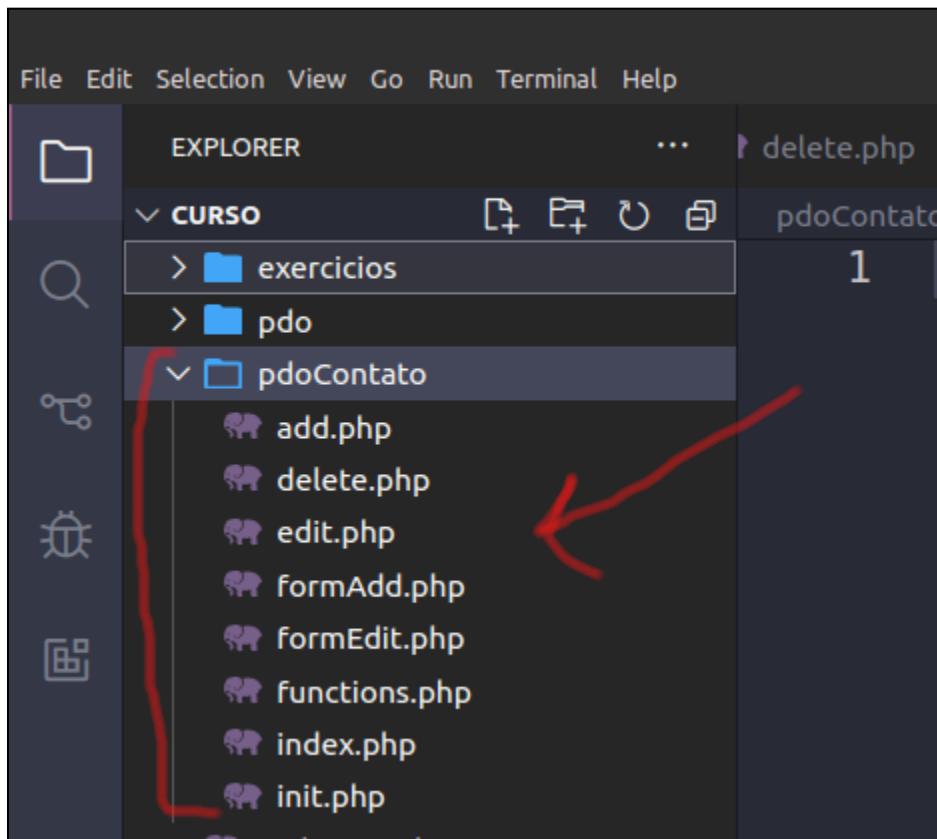
Para facilitar nosso trabalho. Primeiro crie no Visual Studio Code uma pasta (diretório) chamado **pdoContato**.



Adicione os seguintes arquivos dentro deste diretório:

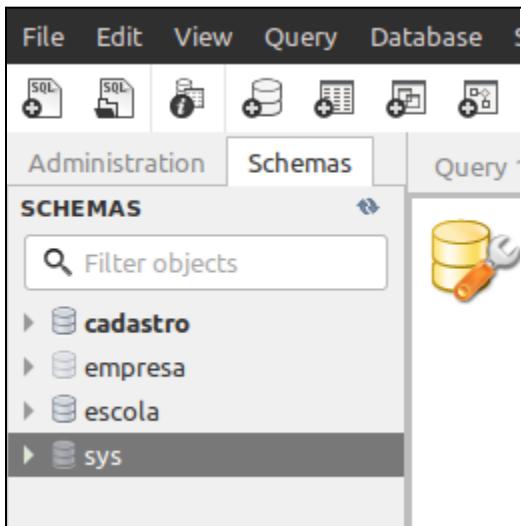
- **add.php**: script que processa o cadastro de usuário.
- **delete.php**: script que processa a remoção de usuário.
- **edit.php**: script que processa a edição de usuário
- **formAdd.php**: formulário de cadastro de usuário
- **formEdit.php**: formulário de edição de usuário
- **functions.php**: arquivo de funções
- **index.php**: listagem de usuários cadastrados
- **init.php**: arquivo de inicialização (Bootstrapping)

Desta forma teremos:



Estrutura do Banco de dados

Nosso banco será bem simples, apenas com uma tabela de usuários. Crie no WorkBench um novo schema chamado **cadastro**.

