INTRODUÇÃO A DOCUMENTOS DINÂMICOS NA WEB EM PHP/SLIM

Prof. Me. Hélio Esperidião

O PHP

PHP é uma sigla recursiva que significa *PHP HyperText Preprocessor.*

O PHP é uma linguagem de código-fonte aberto, muito utilizada na Internet e especialmente criada para o desenvolvimento de aplicativos *Web*

Características do PHP

- É Executado em um servidor web, não no navegador.
- O resultado da execução do php pode ser um código html que é interpretado por um navegador.
- Permite criar paginas dinâmicas
- Manipulação de banco de dados
- Manipulação de arquivos
- Manipulação de *cookies.*
- Sintaxe parecida com C

Vantagens

- É uma linguagem de fácil aprendizado;
- Suporte a um grande número de bancos de dados como: dBase, Interbase, mSQL, mySQL,Oracle, PostgreSQL e vários outros.
- É multiplataforma, tendo suporte aos sistemas Operacionais mais utilizados no mercado;
- Não precisa ser compilado

Composer

O **Composer** é um **gerenciador de dependências para PHP**.

Ele facilita muito o trabalho do desenvolvedor ao lidar com bibliotecas externas que seu projeto precisa, garantindo que tudo funcione de forma compatível.

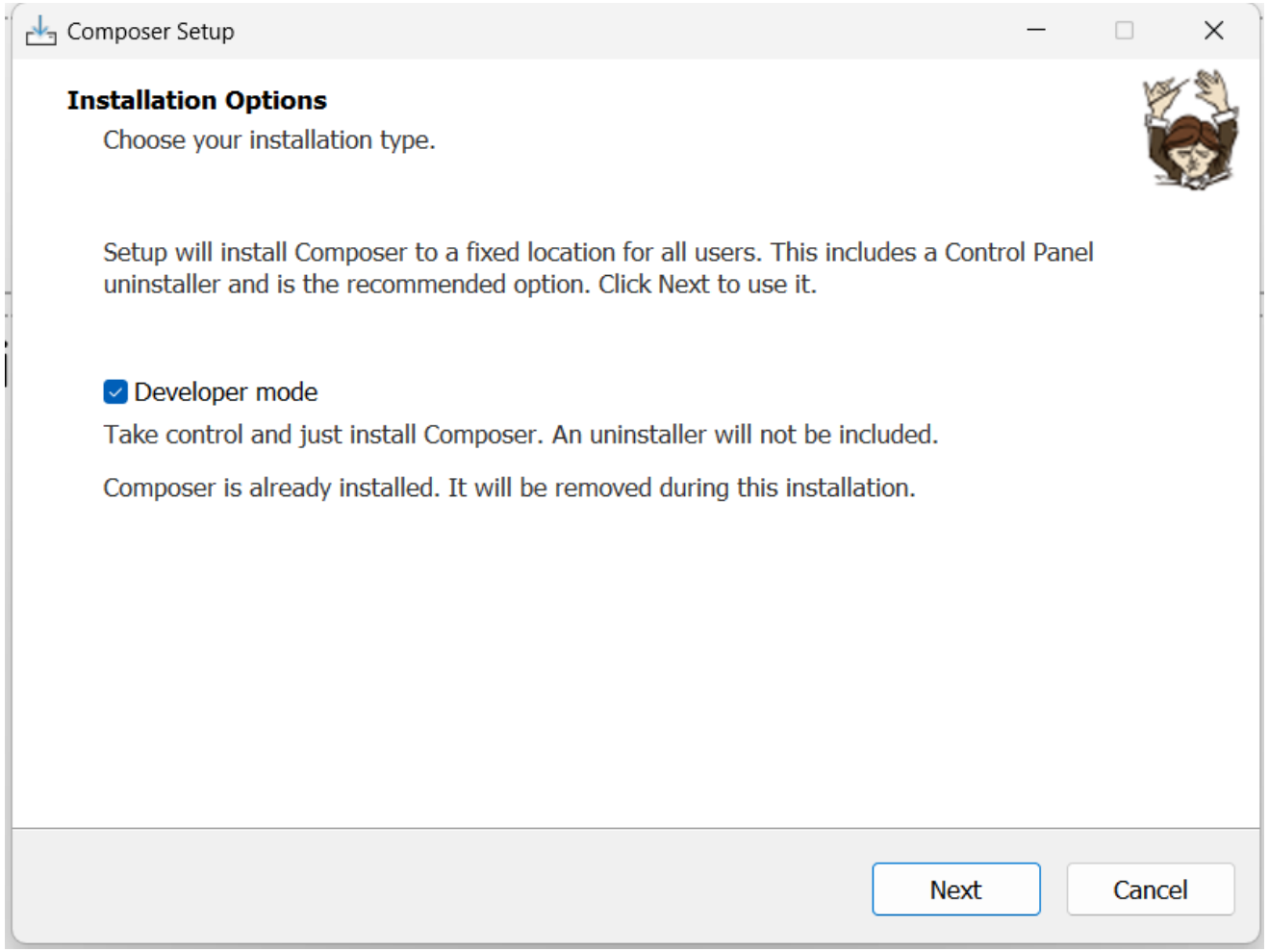
Por que usar

- Evita baixar manualmente bibliotecas externas.
- Mantém seu projeto organizado e fácil de atualizar.
- Facilita a colaboração em equipe, já que qualquer desenvolvedor pode rodar composer install e ter tudo pronto.

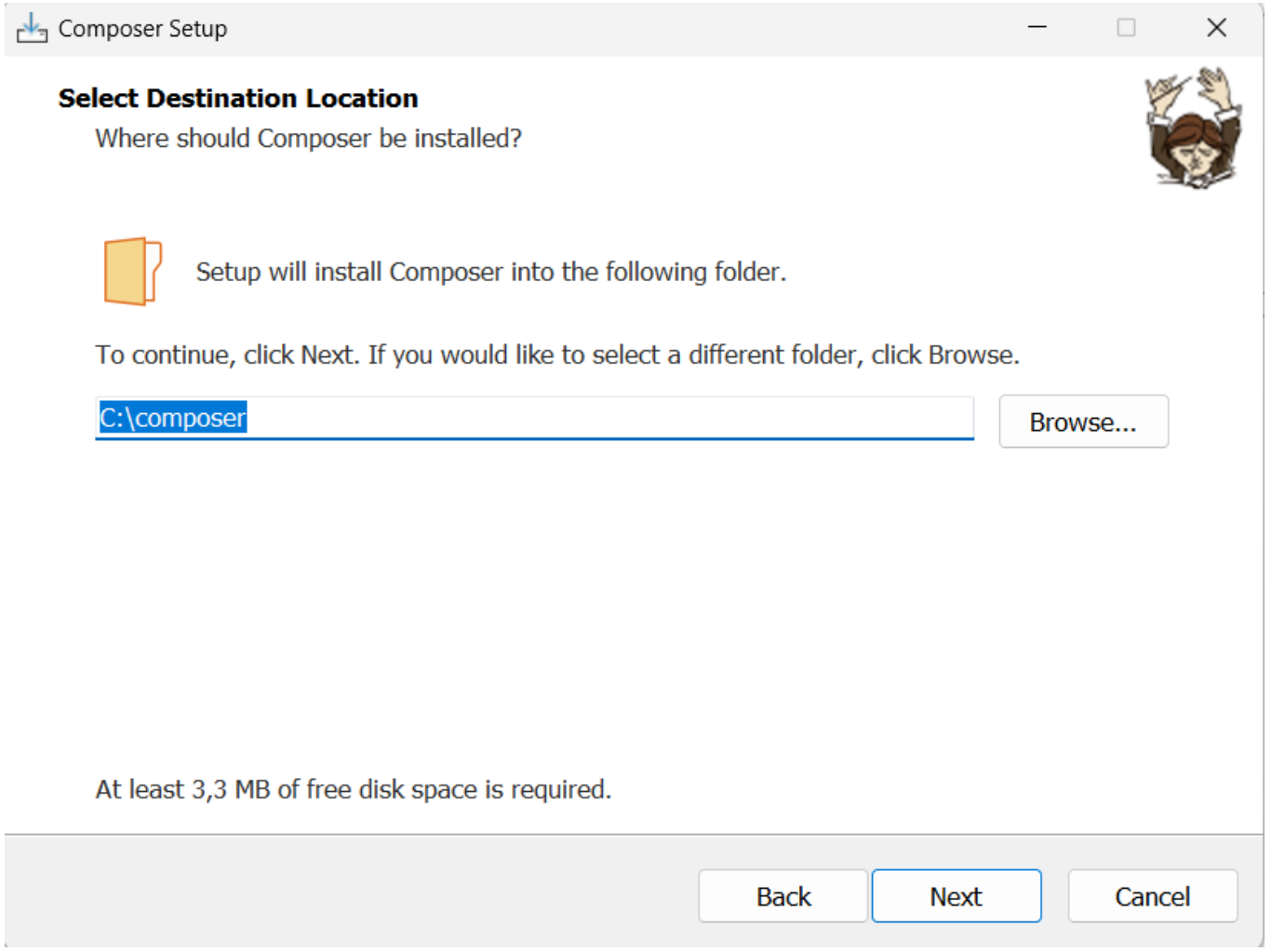
Instalar o composer

- https://getcomposer.org/download/
  - https://getcomposer.org/Composer-Setup.exe

Selecione a opção “Developer Mode”

