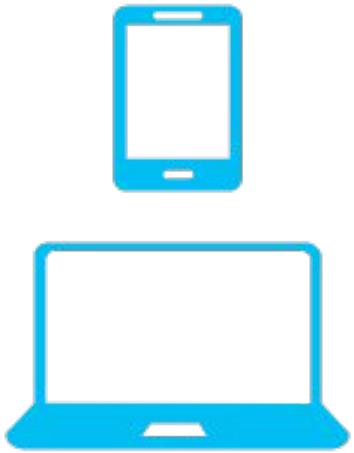
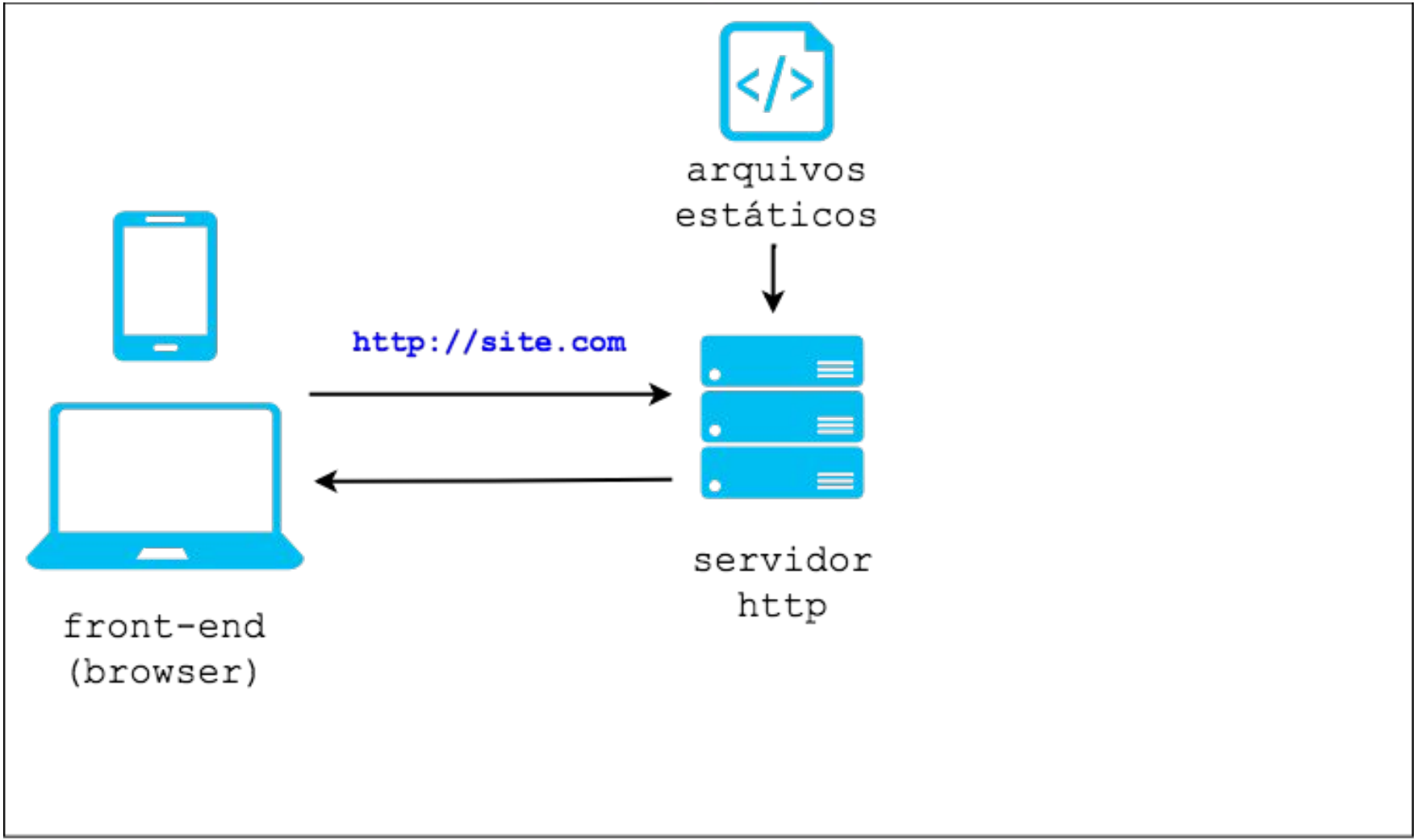