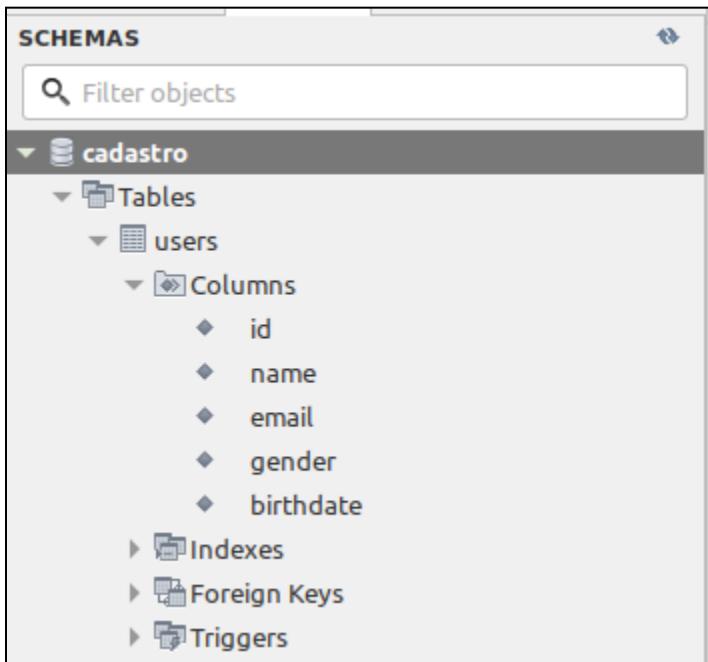


Em uma aba Query, execute o script a seguir:

```
CREATE TABLE users(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT primary KEY, -- id
    name VARCHAR(60) NOT NULL, -- nome
    email VARCHAR(80) NOT NULL, -- email
    gender ENUM('m', 'f') NOT NULL, -- gênero (masculino, feminino)
    birthdate DATE NOT NULL -- data de nascimento
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Ao finalizar a execução deste arquivo teremos uma nova tabela chamada **users**, com os seguintes campos:

- id - campo numérico e auto incrementado.
- name - campo que identifica o nome do usuário.
- email - campo que identifica o seu email.
- gender - gênero, um enumerador com 'm' para masculino e 'f' para feminino.
- birthdate - data de nascimento.



Neste tópico apresentamos com detalhes o conteúdo de cada arquivo do nosso projeto.

Arquivo init

O arquivo init.php define algumas configurações gerais do nosso projeto, como:

- As credenciais de acesso ao banco de dados cadastro.
- A definição da timezone (região).
- Habilitação de erros (mensagens de erro).
- O arquivo de inicialização (bootstrapping) deve ser chamado em todos os nossos scripts.

A ideia aqui é termos um ambiente consistente e organizado, sempre com todas as mesmas configurações.

Nota: os scripts serão comentados para melhor compreensão.

```
<?php  
// constantes com as credenciais de acesso ao banco MySQL  
define('DB_HOST', 'localhost');  
define('DB_USER', 'aluno');
```

```

define('DB_PASS', 'aluno1234');
define('DB_NAME', 'cadastro');

// habilita todas as exibições de erros
ini_set('display_errors', true);
error_reporting(E_ALL);
// define a região para fuso horários e formatos de data
date_default_timezone_set('America/Sao_Paulo');

// inclui o arquivo de funções (funções adicionais)
require_once 'functions.php';

```

Arquivo functions

No arquivo **functions.php** temos três funções, a saber:

- **function db_connect()** - estabelece uma conexão com o banco de dados retornando um objeto de conexão PDO.
- **function dataConvert(\$date)** - função que recebe uma data no formato ISO (yyyy-mm-dd) - padrão americano e converte para o padrão brasileiro: dd/mm/yyyy.
- **function calculateAge(\$bithdate)** - função que recebe uma data de nascimento e calcula a idade do usuário a partir dela.

Observe os comentários no script.