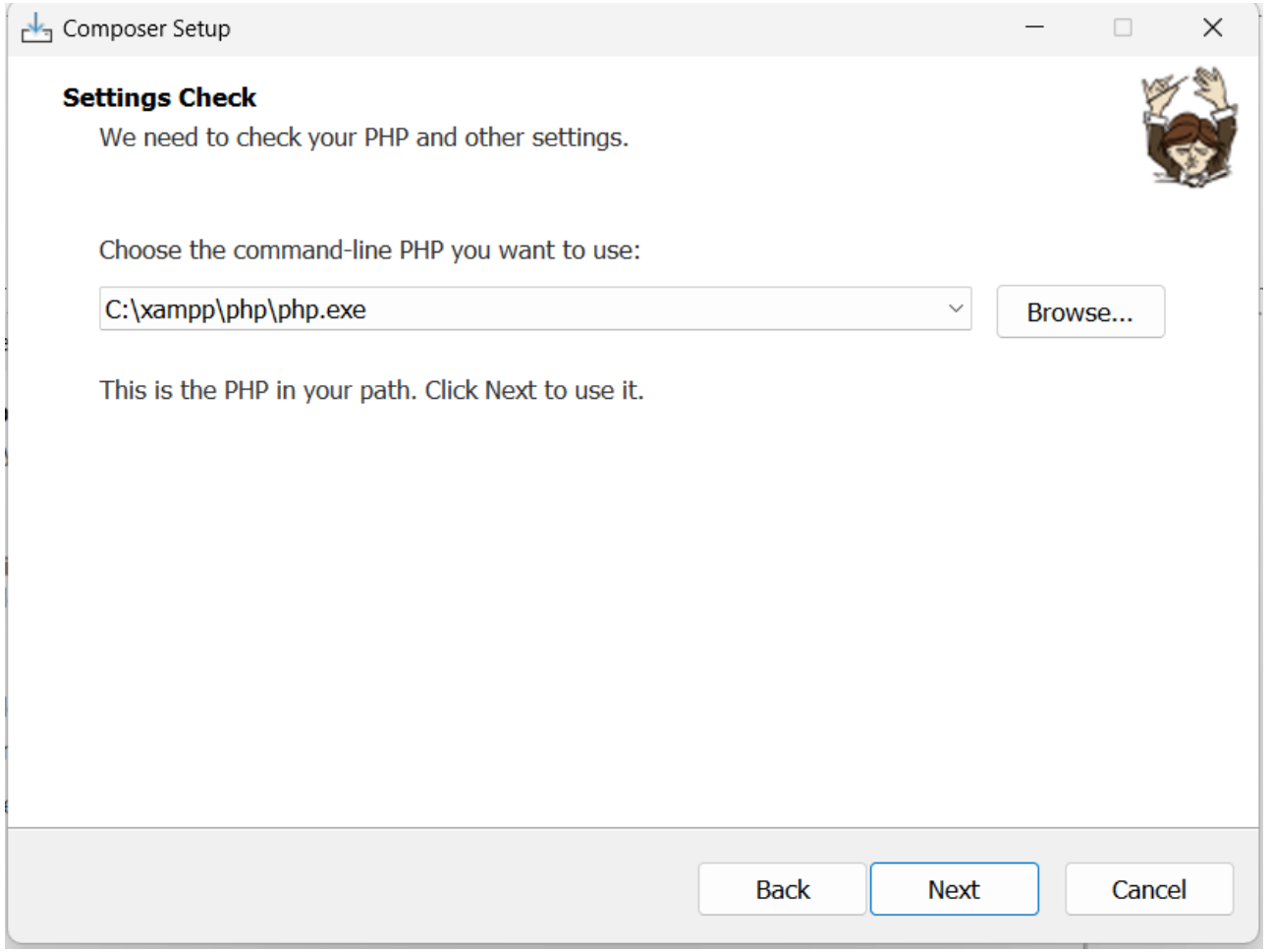


Não modifique a pasta padrão

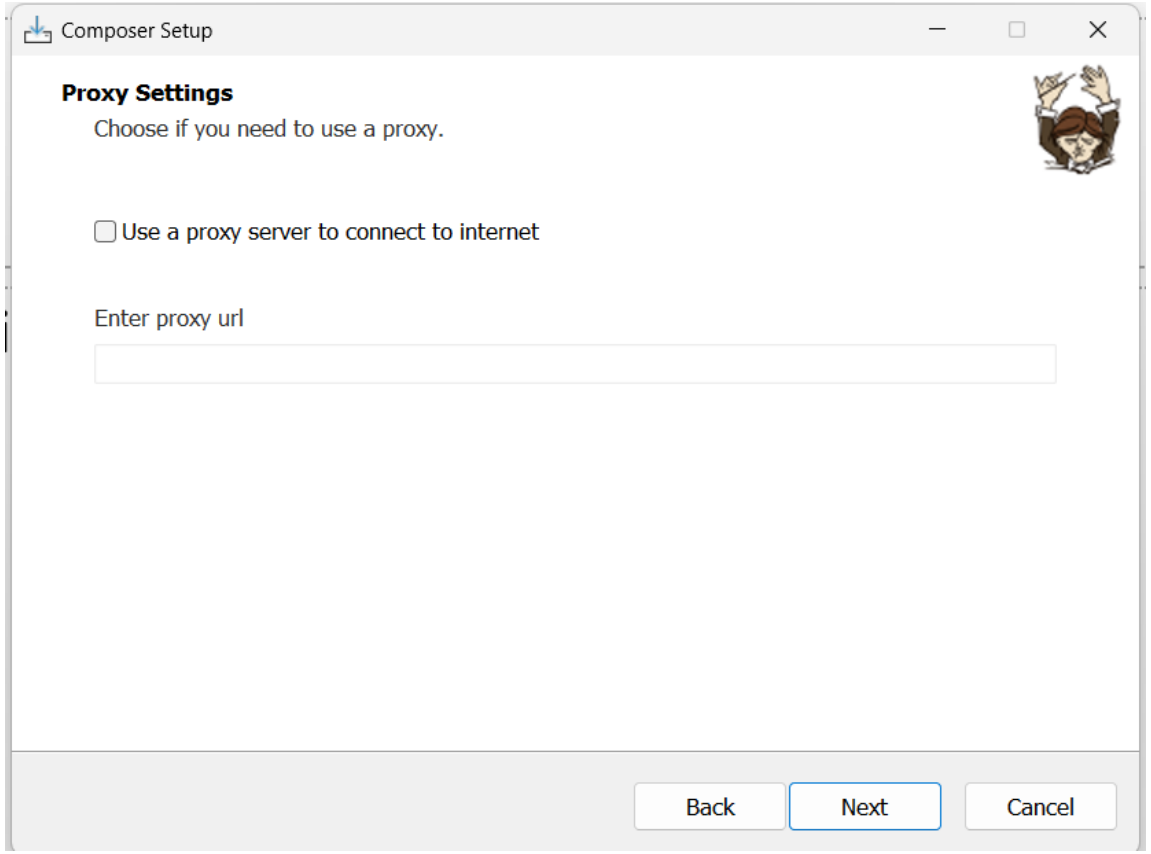


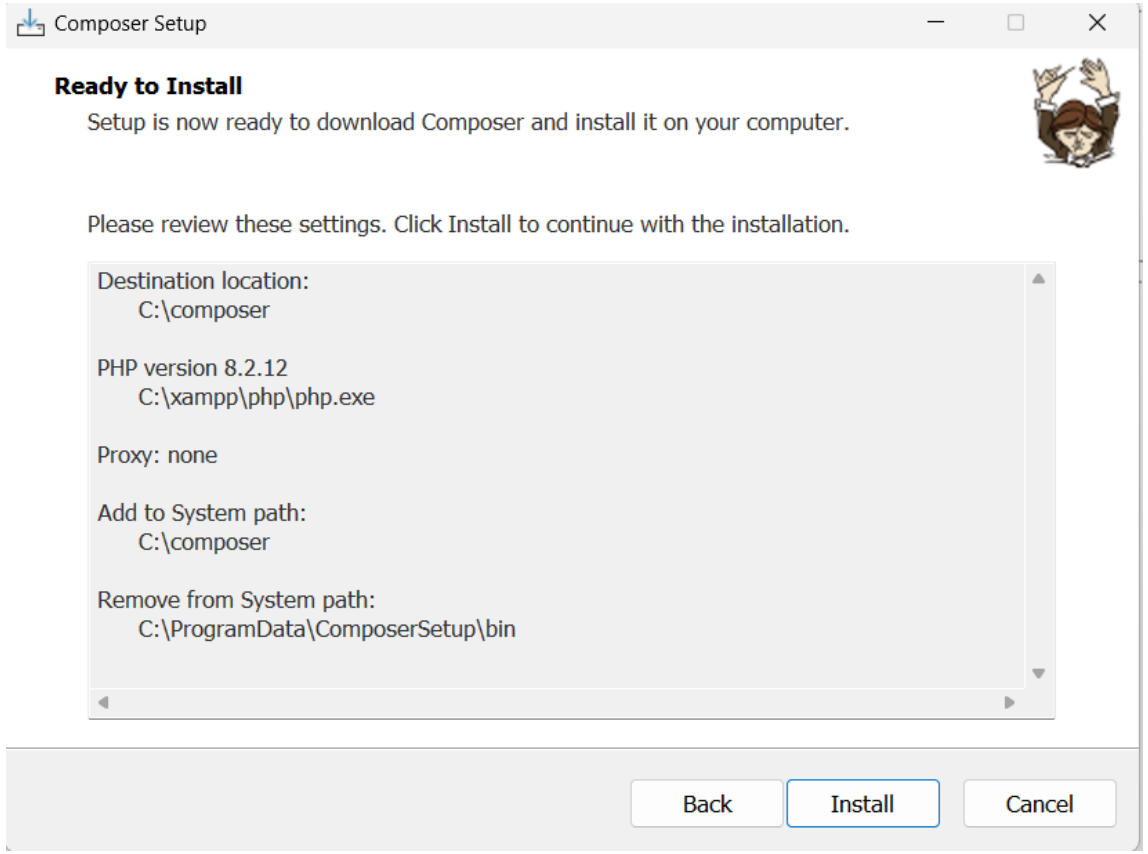
php.exe

- O composer tratalhara em conjunto com o php instalado com com o xampp.
- Por isso é importante não modificar o local padrão de instalação do xampp.



NEXT/install



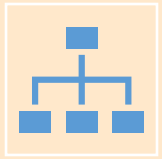


Importante

- Antivírus podem gerar falhas no processo de instalação do composer
  - Desative durante a instalação e uso.
- Firewall
  - Também pode impedir o funcionamento correto do composer.
  - Desative durante a instalação e uso.