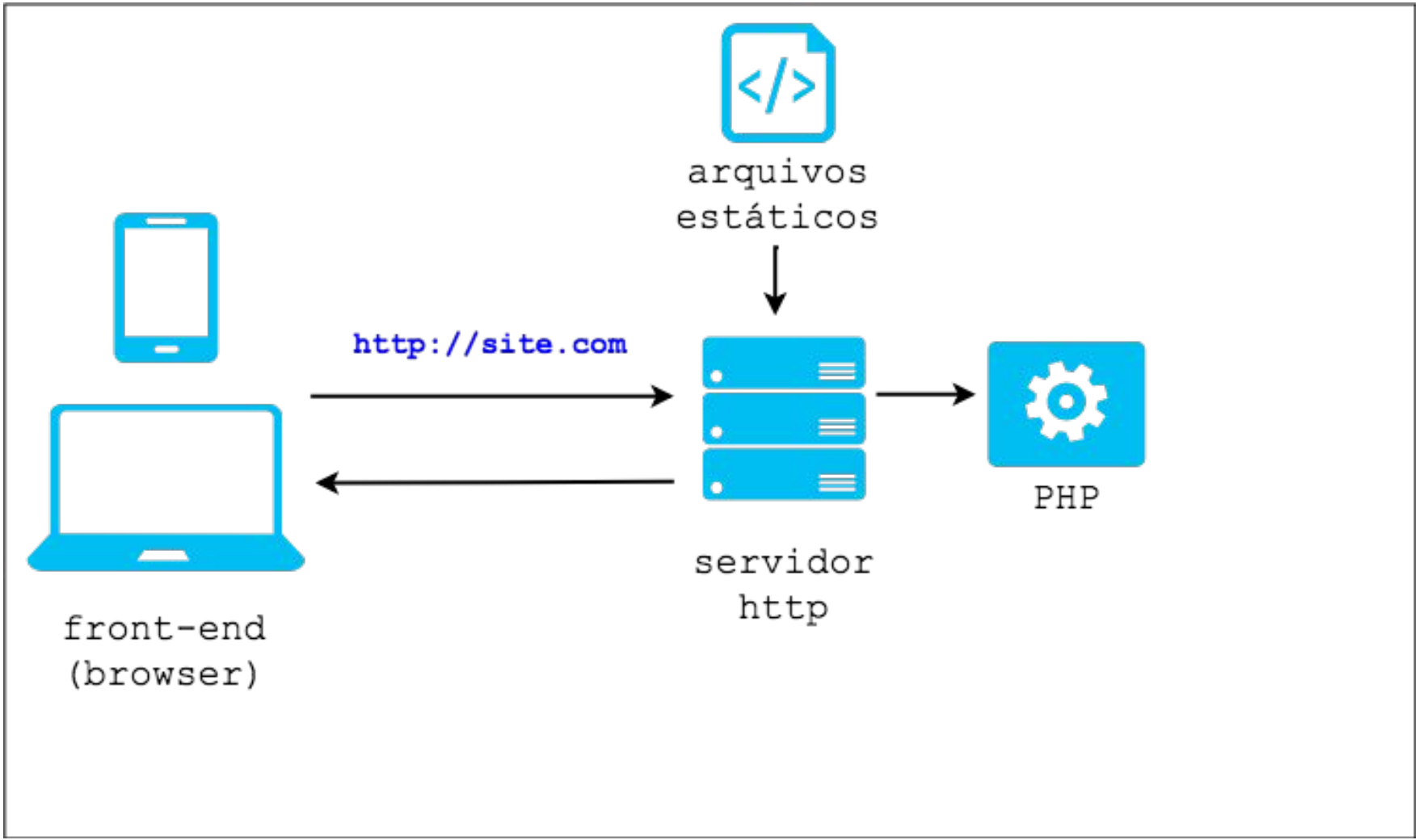
# PHP