```

<?php

//Conecta com o MySQL usando PDO
function db_connect() {
    $PDO = new PDO('mysql:host=' . DB_HOST . ';dbname=' . DB_NAME .
    ';charset=utf8', DB_USER, DB_PASS);
    return $PDO;
}

//Função para conversão de datas
//Fonte:

```

```

href="http://rberaldo.com.br/php-usando-a-classe-nativa-datetime/
function dateConvert($date) {
    if (!strstr($date, '/')) {
        // $date está no formato ISO (yyyy-mm-dd) e deve ser
        convertida
        // para dd/mm/yyyy
        sscanf($date, '%d-%d-%d', $y, $m, $d);
        return sprintf('%02d/%02d/%04d', $d, $m, $y);
    } else {
        // $date está no formato brasileiro e deve ser convertida
        para ISO
        sscanf($date, '%d/%d/%d', $d, $m, $y);
        return sprintf('%04d-%02d-%02d', $y, $m, $d);
    }

    return false;
}

//Função que calcula a idade a partir da data de nascimento
//Fonte: http://rberaldo.com.br/php-usando-a-classe-nativa-datetime/
function calculateAge($birthdate) {
    $now = new DateTime();
    $diff = $now->diff(new DateTime($birthdate));

    return $diff->y;
}

```

Arquivo index

O arquivo **index.php** é o ponto de entrada do nosso aplicativo. Ao ser executado, tem como objetivo apresentar a lista de usuários e links para navegação para outras funções. O código está todo documentado.

```

<?php
require_once 'init.php';
// abre a conexão

```

```

$PDO = db_connect();
// SQL para contar o total de registros
// A biblioteca PDO possui o método rowCount(),
// mas ele pode ser impreciso.
// É recomendável usar a função COUNT da SQL
$sql_count = "SELECT COUNT(*) AS total FROM users";
// SQL para selecionar os registros
$sql = "SELECT * FROM `users`";

// conta o total de registros
$stmt_count = $PDO->prepare($sql_count);
$stmt_count->execute();
$total = $stmt_count->fetchColumn();

// seleciona os registros
$stmt = $PDO->prepare($sql);
$stmt->execute();
?>
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Sistema de Cadastro</title>
    </head>
    <body>
        <h1>Sistema de Cadastro</h1>
        <p><a href="formAdd.php">Adicionar Usuário</a></p>
        <h2>Lista de Usuários</h2>
        <p>Total de usuários: <?php echo $total ?></p>
        <?php if ($total > 0): ?>
            <table width="50%" border="1">
                <thead>
                    <tr>
                        <th>Nome</th>
                        <th>Email</th>
                        <th>Gênero</th>
                        <th>Data de Nascimento</th>
                </thead>