FrameWork

Um framework é, basicamente, um conjunto estruturado de códigos, bibliotecas e regras que fornece uma base para desenvolver aplicações de forma mais rápida e organizada.



Ele não é um programa em si, mas sim uma “armação” pronta que você pode usar para construir software seguindo certos padrões.

Slim Framework

- É um **micro-framework para PHP** que serve para **desenvolver aplicações web simples ou APIs (interfaces de programação)** de forma leve e rápida.
- Ele implementa funções essenciais para tratar **requisições HTTP**, definir **rotas** (endereços da aplicação) e gerar **respostas**, sem trazer toda a estrutura “pesada” de frameworks maiores.
- O **Slim Framework é um micro-framework PHP leve e flexível para criar aplicações web e APIs de forma rápida e organizada**, deixando o desenvolvedor adicionar apenas o que for realmente necessário.

Rotas na web

Uma rota em web é um “caminho” que será “chamado” por uma aplicação

- Este caminho é responsável por **um** ou mais serviços.

Cada rota pode ter uma ou mais funções.

Ao receber uma chamada a rota faz todo o processamento necessário para retornar os dados que foram solicitados.

Roteamento

O Roteamento refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) e um método de solicitação HTTP específico (GET, POST, etc).

Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.

Exemplo

- **(URI):** http://localhost:8080/retangulos/areas/5/10
  - **Rota**: /retangulos/areas/5/10
  - **Função**: Retorna a área de um retângulo com base 5 altura 10
- **(URI):** http://localhost:8080/retangulos/perimetros/5/10
  - **Rota**: /retangulos/perimetros/5/10
  - **Função**: Retorna o perímetro de um retângulo com base 5 altura 10

O que é o método de envio (HTTP Method)?

- O **método HTTP** indica **a intenção da requisição** enviada do cliente para o servidor.
- Exemplos comuns:
  - **GET** → buscar informações
  - **POST** → enviar/criar algo
  - **PUT** → atualizar algo
  - **DELETE** → remover algo

Exemplo de uso

- GET /retangulos/areas/5/10
  - GET: quero obter um dado
  - /retangulos/areas/5/10 é o caminho do recurso
  - Resultado retorna a área do retângulo (5 × 10)
  - **GET** é usado apenas para leitura
  - Não altera nada no servidor
  - Só calcula e retorna o valor

Métodos de envio de requisições na web

Método	**Uso principal**	**Exemplo**
GET	Buscar dados	Obter área do retângulo
POST	Criar dados	Criar um retângulo
PUT	Atualizar dados	Atualizar dimensões
DELETE	Remover dados	Apagar um retângulo

> “O método de envio diz **o que o cliente quer fazer**, enquanto a rota diz **onde isso será feito**.”

Desmistificando

Termo	**O que é**
URI	Endereço completo
Rota	Caminho dentro da aplicação
Endpoint	Combinação de rota + método HTTP que executa uma ação

Endpoint

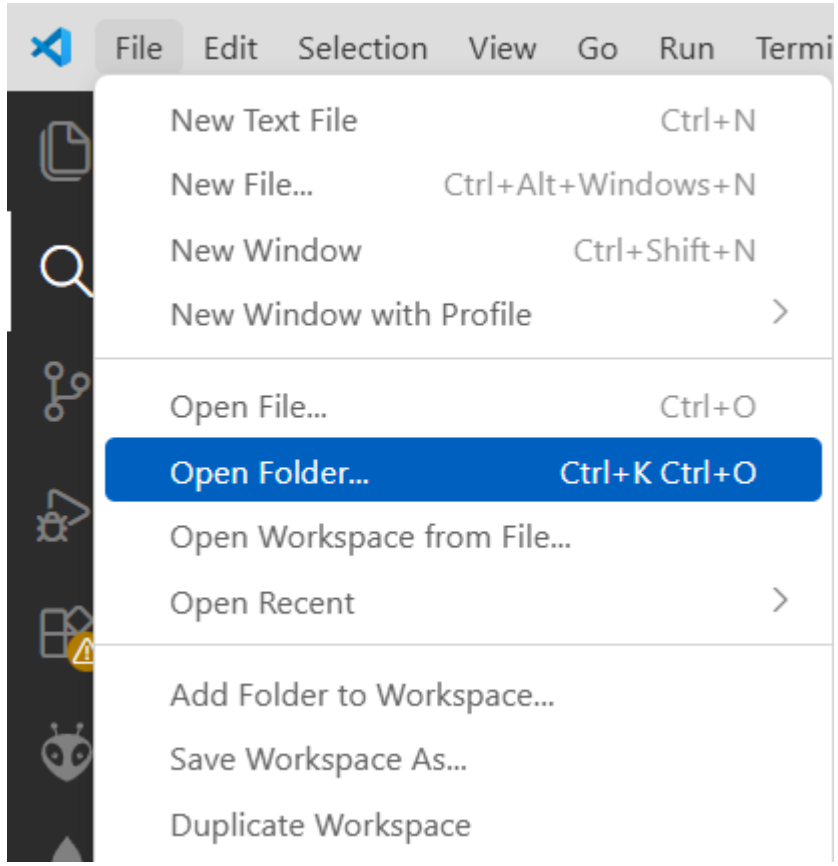
- Um **endpoint** é o **ponto final de acesso a um recurso**.
- Ele normalmente envolve:
  - URI completa
  - Método HTTP (GET, POST, PUT, DELETE...)
- Por exemplo:
  - GET http://localhost:8080/retangulos/areas/5/10
- Isso é um **endpoint**, porque define:
  - O endereço
  - O método (GET)
  - O comportamento (retornar área)

Documentação comum

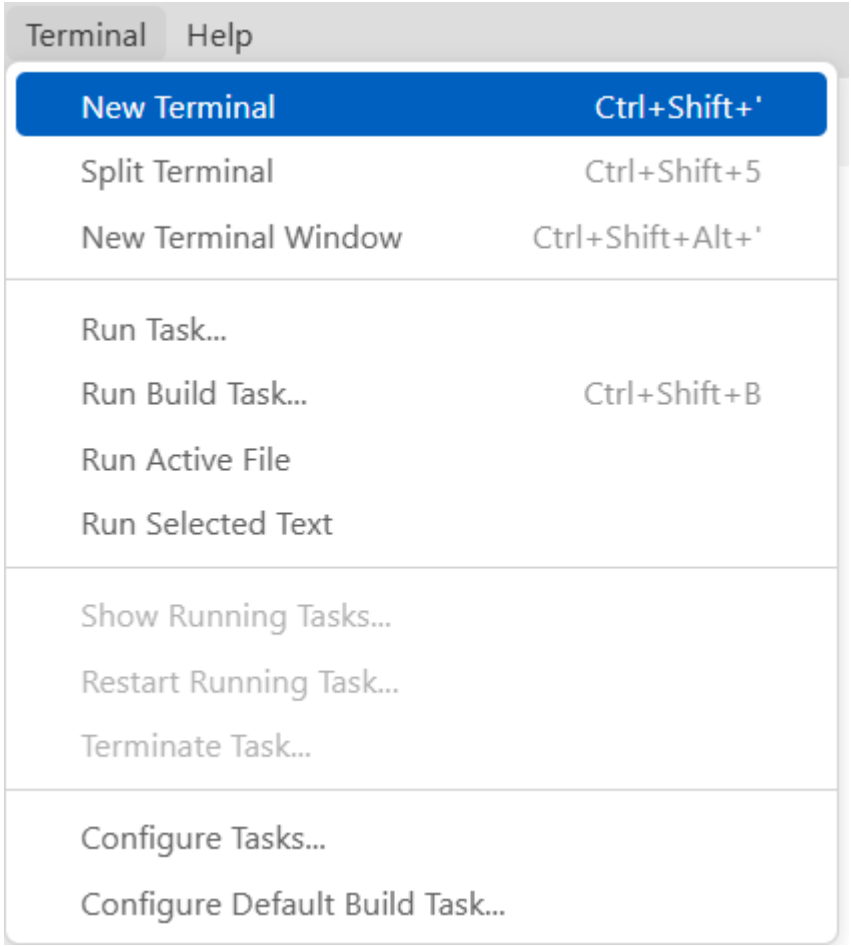
- Os dados enviados são delimitados por { }
- É suprimida a url:
  - GET /retangulos/areas/{base}/{altura}
    GET /retangulos/perimetros/{base}/{altura}

Olá mundo em php/slim

Abra uma pasta no visual studio code



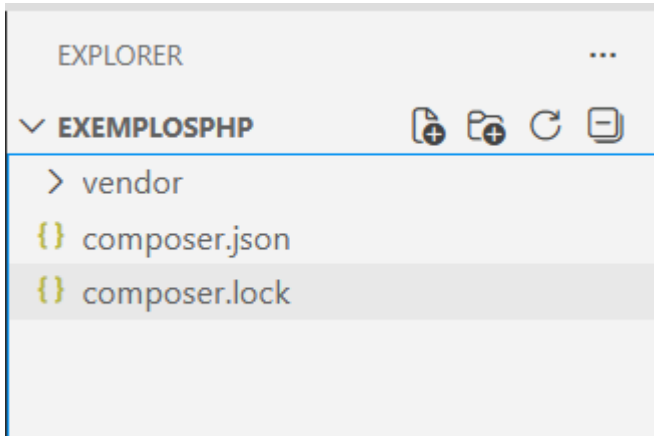
Abra o terminal do Visual studio code



Digite no terminal

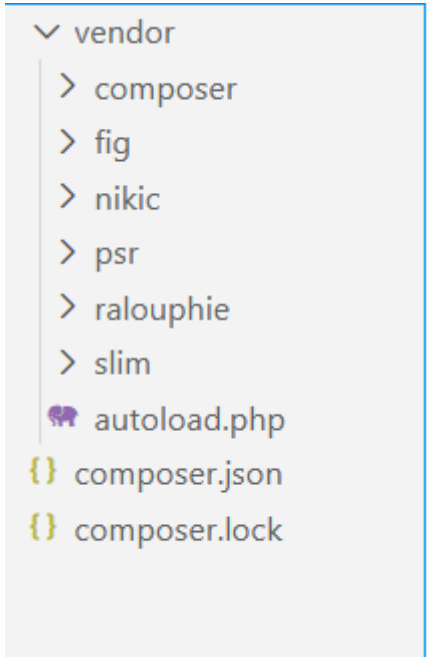
- **composer require slim/slim slim/psr7**