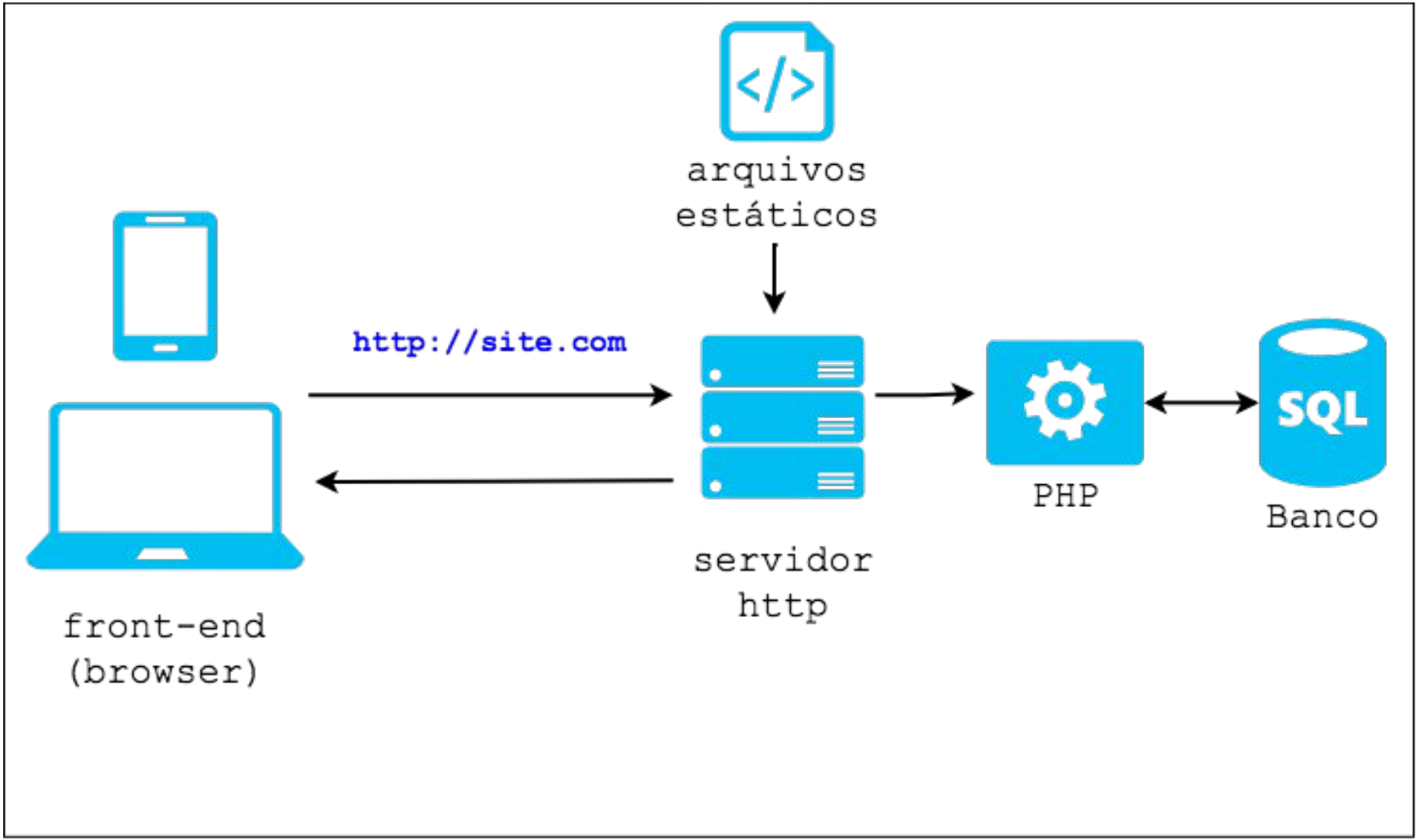
## Aula 1

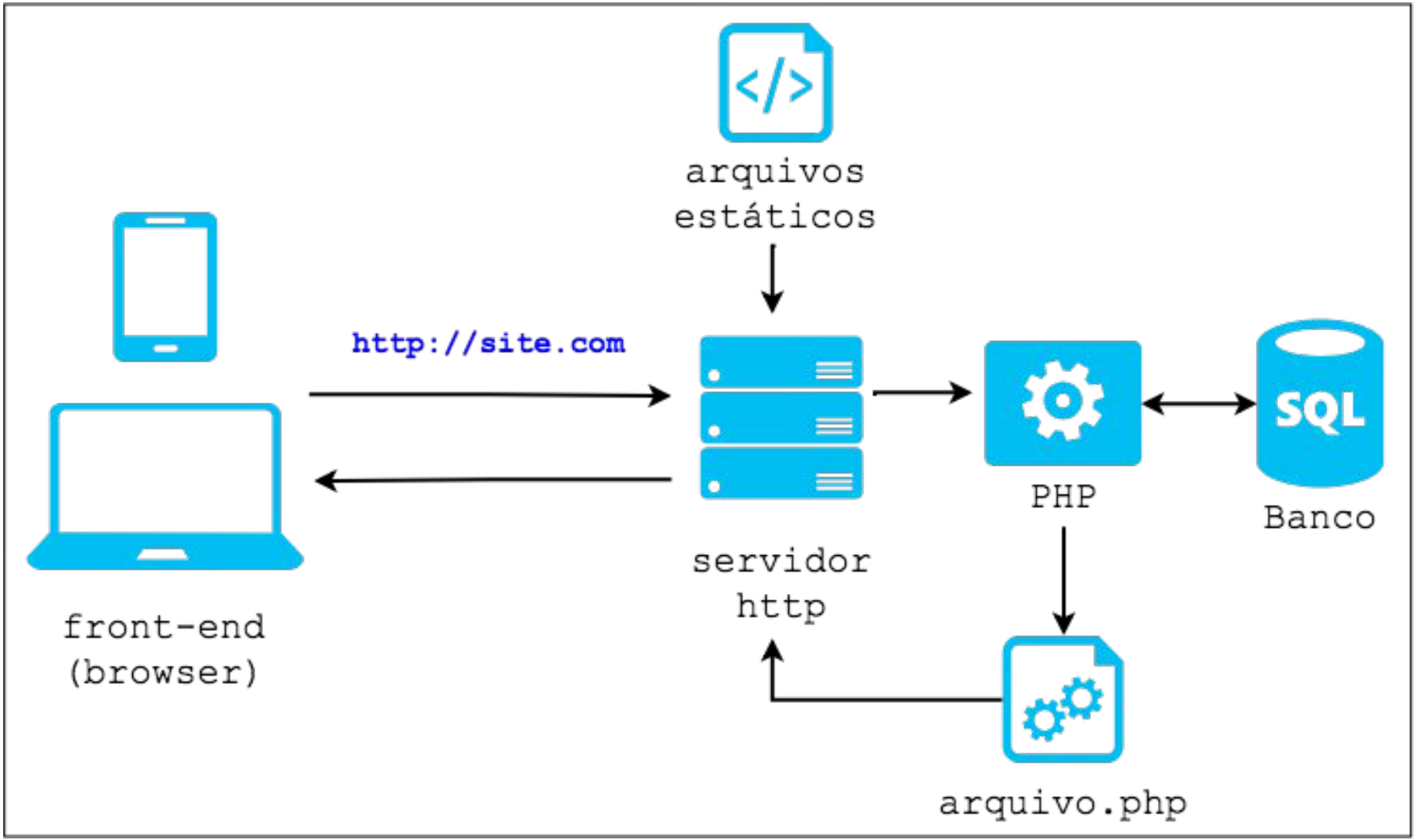


front-end
(browser)


arquivos
estáticos
http://site.com
servidor
http
front-end
(browser)


arquivos
estáticos
http://site.com
PHP
servidor
http
front-end
(browser)


arquivos
estáticos
http://site.com
front-end
(browser)
servidor
http
PHP
SQL
Banco


arquivos
estáticos
http://site.com
front-end
(browser)
servidor
http
PHP
SQL
Banco
arquivo.php