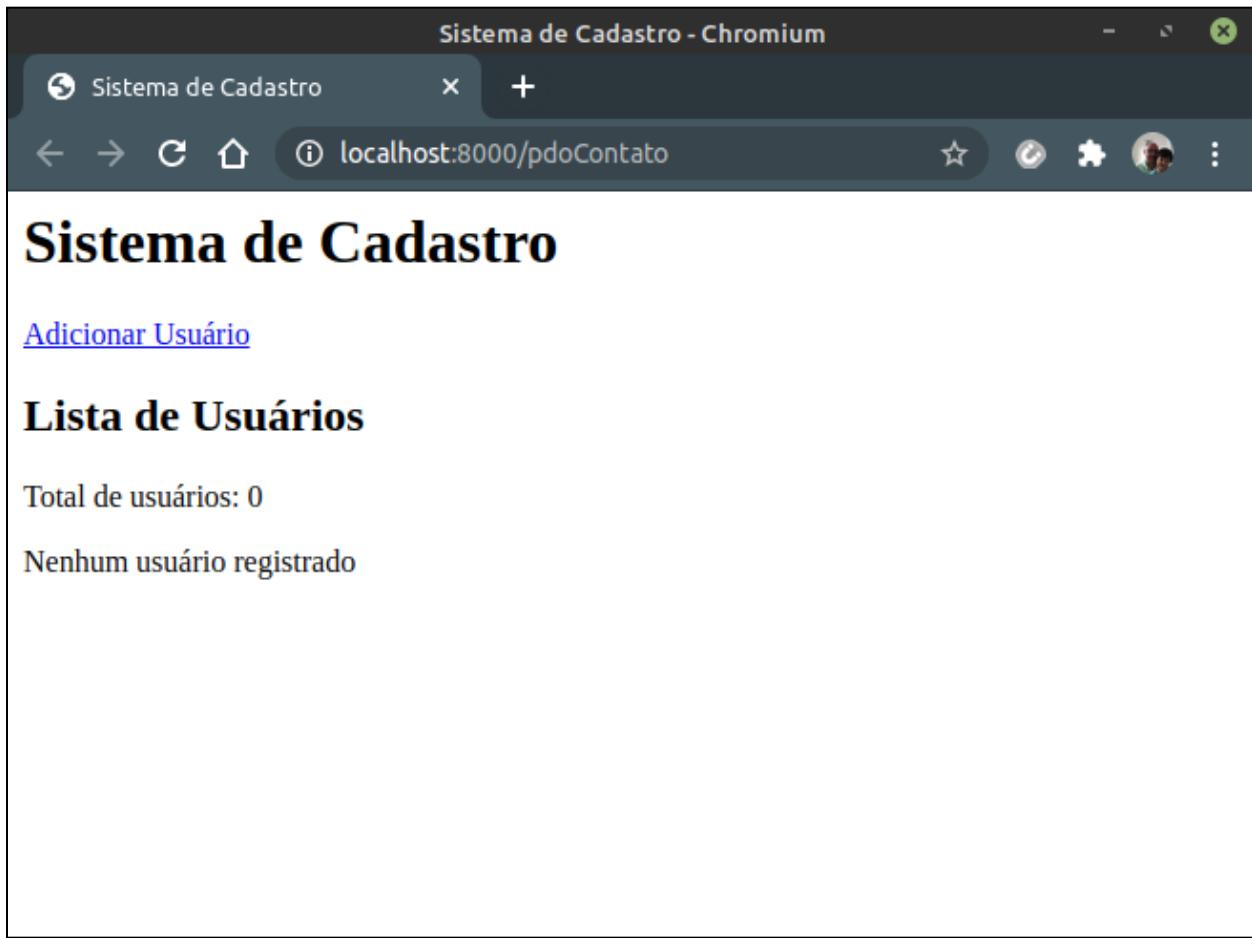
```

```

        <th>Idade</th>
        <th>Ações</th>
    </tr>
</thead>
<tbody>
    <?php while ($user =
$stmt->fetch(PDO::FETCH_ASSOC)): ?>
    <tr>
        <td><?=$user['name'] ?></td>
        <td><?=$user['email'] ?></td>
        <td><?=( $user['gender'] == 'm') ?>
'Masculino' : 'Feminino' ?></td>
        <td><?=dateConvert($user['birthdate']) ?>
?></td>
        <td><?=calculateAge($user['birthdate']) ?>
anos</td>
        <td>
            <a href="formEdit.php?id=<?php echo
$user['id'] ?>">Editar</a>
            <a href="delete.php?id=<?php echo
$user['id'] ?>"
                onclick="return confirm('Tem
certeza de que deseja remover?');">
                Remover
            </a>
        </td>
    </tr>
    <?php endwhile; ?>
</tbody>
</table>
<?php else: ?>
    <p>Nenhum usuário registrado</p>
<?php endif; ?>
</body>
</html>

```

Resultado da execução do arquivo index.php até o momento:



Como nossa lista está vazia, ou seja, ainda não foi criado nenhum usuário. O link **Adicionar Usuário** deve apresentar, na tela, um formulário HTML próprio para a inclusão de dados no Banco de Dados. Os dois arquivos que devemos preparar para esta funcionalidade, além do arquivo de cadastro “**formAdd.php**” também precisamos do script que fará o tratamento dos dados, o arquivo “**add.php**”.

Arquivo **formAdd**

A seguir o código do arquivo **formAdd.php**.