- Será criada a pasta vendor
- E os arquivos compser.json e composer.lock



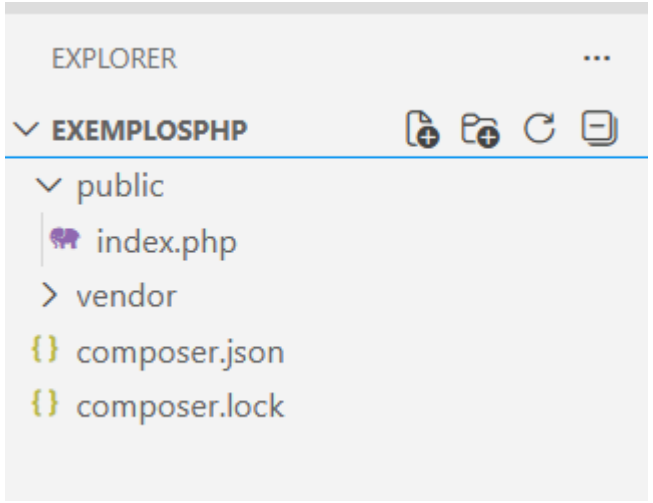
Estrutura de arquivos

- **Pasta vendor/**
  - A pasta vendor contém **todas as dependências do projeto**.
  - Ou seja:
    - Código do **Slim**
    - Código do **PSR-7**
    - Outras bibliotecas que eles usam internamente
- O arquivo vendor/autoload.php
  - Ela é basicamente:
  - “A pasta onde ficam as bibliotecas que seu projeto usa”
    - Importante: Essa pasta **não deve ser alterada manualmente**.



Crie:

- Pasta: **public**
  - Arquivo: index.php



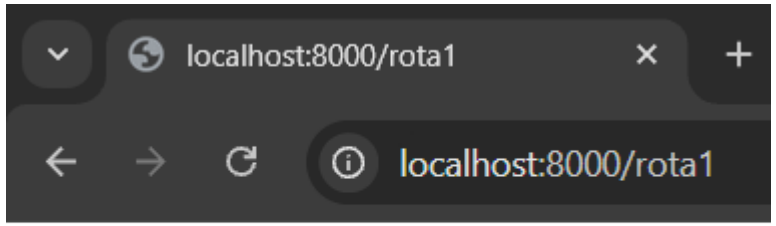

```php
<?php
// Carregar autoload do Composer
require __DIR__ . '/../vendor/autoload.php';


// Criar aplicação Slim
$app = \Slim\Factory\AppFactory::create();



// Rota 1: Ola mundo
$app->get('/rota1', function ($request, $response) {
    $response->getBody()->write('Olá Mundo!');
    return $response;
});


// Rota 1: Ola web
$app->get('/rota2', function ($request, $response) {
    $response->getBody()->write('Olá web!');
    return $response;
});


// Executar a aplicação
$app->run();
```

Entendendo o básico

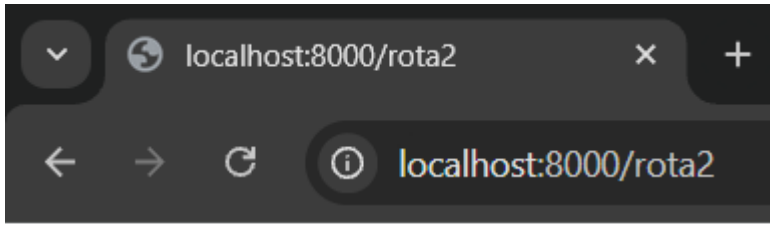
- public/index.php
  - É o arquivo que **inicia a aplicação**
- É o arquivo que **inicia a aplicação**
  - public/index.php
    - É onde você define as (rotas), tipo:
      - /usuarios
      - /teste
      - /teste/teste/teste
      - /api/v1/clientes
- Quais rotas foram definidas no exemplo anterior?

- Testando as rotas

Digite no terminal
php -S localhost:8000 -t public/



Olá Mundo!



Olá web!

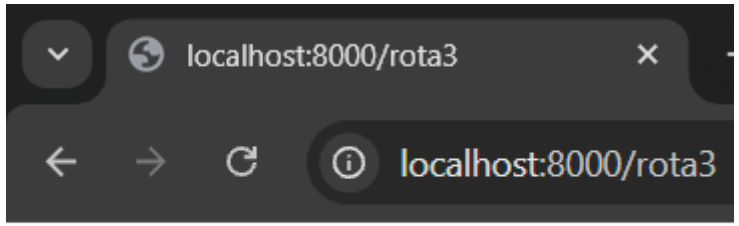
Caso não reconheça o comando:
c:\xampp\php\php.exe -S localhost:8000 -t public/

Variáveis em php

- Regras para variáveis PHP:
  - Uma variável começa com o $sinal, seguido pelo nome da variável
  - Um nome de variável deve começar com uma letra ou o caractere de sublinhado
  - Um nome de variável não pode começar com um número
  - Um nome de variável pode conter apenas caracteres alfanuméricos e sublinhados (Az, 0-9 e _)
  - Os nomes das variáveis diferenciam maiúsculas de minúsculas ($a e $A são duas variáveis diferentes)

- No PHP, uma variável começa com o **$** seguido do nome da variável:
- *O tipo é definido em tempo de execução.*

$x = 5;



o valor de x é igual a **1**

```php
// Rota 3: variáveis
$app->get('/rota3', function ($request, $response) {
    $x = 1;
    // Exibe o texto "valor da variável = 1."
    // O PHP interpreta $x dentro da string porque usamos aspas duplas
    $y = "<p> o valor de x é igual a <b>$x</b></p>";


    $response->getBody()->write($y);
    return $response;
});
```

Operadores Matemáticos

Operador	Função
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da Divisão)
++	Adiciona 1 em uma variável numérica
--	Subtrai 1 em uma variável numérica
+=	adiciona o valor à direita à variável ou propriedade à esquerda e atribui o resultado à variável ou à propriedade à esquerda
-=	subtrai o valor à direita à variável ou propriedade à esquerda e atribui o resultado à variável ou à propriedade à esquerda
.=	Concatena o valor à direita à variável à esquerda e atribui o resultado à variável à esquerda

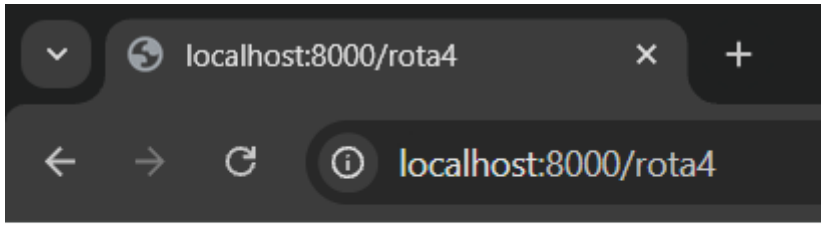
Operadores relacionais

$a == $b	Igual	Verdadeiro (**true**) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (**true**) se $a é igual a $b, e eles são do mesmo tipo.
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (introduzido no PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.

String

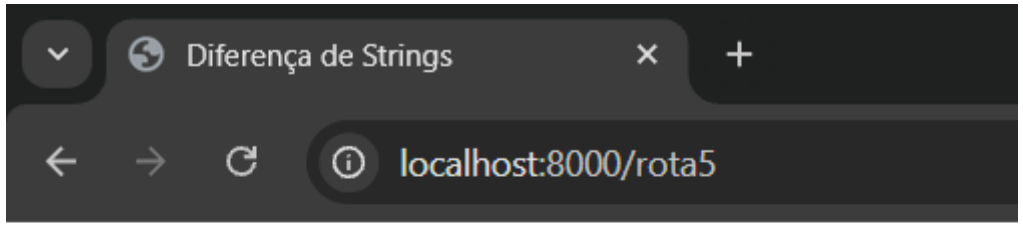
- O tipo de dados STRING é uma cadeia de caracteres alfanuméricos (letras, números e caracteres especiais). O tipo STRING pode ser utilizado de duas maneiras:
- Utilizando aspas simples o valor da variável será exatamente o texto contido entra as aspas.
- Utilizando aspas duplas qualquer variável ou caractere de escape será expandido antes de ser atribuído.

```php
// Rota 4: variáveis
$app->get('/rota4', function ($request, $response)
{
    $x = "helio";
    $y = "esperidião";


    $v1 = "$x $y";
    $v2 = '$x $y';



    $response->getBody()->write($v1);
    $response->getBody()->write("<br>");
    $response->getBody()->write($v2);
    return $response;
});
```



helio esperidião
$x $y

```php
$app->get('/rota5', function ($request, $response) {
    $x = "helio";
    $y = "esperidião";

    $v1 = "$x $y";
    $v2 = '$x $y';

    $html = "<!DOCTYPE html>
            <html>
            <head><title>Diferença de Strings</title></head>
            <body>
                <h3>Diferença entre aspas duplas e simples:</h3>
                <p>Com aspas duplas ($v1): <strong>$v1</strong></p>
                <p>Com aspas simples ($v2): <strong>$v2</strong></p>
                <br>
                <p><em>Obs: Aspas duplas interpolam variáveis, aspas simples não.</em></p>
            </body>
            </html>";

    $response->getBody()->write($html);

    // Opcional: Definir cabeçalho HTML explicitamente
    return $response;
});
```



Diferença entre aspas duplas e simples:

Com aspas duplas ($v1): **helio esperidião**

Com aspas simples ($v2): **$x $y**

Obs: Aspas duplas interpolam variáveis, aspas simples não.