# Primeiros Passos

# Primeiros passos (exe1.php)

Veja como inserir código PHP nas páginas e declarar uma variável.

```php
<?php
    $nome = 'Fulano';
?>
```

Veja uma concatenação de strings para exibir na página.

```php
<?php
    for ($i=0; $i<10; $i++) {
        echo $i . ' ';
    }
?>
```

Simplificando a exibição na página.

```php
<?= $nome ?>
```

Intercalando PHP e HTML...

```php
<?php
    $variavel = false;
    if ($variavel) {
        ?>
            <span>Legal!</span>
        <?php
    } else {
        ?>
            <span>Ups...</span>
        <?php
    }
?>
```

# Funções e arrays (exe2.php)

Exemplo de função

```php
function somar($n1,$n2) {
    return $n1 + $n2;
}
$resultado = somar(5,10);
```

Exemplo de array

```php
$frutas = array(
    0 => "Banana",
    1 => "Maça",
    2 => "Pera",
    "outro" => "Laranja"
);
$frutas["mais1"] = "Uva";
```

Exemplo de *foreach* com sintaxe de fechamento através do comando *endforeach*.

```php
<?php foreach ($frutas as $chave => $valor): ?>
    <span><?= $chave . " - " . $valor ?></span>
<?php endforeach; ?>
```

# Classes e objetos (exe3.php)

## Instanciando objetos

```php
$joao = new Pessoa("João");
$maria = new Pessoa("Maria");
$maria->setIdade(20);
```

Acessar atributos privados da classe provoca um erro no PHP.

```php
<?= $maria->nome ?>
```

**Sempre** utilize métodos *getters* para acessar atributos privados.

## Exemplo de classe

```php
class Pessoa {
    private $nome;
    private $idade;
    public function __construct($nome) {
        $this->nome = $nome;
    }
    public function getNome() {
        return $this->nome;
    }
    public function getIdade() {
        return $this->idade;
    }
    public function setIdade($idade) {
        if ($idade > 0) {
            $this->idade = $idade;
        }
    }
}
```

# Importando arquivos PHP (exe4.php)

| | |
|---|---|
| **`include`** | Realiza a importação de um arquivo. Caso o arquivo não seja encontrado, mostrará um *warning* na tela. |
| **`require`** | Realizar a importação de um arquivo. Caso o arquivo não seja encontrado, mostrará um *error* na tela. |
| **`require_once`** | Semelhante ao *require*, porém com a diferença que o mesmo arquivo só poderá ser importado uma única vez. Nos casos acima, importar dois arquivos não provocará erro, porém pode afetar a lógica do sistema. |

Usar *require_once* é uma boa prática de projeto. Dessa forma você evita importações repetidas.

# Acesso ao Banco de Dados

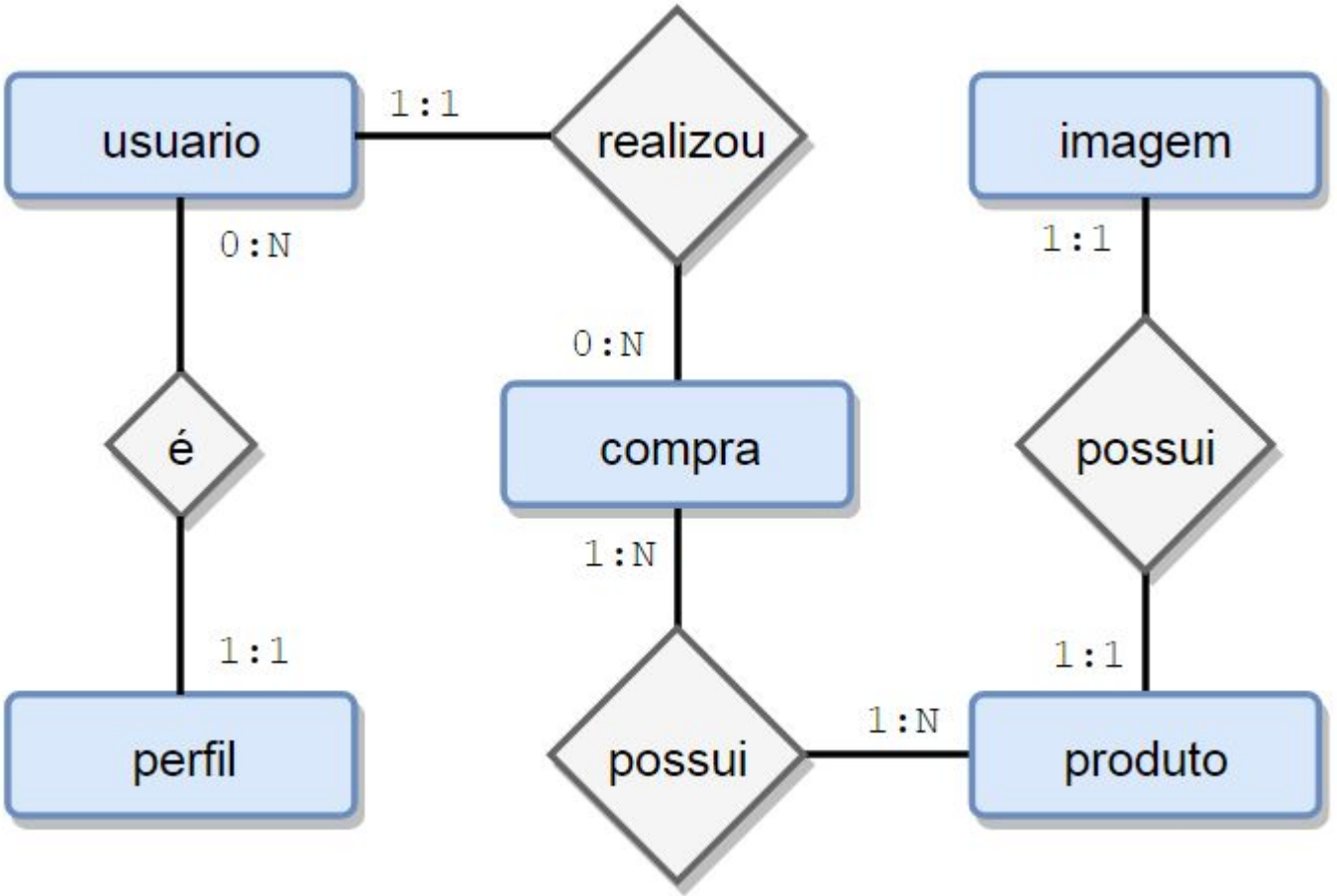
# Consultas no banco de dados (exe5.php)