```
<?php  
require 'init.php';  
?>  
<!doctype html>  
<html lang="pt-br">  
</html>
```

```

<head>
    <meta charset="utf-8">
    <title>Cadastro de Usuário</title>
</head>
<body>
    <h1>Sistema de Cadastro</h1>
    <h2>Cadastro de Usuário</h2>
    <form action="add.php" method="post">
        <label for="name">Nome: </label>
        <br>
        <input type="text" name="name" id="name">
        <br><br>
        <label for="email">Email: </label>
        <br>
        <input type="text" name="email" id="email">
        <br><br>
        Gênero:
        <br>
        <input type="radio" name="gender" id="gender_m" value="m">
        <label for="gender_m">Masculino </label>
        <input type="radio" name="gender" id="gender_f" value="f">
        <label for="gender_f">Feminino </label>
        <br><br>
        <label for="birthdate">Data de Nascimento: </label>
        <br>
        <input type="text" name="birthdate" id="birthdate"
placeholder="dd/mm/YYYY">
        <br><br>
        <input type="submit" value="Cadastrar">
    </form>
</body>
</html>

```

Arquivo add

Agora o arquivo que recebe e trata os dados, o arquivo **add.php**.

```
<?php
require_once 'init.php';

// pega os dados do formulário
$name = isset($_POST['name']) ? $_POST['name'] : null;
$email = isset($_POST['email']) ? $_POST['email'] : null;
$gender = isset($_POST['gender']) ? $_POST['gender'] : null;
$birthdate = isset($_POST['birthdate']) ? $_POST['birthdate'] : null;
var_dump($_POST);

// validação (bem simples, só pra evitar dados vazios)
if (empty($name) || empty($email) || empty($gender) ||
empty($birthdate))
{
    echo "Volte e preencha todos os campos";
    exit;
}
// a data vem no formato dd/mm/YYYY
// então precisamos converter para YYYY-mm-dd
$isoDate = dateConvert($birthdate);
// insere no banco
$PDO = db_connect();
$sql = "INSERT INTO `users` (
    `name`,
    `email`,
    `gender`,
    `birthdate`
) VALUES (
    :name,
    :email,
    :gender,
    :birthdate
)";