String

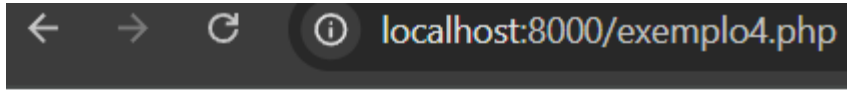
O tipo de dados STRING é uma cadeia de caracteres alfanuméricos (letras, números e caracteres especiais). O tipo STRING pode ser utilizado de duas maneiras:

Utilizando aspas simples o valor da variável será exatamente o texto contido entra as aspas, com exceção de: **\\ e \'.**

Utilizando aspas duplas qualquer variável ou caractere de escape será expandido antes de ser atribuído.

```php
$app->get('/rota6', function ($request, $response) {
    $v1 = 1; // A variável $v1 recebe um valor inteiro
    $v2 = 1.0; // A variável $v2 recebe um valor float (número decimal)
    $v3 = TRUE; // A variável $v3 recebe um valor booleano TRUE
    $v4 = "oi"; // A variável $v4 recebe uma string "oi"



    $v5 = [10, 20, 30]; // A variável $v5 recebe um array
    $v6 = 10e4; // Recebe em notação científica (10 × 10^4 = 100000)
     // Exibe os valores e seus tipos
     $response->getBody()->write( "valor de v1: $v1 - " . gettype($v1) . "<br>");
     $response->getBody()->write( "valor de v2: $v2 - " . gettype($v2) . "<br>");
     $response->getBody()->write( "valor de v3: $v3 - " . gettype($v3) . "<br>");
     $response->getBody()->write( "valor de v4: $v4 - " . gettype($v4) . "<br>");
     $response->getBody()->write( "valor de v5: $v5[2] - " . gettype($v5) . "<br>");
     $response->getBody()->write("valor de v6: $v6 - " . gettype($v6) . "<br>");

    return $response;
});
```

Exemplos de tipos



valor de v1: 1 - integer
valor de v2: 1 - double
valor de v3: 1 - boolean
valor de v4: oi - string
valor de v5: 30 - array
valor de v6: 100000 - double

Envio de dados HTTP

Método POST

- Encapsula os dados e os envia junto do cabeçalho http.

Método GET

- Os dados são enviados por meio da url.

Método GET
Como identificar as variáveis GET na URI?

Uri: http://helioesperidiao.com/rota?v1=valor&v2=valor

No caso:
- **Protocolo**: `http`
- **Domínio**: `helioesperidiao.com`
- **Caminho (rota)** → `/rota`
- **Query string** (parâmetros) → `?v1=valor&v2=valor`

As variáveis estão **depois do ponto de interrogação (?)**

Essa parte é chamada de **query string**.
?v1=valor&v2=valor

Variável	**Valor**
v1	valor
v2	valor

Exemplo google

- Observe os links abaixo, perceba que é enviada a variável **q** para “search”. Search é o serviço web da google que faz a busca.
- O dado é enviado pelo método **GET**

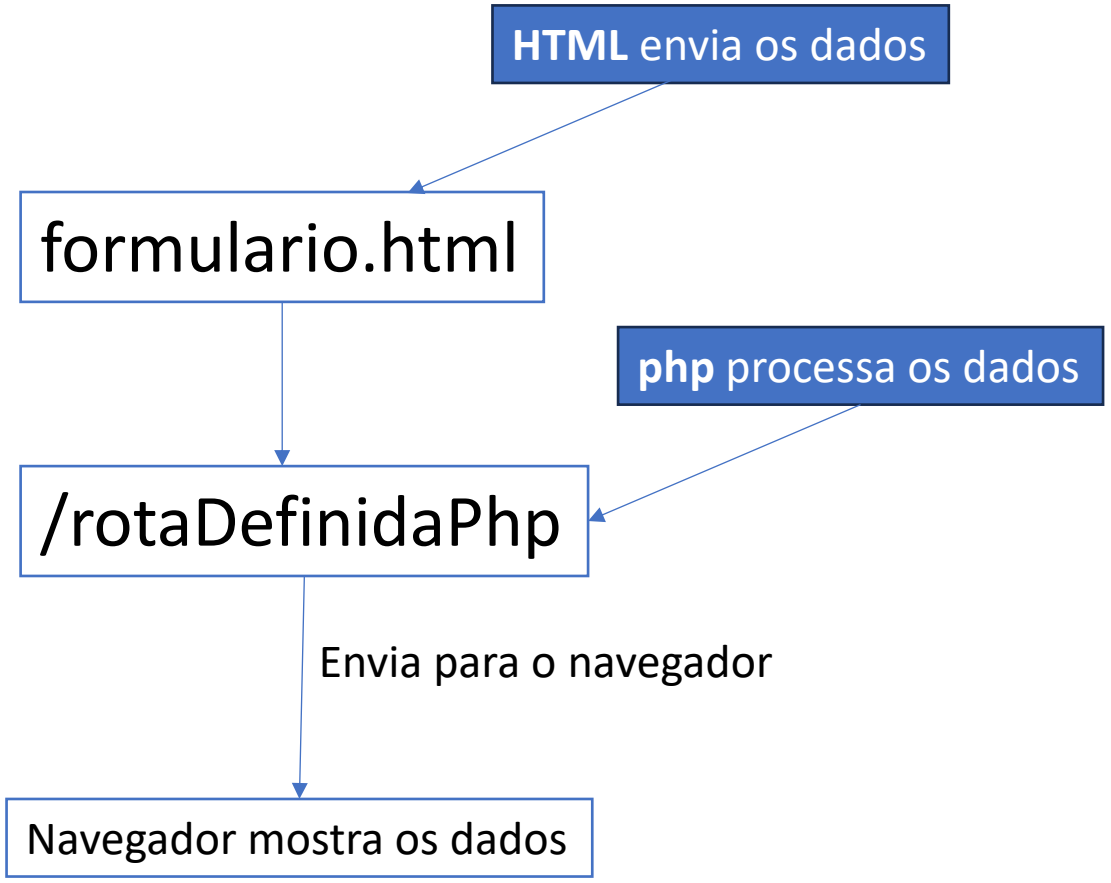
- https://www.google.com/search?q=helioesperidiao
- https://www.google.com/search?q=batataFrita
- https://www.google.com/search?q=Queijo

Método post

- Envio de dados pelo corpo da requisição
  - Quando você usa POST, os dados do formulário não aparecem na **URN** (diferente do GET).
  - Eles são enviados “dentro” da requisição HTTP, no corpo da mensagem.
- Mais seguro para dados sensíveis
  - Por não aparecer na URL, é mais adequado para informações como senhas, CPF, RG, etc.
- Uso no PHP
  - Os dados enviados via POST ficam disponíveis na super global $_POST, que é um array associativo.
  - Cada campo do formulário é acessado pelo nome do input.

Arquitetura de arquivos

- Durante as aulas de c e durante as aulas de lógica para programação os programas trabalhavam com um único arquivo.
- Sites e aplicações para web são constituídos por muitos arquivos. Veja ao lado a arquitetura base para uma aplicação web simples.



Formulários
GET

- Permitem o envio de dados para servidores.
- Permitem o envido de dados para scripts dentro da mesma página
- Action: Para onde os dados serão enviados
- Method: Forma de Envio de dados

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulários</title>
</head>
<body>
  <form action= "/rotaDestino" method="get">



  </form>

</body>
</html>
```

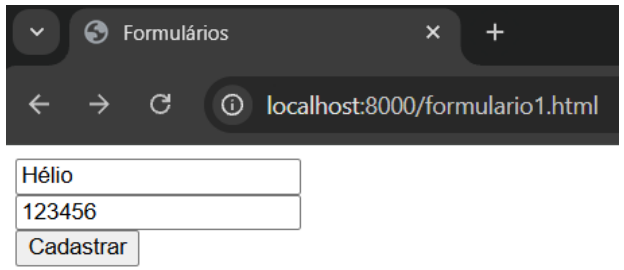
Caixa de texto

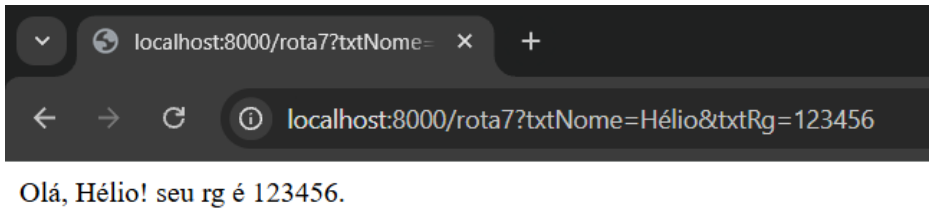
Formulario1.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Formulários</title>
</head>
<body>
 <form action="/rota7 " method="get">
  <input type="text" name="txtNome"  placeholder="Nome"><br>
  <input type="text" name="txtRg" placeholder="RG"><br>
  <input type="submit" value="Cadastrar">
 </form>
</body>
</html>
```

/rota7

```
$app->get('/rota7', function ($request, $response) {
    // Obter todos os parâmetros GET
    $queryParams = $request->getQueryParams();

    // Acessar parâmetros individuais
    $nome = $queryParams['txtNome'] ?? 'Não informado';
    $txtRg = $queryParams['txtRg'] ?? 'Não informado';

    $resultado = "Olá, $nome! seu rg é $txtRg.";

    $response->getBody()->write($resultado);
    return $response;
});
```





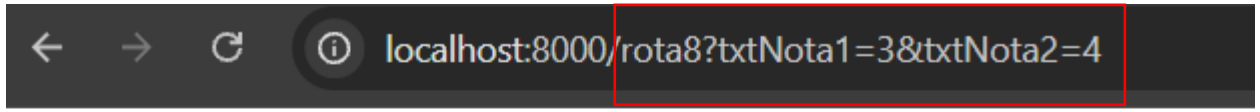
Como identificar as variáveis?