Exemplo de acesso ao banco de dados (MySQL Improved).

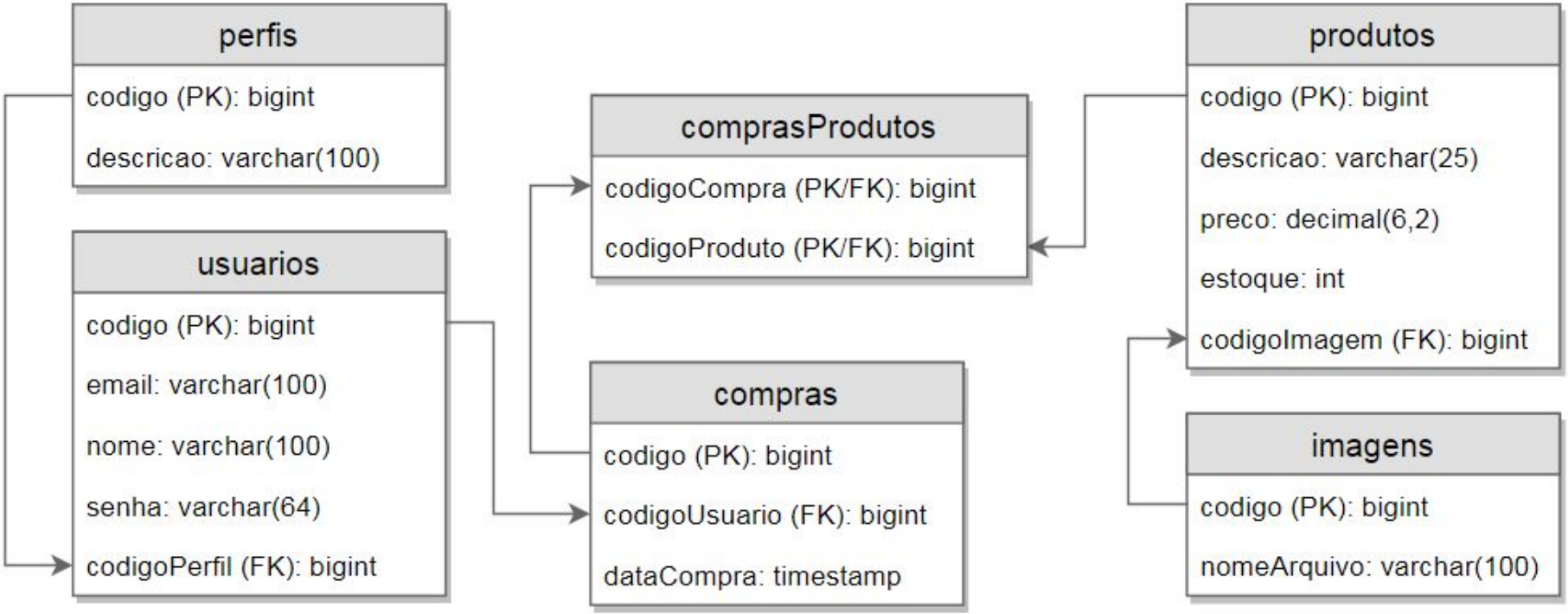
```php
$conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');

$resultado = $conexao->query('select * from perfis');

if ($resultado->num_rows > 0) {
    while ($linha = $resultado->fetch_assoc()) {
        echo $linha["codigo"] . " - " . $linha["descricao"] . "<br />";
    }
}

$conexao->close();
```