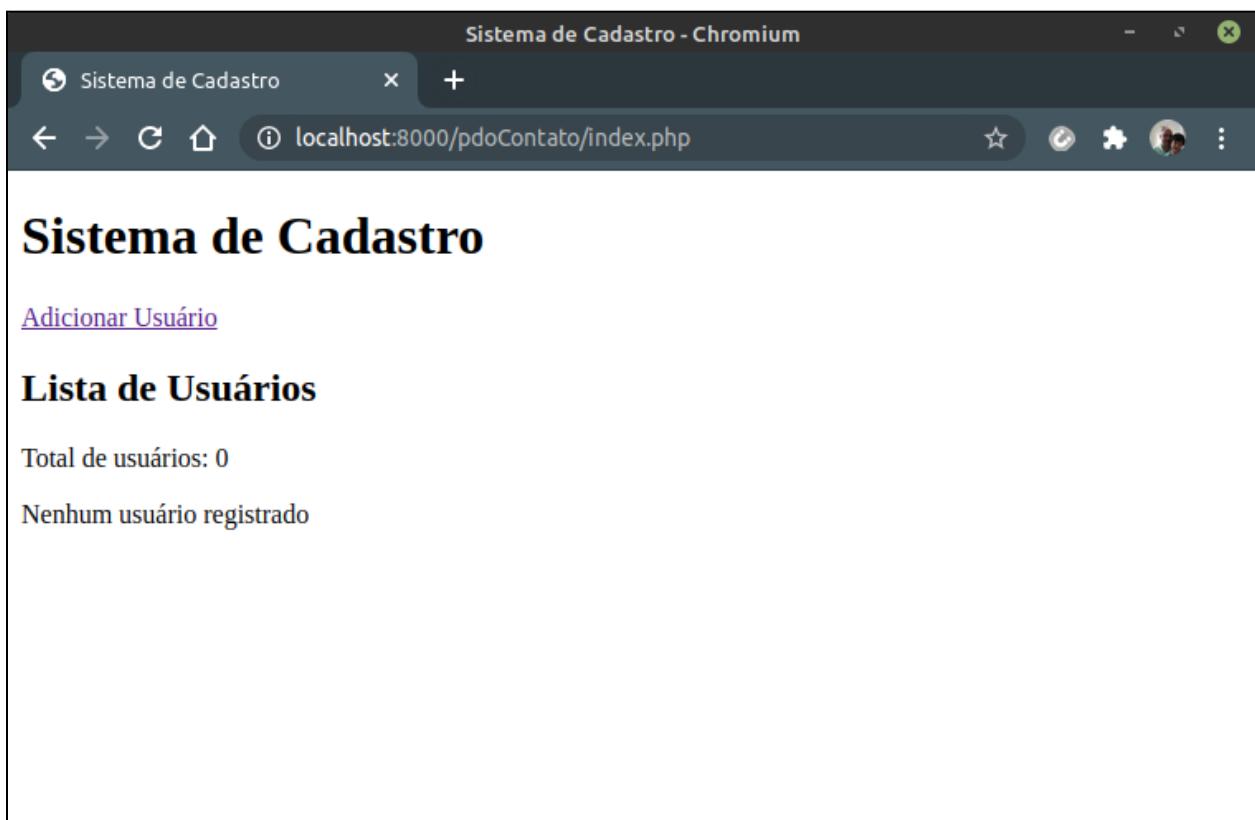
```

```
$stmt = $PDO->prepare($sql);
$stmt->bindParam(':name', $name, PDO::PARAM_STR);
$stmt->bindParam(':email', $email, PDO::PARAM_STR);
$stmt->bindParam(':gender', $gender, PDO::PARAM_STR);
$stmt->bindParam(':birthdate', $isoDate);
if ($stmt->execute())
{
    header('Location: index.php');
}
else
{
    echo "Erro ao cadastrar <br>";
    print_r($stmt->errorInfo());
}
```

Neste momento, levante o servidor local e execute o arquivo no index.php no navegador.

```
$ php -S localhost:8000
```

Digite o caminho completo do arquivo index no navegador:



Agora clique no link Adicionar Usuário para cadastrar um confirmando o envio, ao final da digitação.

Cadastro de Usuário - Chromium

Cadastro de Usuário

localhost:8000/pdoContato/formAdd.php

Sistema de Cadastro

Cadastro de Usuário

Nome:

Pedro H Costa

Email:

pedro@gmail.com

Gênero:

Masculino Feminino

Data de Nascimento:

06/08/1979

Ao confirmar, se tudo estiver correto, aparecerá a tela inicial (index.php) novamente.

Nome	Email	Gênero	Data de Nascimento	Idade	Ações
Pedro H Costa	pedro@gmail.com	Masculino	06/08/1979	41 anos	Editar Remover

Agora falta a programação do formulário de edição, do script que trata a alteração e por fim, o arquivo que trata a exclusão dos dados.

Arquivo formEdit

Abaixo o código comentado do arquivo **formEdit.php**.