formularioRota8.html

```html
<!DOCTYPE html>

<html>

<head>

  <title>Formulários</title>

</head>

<body>

  <form action="/rota8" method="get">

    <input type="text" name="txtNota1"><br>

    <input type="text" name="txtNota2"><br>

    <input type="submit" value="Calcular Media">

  </form>

</body>

</html>
```

/rota8

```php
// Rota 8: Cálculo de média escolar (versão simples)
$app->get('/rota8', function ($request, $response) {
    $queryParams = $request->getQueryParams();
    $n1 = $queryParams ['txtNota1'] ?? 0;
    $n2 =  $queryParams ['txtNota2'] ?? 0;
    // Calcula a média
    $media = ($n1 + $n2) / 2;
    // Monta o resultado
    $resultado = sprintf("nota final: %.2f ", $media);
    // Verifica aprovação
    if ($media >= 6) {
        $resultado .= "aprovado";
    } else {
        $resultado .= "reprovado";
    }
    // Escreve o resultado
    $response->getBody()->write($resultado);
    return $response;
});
```



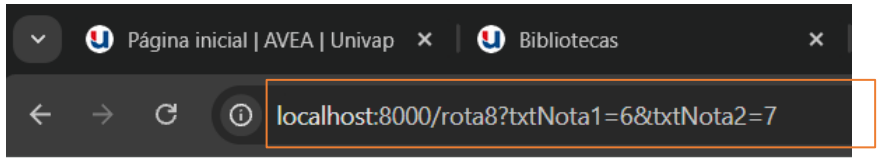
nota final: 3.50 reprovado

Não é necessário um formulário para trabalhar com o método get

Altere direto na uri do navegador:

http://localhost:8080/rota8?**txtNota1**=6&**txtNota2**=7

http://localhost:8080/rota8?**txtNota1**=8&**txtNota2**=9



nota final: 6.50 aprovado

nota final: 8.50 aprovado

```php
$app->get('/rota8', function ($request, $response) {
    $queryParams = $request->getQueryParams();
    $n1 = $queryParams ['txtNota1'] ?? 0;
    $n2 =  $queryParams ['txtNota2'] ?? 0;
    $media = ($n1 + $n2) / 2;
    $resultado = sprintf("nota final: %.2f ", $media);
    if ($media >= 6) {
        $resultado .= "aprovado";
    } else {
        $resultado .= "reprovado";
    }
    $response->getBody()->write($resultado);
    return $response;
});
```

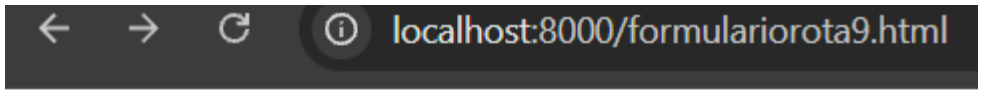
Password

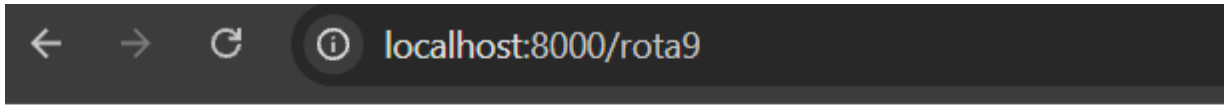
formularioRota9.html

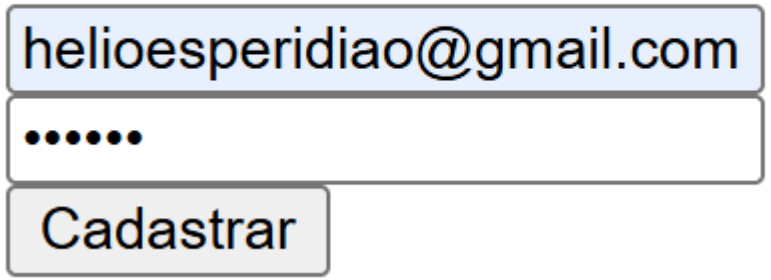
```html
<!DOCTYPE html>
<html>
<head>
  <title>Formulários</title>
</head>
<body>
  <form action="/rota9" method="post">
    <input type="email" name="txtEmail"><br>
    <input type="password" name="txtSenha"><br>
    <input type="submit" value="Cadastrar">
  </form>

</body>
</html>
```

/rota9

```php
// Rota 9: recebe email e senha
$app->post('/rota9', function ($request, $response) {
    // Captura os dados enviados pelo formulário via POST
    $dados = $request->getParsedBody();

    $email = $dados['txtEmail'] ?? '';
    $senha = $dados['txtSenha'] ?? '';
    // Monta a resposta
    $resultado = "[$email] [$senha]";
    // Escreve a resposta
    $response->getBody()->write($resultado);
    return $response;
});
```







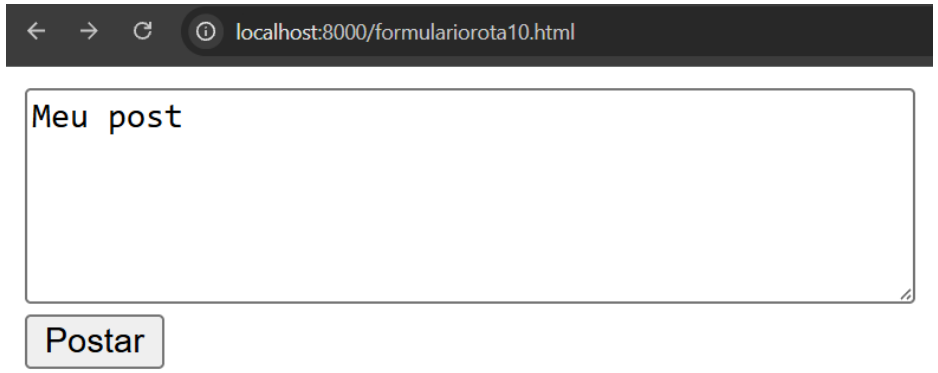
[helioesperidiao@gmail.com] [qweqwe]

Onde estão as variáveis na rota?

<textarea>

formularioRota10.html

```html
<!DOCTYPE html>
<html>
<head>
  <title>Formulários</title>
</head>
<body>
  <form action="/rota10" method="get">
    <textarea cols="45" rows="5" name="txtPost"></textarea><br>
    <input type="submit" value="Postar">
  </form>

</body>
</html>
```



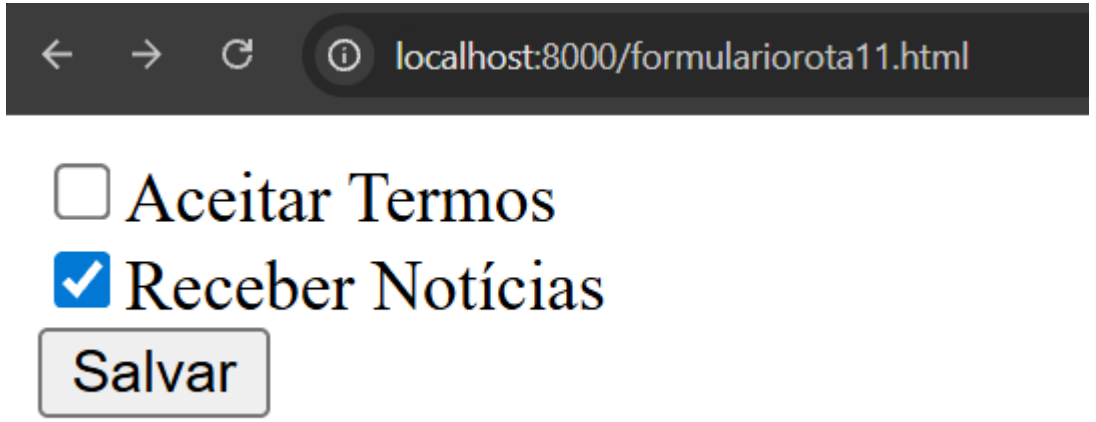
/rota10

```php
// Rota 10: Recebe os dados enviados de um <textArea>
$app->get('/rota10', function ($request, $response) {
    $queryParams = $request->getQueryParams();
    $dadoRecebido = $queryParams ['txtPost'] ?? "";
      // Monta o resultado
    $resultado = "Foi enviado:  $dadoRecebido";
    // Escreve o resultado
    $response->getBody()->write($resultado);
    return $response;
});
```

checkbox

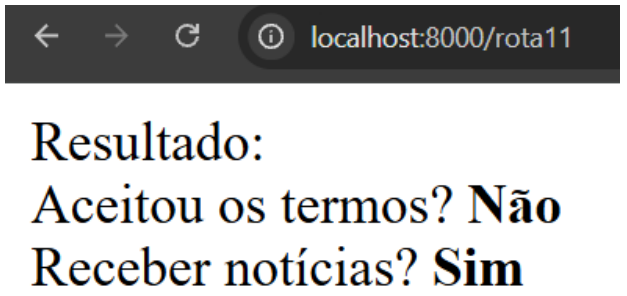
formularioRota11.html

```html
<!DOCTYPE html>
<html>
<body>
 <form action="/rota11" method="post">
   <input type="checkbox" name="chkTermos">Aceitar Termos<br>
   <input type="checkbox" name="chkNoticias">Receber Notícias
  <br>
  <input type="submit" value="Salvar">
 </form>
</body>
</html>
```



/rota11

```php
// Rota 11: Recebe os dados enviados de 2 componentes do tipo checkbox
$app->post('/rota11', function ($request, $response) {
    $dados = $request->getParsedBody();
    // Verifica se os checkboxes foram marcados
    $chkTermos = $dados['chkTermos'] ?? false;
    $chkNoticias = $dados['chkNoticias'] ?? false;
    // Monta o resultado
    $resultado = "Resultado:<br>";
    $resultado .= "Aceitou os termos? ";
    $resultado .= $chkTermos ? "<b>Sim</b><br>" : "<b>Não</b><br>";
    $resultado .= "Receber notícias? ";
    $resultado .= $chkNoticias ? "<b>Sim</b><br>" : "<b>Não</b><br>";
    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;

});
```