Opte por utilizar as bibliotecas MySQL Improved ou PDO.
https://secure.php.net/manual/pt_BR/book.mysqli.php

# Diagrama de Entidade e Relacionamento (site Roupas)



# Diagrama de Estrutura de Dados (site Roupas)



# Inserir no banco de dados (exe6.php)

Exemplo de *insert* no banco de dados (MySQL Improved).

```php
$conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');

$resultado = $conexao->query("insert into perfis (descricao) values ('Administrador')");
if ($resultado) {
    echo "Registro inserido com sucesso!";
} else {
    echo "Erro ao inserir registro!";
}

$conexao->close();
```

O problema neste caso é que ocorrendo um erro, não será possível realizar um tratamento adequado pois *$resultado* só retorna um valor booleano.

# Inserção de registros (exe7.php)

É uma boa prática configurar o **MySQL Improved** para disparar erros.

```
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
```

Exemplo de *insert* no banco de dados com tratamento de erros.

```
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);

try {
    $conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');
    $conexao->query("insert into perfis (descricao) values (null)");
    $conexao->close();
} catch(Exception $erro) {
    echo '<span><strong>Mensagem de erro:</strong> ' . $erro->getMessage() . '</span>';
}
```

# Atualização de registros (exe8.php)

Exemplo de *update* no banco de dados.

```php
$conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');
$conexao->query("update perfis set descricao = 'SuperUser' where descricao = 'Administrador'");

if ($conexao->affected_rows > 0) {
    echo "Registro alterado com sucesso!";
} else {
    echo "Nenhum registro foi alterado!";
}
```

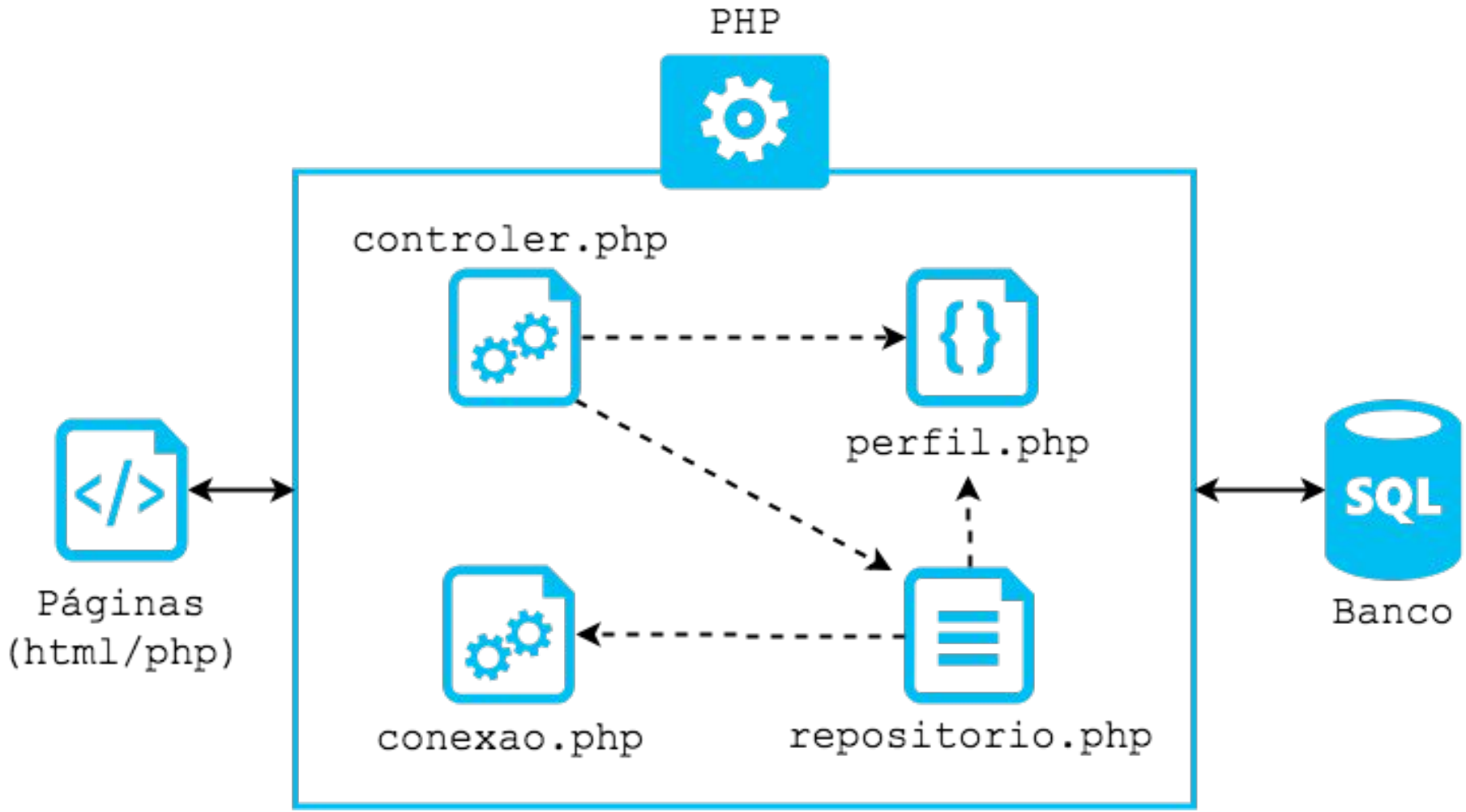
É uma boa prática utilizar o atributo *affected_rows* para verificar a quantidade de linhas que foram alteradas pelo UPDATE.