```
<?php
require 'init.php';
// pega o ID da URL
$id = isset($_GET['id']) ? (int) $_GET['id'] : null;
// valida o ID
if (empty($id)) {
    echo "ID para alteração não definido";
    exit;
}
// busca os dados do usuário a ser editado
$PDO = db_connect();
```

```

$sql = "SELECT name, email, gender, birthdate FROM users WHERE id =
:id";
$stmt = $PDO->prepare($sql);
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->execute();
$user = $stmt->fetch(PDO::FETCH_ASSOC);
// se o método fetch() não retornar um array, significa que o ID não
corresponde
// a um usuário válido
if (!is_array($user)) {
    echo "Nenhum usuário encontrado";
    exit;
}
?>
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Edição de Usuário</title>
    </head>
    <body>
        <h1>Sistema de Cadastro</h1>
        <h2>Edição de Usuário</h2>
        <form action="edit.php" method="post">
            <label for="name">Nome: </label>
            <br>
            <input type="text" name="name" id="name" value="<?php
echo $user['name'] ?>">
            <br><br>
            <label for="email">Email: </label>
            <br>
            <input type="text" name="email" id="email" value="<?php
echo $user['email'] ?>">
            <br><br>
            Gênero:
            <br>
            <input type="radio" name="gender" id="gender_m" value="m"

```

```

<?php if ($user['gender'] == 'm'): ?>
    checked="checked" <?php endif; ?>>
    <label for="gener_m">Masculino </label>
    <input type="radio" name="gender" id="gender_f" value="f"
<?php if ($user['gender'] == 'f'): ?>
    checked="checked" <?php endif; ?>>
    <label for="gener_f">Feminino </label>
    <br><br>
    <label for="birthdate">Data de Nascimento: </label>
    <br>
    <input type="text" name="birthdate" id="birthdate"
placeholder="dd/mm/YYYY"
value="<?php echo dateConvert($user['birthdate']) ?>">
<br><br>
    <input type="hidden" name="id" value="<?php echo $id ?>">
    <input type="submit" value="Alterar">
</form>
</body>
</html>

```

Arquivo edit

Agora o arquivo que trata a alteração, **edit.php**.

```

<?php
require_once 'init.php';
// resgata os valores do formulário
$name = isset($_POST['name']) ? $_POST['name'] : null;
$email = isset($_POST['email']) ? $_POST['email'] : null;
$gender = isset($_POST['gender']) ? $_POST['gender'] : null;
$birthdate = isset($_POST['birthdate']) ? $_POST['birthdate'] : null;
$id = isset($_POST['id']) ? $_POST['id'] : null;
// validação (bem simples, mais uma vez)
if (empty($name) || empty($email) || empty($gender) ||
empty($birthdate))
{
    echo "Volte e preencha todos os campos";
}

```

```

    exit;
}
// a data vem no formato dd/mm/YYYY
// então precisamos converter para YYYY-mm-dd
$isoDate = dateConvert($birthdate);
// atualiza o banco
$PDO = db_connect();
$sql = "UPDATE users SET name = :name, email = :email,"
       . " gender = :gender, birthdate = :birthdate WHERE id = :id";
$stmt = $PDO->prepare($sql);
$stmt->bindParam(':name', $name);
$stmt->bindParam(':email', $email);
$stmt->bindParam(':gender', $gender);
$stmt->bindParam(':birthdate', $isoDate);
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
if ($stmt->execute())
{
    header('Location: index.php');
}
else
{
    echo "Erro ao alterar";
    print_r($stmt->errorInfo());
}

```

Arquivo delete

E para finalizar o aplicativo, o código comentado do script que executa a exclusão dos dados. O arquivo **delete.php**.

```

<?php
require_once 'init.php';
// pega o ID da URL
$id = isset($_GET['id']) ? $_GET['id'] : null;
// valida o ID
if (empty($id))
{

```