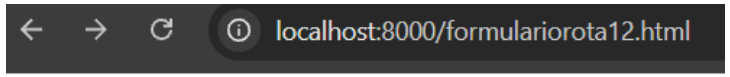
radio

paginaDestino8.php

formularioRota12.html

```
<!DOCTYPE html>
<html>
<body>
<form action="/rota12" method="post">
  <p>Eu sei mais sobre:</p>
  <input type="radio" name="rdoFavorito" value="1"> HTML  <br>
  <input type="radio" name="rdoFavorito" value="2"> CSS  <br>
  <input type="radio" name="rdoFavorito" value="3"> Python  <br>
  <input type="submit" value="Enviar">
</form>
</body>
</html>
```

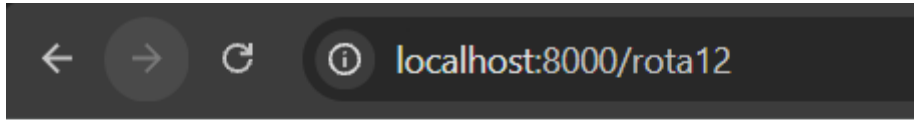
```
// Rota 12: Recebe os dados enviados de 3 componentes do
tipo radio
$app->post('/rota12', function ($request, $response) {
    $dados = $request->getParsedBody();
    // Verifica se os checkboxes foram marcados
    $favorito = $dados['rdoFavorito'] ?? "Não selecionado";
    // Monta o resultado
    $resultado = "Favorito:<br> $favorito";
    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;
});
```



Eu sei mais sobre:

- ○ HTML
- ◉ CSS
- ○ Python

Enviar



Favorito:
2

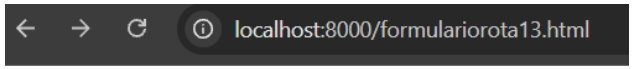
Caixa de seleção

formularioRota13.html

```html
<!DOCTYPE html>
<html>
<body>
<form action="/rota13" method="post">
  <p>Eu sei mais sobre:</p>
  <select name="cboLinguagem">
      <option value="1">HTML</option>
      <option value="2">CSS</option>
  </select> <br><br><br>
  <input type="submit" value="Enviar">
</form>
</body>
</html>
```

/rota13

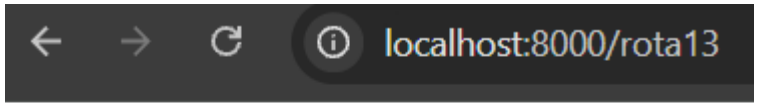
```php
// Rota 13: Recebe os dados enviados de uma caixa de seleção
$app->post('/rota13', function ($request, $response) {
    $dados = $request->getParsedBody();
    // Verifica se os checkboxes foram marcados
    $linguagem = $dados['cboLinguagem'] ?? "Não selecionado";
    // Monta o resultado
    $resultado = "Favorito:<br> $linguagem";
    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;

});
```



Eu sei mais sobre:

HTML

Enviar

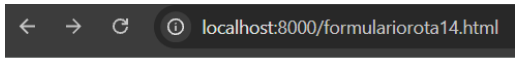


Favorito:
1

Caixa de lista

formularioRota14.html

```
<!DOCTYPE html>
<html>
<body>
    <form action="/rota14" method="post">
        <p>Eu sei mais sobre:</p>
        <select name="lstLinguagem" size="4">
            <option value="a">HTML</option>
            <option value="b">CSS</option>
        </select> <br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```



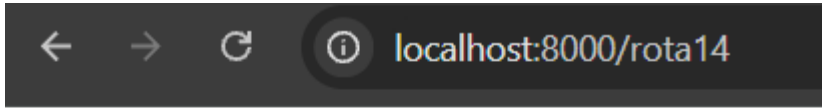
Eu sei mais sobre:

HTML
CSS

Enviar

/rota14

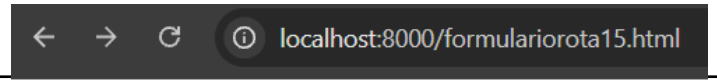
```
// Rota 14: Recebe os dados enviados de uma caixa de seleção
$app->post('/rota14', function ($request, $response) {
    $dados = $request->getParsedBody();
    // Verifica se os checkboxes foram marcados
    $linguagem = $dados['lstLinguagem'] ?? "Não selecionado";
    // Monta o resultado
    $resultado = "Favorito:<br> $linguagem";


    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;


});
```



Favorito:
b

formularioRota15.html

```html
<!DOCTYPE html>
<html>
<body>
    <form action="/rota15" method="get">
        <p>Selecione um número:</p>
        <select name="cboNumero">
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5</option>
            <option value="6">6</option>
            <option value="7">7</option>
            <option value="8">8</option>
            <option value="9">9</option>
            <option value="10">10</option>
        </select>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

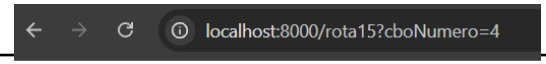


Selecione um número:

4 Enviar

/rota15

```php
$app->get('/rota15', function ($request, $response) {
    // Pega os parâmetros da URL (?cboNumero=5)
    $queryParams = $request->getQueryParams();
    // Verifica se foi enviado, senão usa 1 como padrão
    $numero = $queryParams['cboNumero'] ?? 1;
    // Monta a saída
    $resultado = "<h2>Tabuada do $numero</h2>";
    for ($i = 1; $i <= 10; $i++) {
        $calc = $numero * $i;
        $resultado .= "$numero x $i = $calc<br>";
    }
    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;
});
```



Tabuada do 4

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

formularioRota16.html

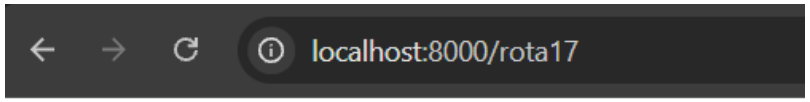
```html
<!DOCTYPE html>
<html>
<body>
    <form action="/rota16" method="get">
        <p>Selecione um número:</p>
        <select name="cboNumero">
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5</option>
            <option value="6">6</option>
            <option value="7">7</option>
            <option value="8">8</option>
            <option value="9">9</option>
            <option value="10">10</option>
        </select>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

/rota16

```php
$app->get('/rota16', function ($request, $response) {
    // Pega os parâmetros da URi (?cboNumero=5)
    $queryParams = $request->getQueryParams();
    // Verifica se foi enviado, senão usa 1 como padrão
    $numero = $queryParams['cboNumero'] ?? 1;
    // Monta a saída
    $resultado = "<h2>Tabuada do $numero</h2>";
    $i=0;
    while ($i <= 10 ) {
        $calc = $numero * $i;
        $resultado .= "$numero x $i = $calc<br>";
        $i++;
    }
    // Escreve no response
    $response->getBody()->write($resultado);
    return $response;
});
```

Estruturas de repetição em php (foreach)

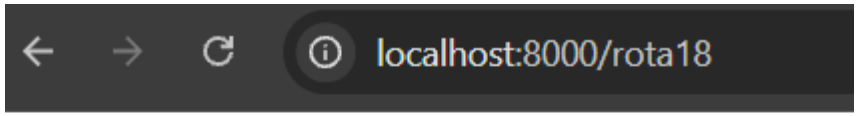
```php
$app->get('/rota17', function ($request, $response) {
    $vetor = array("html", "css", "php", "javaScript");
    $resposta = "";
    foreach ($vetor as $posicao) {
        $resposta .= "$posicao <br>";
    }
    // Escreve no response
    $response->getBody()->write($resposta);
    return $response;
});
```



html
css
php
javaScript

Foreach – Array associativo

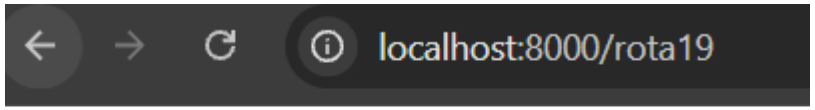
```php
$app->get('/rota18', function ($request, $response) {


    $cliente['nome'] = "helio";
    $cliente['cargo'] = "prof";
    $cliente['telefone'] = "12991777777";
    $cliente['cpf'] = "000.000.000-00";
    $resposta = "";
    foreach ($cliente as $posicao) {
        $resposta .= "$posicao <br>";
    }
```



helio
prof
12991777777
000.000.000-00

Array associativo

```php
$app->get('/rota19', function ($request, $response) {


    $cliente['nome'] = "helio";
    $cliente['cargo'] = "prof";
    $cliente['telefone'] = "12991777777";
    $cliente['cpf'] = "000.000.000-00";
    $resposta = "";

    // Escreve no response
    $response->getBody()->write($cliente['nome']);
    return $response;
});
```



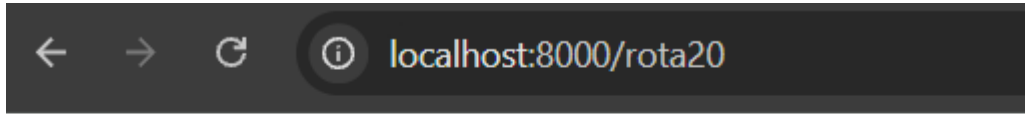
helio

Constantes matemáticas:
https://www.php.net/manual/pt_BR/math.constants.php

Constant	Value	Description
M_E	2.7182818284590452354	Returns e
M_EULER	0.57721566490153286061	Returns Euler constant
M_LNPI	1.14472988584940017414	Returns the natural logarithm of PI: log_e(pi)
M_LN2	0.69314718055994530942	Returns the natural logarithm of 2: log_e 2
M_LN10	2.30258509299404568402	Returns the natural logarithm of 10: log_e 10
M_LOG2E	1.4426950408889634074	Returns the base-2 logarithm of E: log_2 e
M_LOG10E	0.43429448190325182765	Returns the base-10 logarithm of E: log_10 e
M_PI	3.14159265358979323846	Returns Pi
M_PI_2	1.57079632679489661923	Returns Pi/2
M_PI_4	0.78539816339744830962	Returns Pi/4
M_1_PI	0.31830988618379067154	Returns 1/Pi
M_2_PI	0.63661977236758134308	Returns 2/Pi
M_SQRTPI	1.77245385090551602729	Returns the square root of PI: sqrt(pi)
M_2_SQRTPI	1.12837916709551257390	Returns 2/square root of PI: 2/sqrt(pi)
M_SQRT1_2	0.70710678118654752440	Returns the square root of 1/2: 1/sqrt(2)
M_SQRT2	1.41421356237309504880	Returns the square root of 2: sqrt(2)
M_SQRT3	1.73205080756887729352	Returns the square root of 3: sqrt(3)

Exemplos de uso de constantes matemáticas:

```php
$app->get('/rota20', function ($request, $response) {

    $x ="valor de pi = ". M_PI;

    // Escreve no response
    $response->getBody()->write($x);
    return $response;
});
```



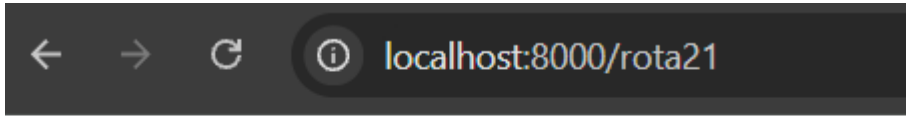
valor de pi = 3.14159265358898

Funções matemáticas: https://www.php.net/manual/pt_BR/ref.math.php

abs()	Retorna o absoluto (positive) valor de um número
acos()	Retorna o arco co-seno de um número
acosh()	Retorna o cosseno hiperbólico inverso de um número
asin()	Retorna o arco seno de um número
asinh()	Retorna o seno hiperbólico inverso de um número
atan()	Retorna o arco tangente de um número em radianos
atan2()	Retorna o arco tangente de duas variáveis x e y
atanh()	Retorna a tangente hiperbólica inversa de um número
base_convert()	Converte um número a partir de uma base numérica para outra
bindec()	Converte um número binário em um número decimal

Exemplo de uso de funções matemáticas.

```php
$app->get('/rota21', function ($request, $response) {
    $n = 16;
    $x ="Raiz de  = ". sqrt($n);
    // Escreve no response
    $response->getBody()->write($x);
    return $response;
});
```



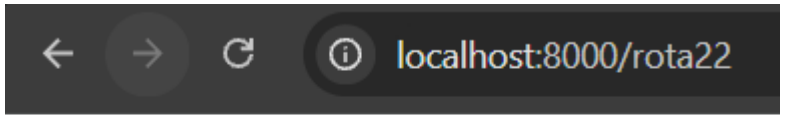
Raiz de = 4

Uso de funções de strings

- Durante o desenvolvimento de uma aplicação utilizamos o tipo de dado string para uma cadeia de caracteres ou texto.
- Diferente de outros tipos como int que trabalham com valores simples, uma string é um objeto composto de uma coleção de caracteres.
- Uma string pode utilizar métodos para manipular sua coleção de caracteres.
- Esses métodos permitem algumas tarefas, dentre elas:
  - Obter o seu tamanho ou quantidade de caracteres
  - Obter um trecho específico do texto
  - Substituir um trecho específico do texto
  - Verificar a ocorrência de um trecho específico de texto

explode - Divide uma String em um vetor

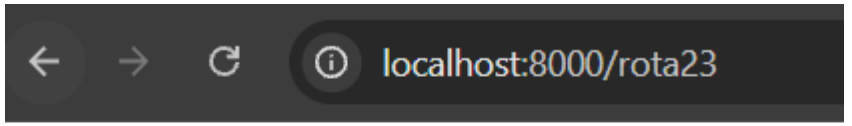
```php
$app->get('/rota22', function ($request, $response) {
    $string = "p1;p2;p3;p4;p5;p6";
    $vetor = explode(";", $string);
    $resposta = $vetor[0];


    // Escreve no response
    $response->getBody()->write($resposta);
    return $response;
});
```



p1

explode - Divide uma String em um vetor

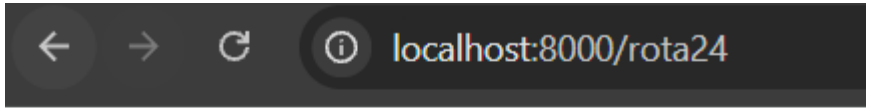
```php
$app->get('/rota23', function ($request, $response) {
    $string = "1;2;3;4;5;6";
    $vetor = explode(";", $string);
    $resposta = "";
    foreach ($vetor as $posicao) {
        $resposta .= "$posicao<br>";
    }

    // Escreve no response
    $response->getBody()->write($resposta);
    return $response;
});
```



1
2
3
4
5
6

Join – junta as strings dentro de um vetor em uma string única.

```php
$app->get('/rota24', function ($request, $response) {
    $string = "v1;v2;v3;v4;v5;v6";
    $vetor = explode(";", $string);
    $resposta = "";
    foreach ($vetor as $posicao) {
        $resposta .= "$posicao<br>";
    }


    // Escreve no response
    $response->getBody()->write($resposta);
    return $response;
});
```



v1
v2
v3
v4
v5
v6

password_hash

- Atualmente o PASSWORD_DEFAULT usa **bcrypt** (ou algoritmos ainda mais fortes em versões futuras do PHP).
- Resistente a *brute force*
- Projetado especificamente para senhas

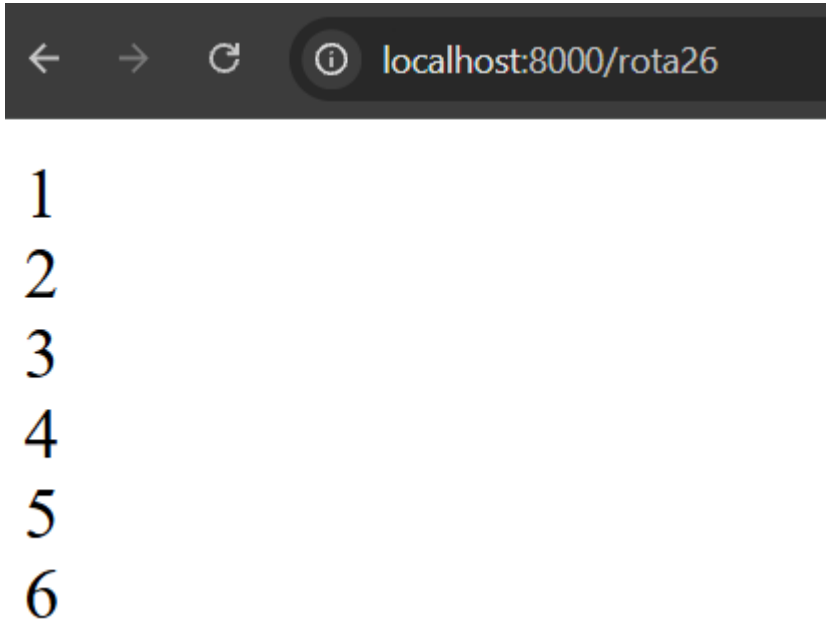
```php
$app->get('/rota25', function ($request, $response) {
    // Criar hash seguro
    $senha = "batataFrita123";
    $hash = password_hash($senha, PASSWORD_DEFAULT);


    // Verificar senha na autenticação
    if (password_verify($senha, $hash)) {
        //Senha correta";
    } else {
        //"Senha incorreta";
    }
    $resposta = "senha = $senha <br> hash = $hash";


    $response->getBody()->write($resposta);
    return $response;
});
```

senha = batataFrita123
hash = $2y$10$yqb2OfmXMgyhbLrHA1Mh1eGLSusli4ccK6QFlYy5GkxkUXphkfWVG

senha = batataFrita123
hash = $2y$10$xlUGS9rPyoCV7RT3K21OWOc7CcNRqyicxjO/cfGfC12VJzac6uazy

str_replace – Substitui um conjunto de caracteres por outro.

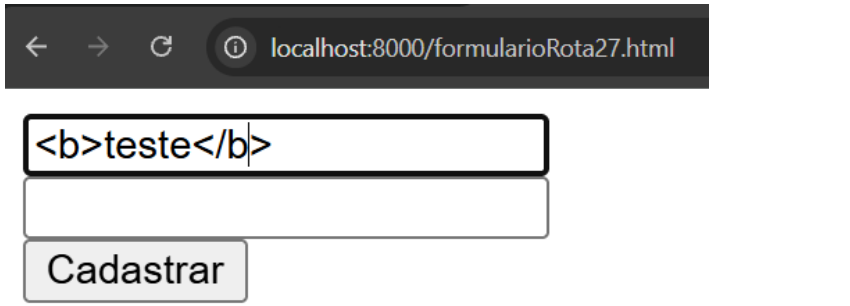
```php
$app->get('/rota26', function ($request, $response) {
    // Criar hash seguro
    $string = "1;2;3;4;5;6";

    $resposta = str_replace(";", "<br>", $string);

    $response->getBody()->write($resposta);
    return $response;
});
```



strip_tags – remove html de uma string

```php
// Rota 27: recebe email e senha
$app->post('/rota27', function ($request, $response) {
    // Captura os dados enviados pelo formulário via POST
    $dados = $request->getParsedBody();


    $email = $dados['txtEmail'] ?? '';
    $senha = $dados['txtSenha'] ?? '';


    $email = strip_tags($email);



    // Escreve a resposta
    $response->getBody()->write("remover tags de: [$email]");
    return $response;
});
```

- Os caracteres foram removidos
não há mais a tag <b>



Outras funções de String: https://www.php.net/manual/pt_BR/ref.strings.php

chr	Retorna um caracter específico
echo	Exibe uma ou mais strings
explode	Divide uma string em strings
fprintf	Escreve uma string formatada para um stream
join	Sinônimo de implode
lcfirst	Torna minúsculo o primeiro caractere de uma string
ltrim	Retira espaços em branco (ou outros caracteres) do início da string
md5	Calcula o "hash MD5" de uma string
money_format	Formata um número como uma string de moeda
nl2br	Insere quebras de linha HTML antes de todas newlines em uma string