# Exclusão de registros (exe9.php)

Exemplo de *delete* no banco de dados.

```php
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
try {
    $conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');
    $conexao->query("delete from perfis where descricao = 'SuperUser'");

    if ($conexao->affected_rows > 0) {
        echo "Registro excluído com sucesso!";
    } else {
        echo "Nenhum registro foi excluído!";
    }
    $conexao->close();
} catch(Exception $erro) {
    echo '<span><strong>Mensagem de erro:</strong> ' . $erro->getMessage() . '</span>';
}
```

# Organizando o código de acesso ao banco


PHP
controler.php
perfil.php
Páginas
(html/php)
SQL
Banco
conexao.php
repositorio.php

# Arquivo de acesso ao banco (exe10/banco.php)

- Para exibir objetos completos na tela, utilize a função *var_dump()*. Uma boa prática é envolvê-la na tag HTML *<pre>* (para formatar a saída).
- Coloque a função *die()* para interromper o fluxo do sistema.
- Arquivos exclusivamente PHP não precisam do fechamento com *?>*

```php
<?php

mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
try {
    $conexao = new mysqli('127.0.0.1','roupas_user','123eja','roupas_database');
} catch(Exception $erro) {
    echo "<pre>";
    var_dump($erro);
    echo "</pre>";
    die();
}
```

# Arquivo da classe Perfil (exe10/perfil.php)

**É uma boa prática utilizar:**

- Atributos privados
- Métodos de acesso
- Constructor para o objeto, preferencialmente recebendo parâmetros essenciais ao mesmo.

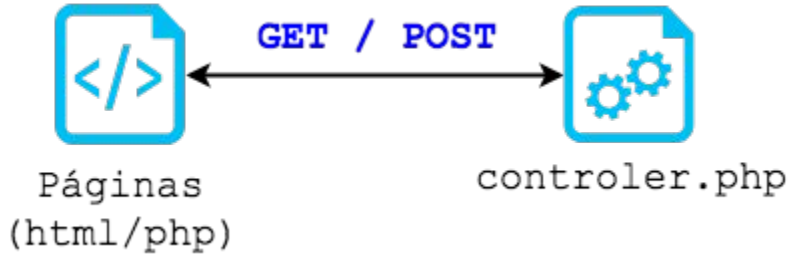
```php
class Perfil {
    private $codigo;
    private $descricao;

    public function __construct($codigo,$descricao) {
        $this->codigo = $codigo;
        $this->descricao = $descricao;
    }
    public function getCodigo() {
        return $this->codigo;
    }
    public function getDescricao() {
        return $this->descricao;
    }
}
```

# Arquivo de repositório (exe10/repositorio.php)

Veja o arquivo *exe10/repositorio.php* para visualizar o código deste script. Na prática ele mantém métodos para listar todos os perfis, excluir um perfil pelo código, e cadastrar um novo perfil pela descrição.

Com exceção da tag <*pre*>, utilizada especialmente neste exercício para exibir erros em tempo de desenvolvimento da aplicação, **nunca** mantenha tags ou formatação HTML dentro deste arquivo.
Ele serve exclusivamente para manipular os dados referentes aos perfis. Métodos de consulta devem retornar uma lista de objetos Perfil, e métodos manipulação de dados (inserir, alterar e excluir) devem apenas retornar sucesso ou falha na operação (ou Exceptions).

# Página/Controller (exe10/index.php)




As páginas HTML/PHP, quando pequenas, podem conter internamente as rotas (Controller) da aplicação, com código PHP específico para manipulação das entidades do sistema. No geral, entretanto, é comum manter arquivos diferentes para as *controllers*. Veremos como separá-las em breve.

## Ok, mas como uma página passa conteúdo para a controller?

A tela de manutenção dos perfis do exercício 10 possui falhas de segurança que serão detalhadas e resolvidas em breve!

# Enviando dados com GET e POST

# Método GET (exe11.php)

A passagem por GET ocorre através das *queries strings*, que são variáveis passadas na URL da página. Use **?** para iniciar uma sequência de *queries string* e **&** para separar as variáveis.

# Método GET (exe11.php)

Para obter um valor da *query string* através do PHP, utilize o array **$_GET**, passando entre colchetes uma string com o nome da variável.