```
echo "ID não informado";
exit;
}
// remove do banco
$PDO = db_connect();
$sql = "DELETE FROM users WHERE id = :id";
$stmt = $PDO->prepare($sql);
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
if ($stmt->execute())
{
    header('Location: index.php');
}
else
{
    echo "Erro ao remover";
    print_r($stmt->errorInfo());
}
```

Exercícios de fixação

1. Como atividade final, você deve produzir uma aplicação CRUD para o banco de dados criado no capítulo passado, onde temos o banco Livraria e a tabela Livros, como base para sua atividade.
2. Desenvolva a atividade como foi apresentado no exemplo deste capítulo, ou seja, separando os arquivos para inclusão, alteração, exclusão e consulta de dados.

Nota: Todos os exercícios deste livro estão disponíveis no Github. Veja no último capítulo do livro.

Considerações Finais

Iniciamos o livro falando sobre o enfoque. Neste contexto apresentamos a linguagem PHP, o histórico da linguagem, seus recursos e preparamos o ambiente para o trabalho, tanto no Sistema Operacional Linux como no Windows.

Em seguida apresentamos o conceito de cliente e servidor, o processo de *request e response* (requisição e resposta), ou seja, como o PHP processa e renderiza em HTML os resultados de seu processamento.

Toda linguagem de programação trata os dados como elementos importantes para a representação das informações a serem processadas. Por este motivo, logo no início apresentamos os conceitos de dados, tipos de dados, constantes e variáveis. Ainda, sobre esse último, foi demonstrado como funciona o escopo destas informações.

No capítulo seguinte apresentamos os operadores aritméticos, lógicos e relacionais, binários e de atribuição.

Um capítulo que merece atenção futura do leitor (aluno) é o de funções, pois neste material não foi abordado o paradigma de Orientação a Objetos. Então as funções funcionam como blocos de códigos separados, que podem ser utilizadas por mais arquivos, usando os comandos include ou require, abordados ao longo do material.

Não menos importante, até mesmo pela forma de trabalho do PHP com arrays associativos, dedicamos um capítulo para tratar do assunto.

Neste material apresentamos uma introdução ao conceito de Sessions, que recomendamos aos alunos a ampliar os estudos deste conceito, entendendo os estados de uma aplicação Web.

Para finalizar foram apresentadas formas de trabalho com o Sistema Gerenciador de Banco de Dados MySQL. Instalamos o servidor no Linux e a ferramenta XAMPP para o Windows. Utilizamos as funções da biblioteca PDO, apresentamos a ferramentas MySQL Workbench, e sua utilização nos dois Sistemas Operacionais. E terminamos o material realizando uma aplicação CRUD (Create, Read, Update e Delete) para gerenciar o cadastro de usuários, cujo cenário foi apresentado no último capítulo.

Repositório de Códigos

Todos os códigos e exemplos deste livro estão disponíveis para download no repositório do Github no seguinte endereço:

https://github.com/maromo71/curso_progweb2



Erros ou faltas

Pedimos encarecidamente a você, caso encontre algum erro, ou ausência de informações neste material, por favor, reporte no seguinte email: etec@maromo.net.

E aproveitamos para agradecer a vocês leitores e também alunos que sem os quais não teríamos realizado este trabalho.

FIM



Muito obrigado,

Professores Marcos e Rodrigo.

Bibliografia

Apache Friends. (n.d.). Retrieved from

https://www.apachefriends.org/pt_br/download.html

Como os navegadores processam os códigos de uma página web? (2015, June 15).

Retrieved from

<https://canaltech.com.br/navegadores/como-os-navegadores-processam-os-codigos-de-uma-pagina-web/>

Hypertext Preprocessor. (n.d.). Retrieved from <http://www.php.net/>

MySQL. (n.d.). Retrieved from <http://www.mysql.com/>

MySQL Workbench 8.0.23. (n.d.). Retrieved from

<https://dev.mysql.com/downloads/workbench/>

PHP: Hypertext Preprocessor. (n.d.). Retrieved from

<https://windows.php.net/download#php-7.4>

Strings - Manual. (n.d.). Retrieved from

https://www.php.net/manual/pt_BR/language.types.string.php#language.types.string.syntax.double

Tecnologia Web para desenvolvedores. (n.d.). Retrieved from

<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/form>

Web service conversation modeling: A cornerstone for e-business automation. (n.d.).

Retrieved from <https://ieeexplore.ieee.org/document/1260703>