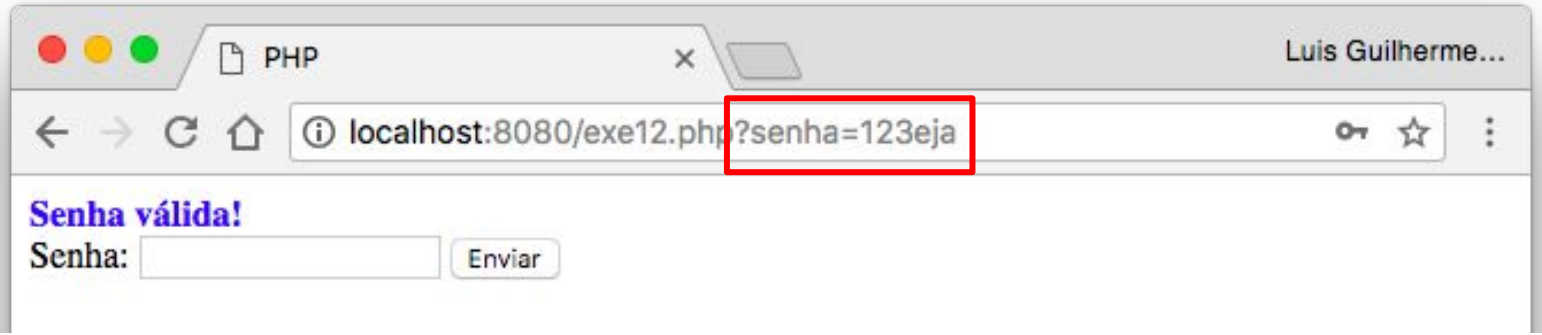
```php
if (isset($_GET['nome'])) {
    $nome = $_GET['nome'];
    echo "<strong>Valor do nome passado por query string:</strong> " . $nome;
} else {
    echo "Nenhum nome informado!";
}
```

Utilize a função *isset()* para verificar se a variável existe.
**Observação:** dependendo da configuração do servidor HTTP, um *warning* será exibido se a variável não existir no **$_GET.** Você pode evitar esse problema afrouxando os logs de erro do PHP com o comando:

```php
error_reporting(E_ERROR | E_PARSE);
```

# Cuidados com o GET (exe12.php)

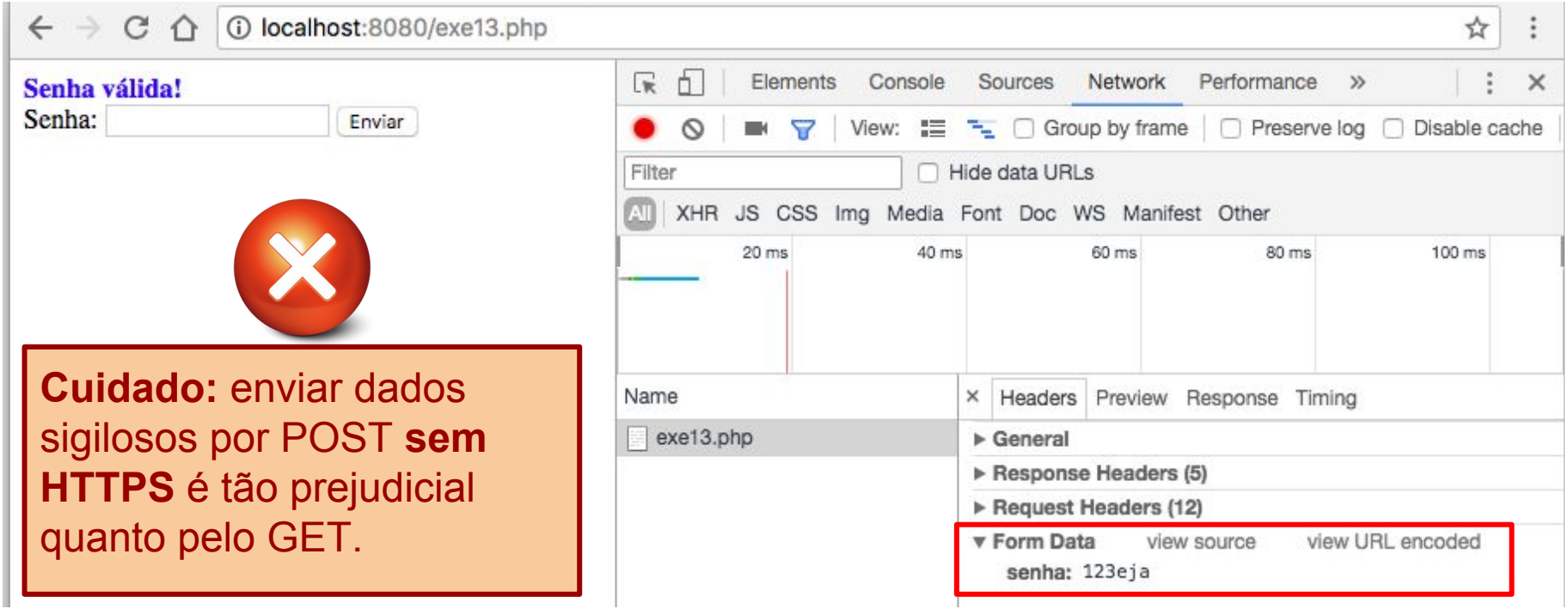
- Cuidado com os dados que são passados para o servidor através de *query string*, pois eles ficam expostos na URL.
- Cuidado também porque apesar das RFC 2616 e RFC 7230 serem mais abrangentes no que toca o limite da URL, na prática devemos trabalhar com no máximo 2000 caracteres.
- Por fim, cuidado também porque manter eventos de manipulação de dados por GETs pode fazer com que **bots** de consulta (do Google, por exemplo) entre nos links e execute as ações.



# Método POST (exe13.php)

A passagem por POST envia os dados no *body* da solicitação.

É uma boa prática utilizar o método POST para enviar formulários.




**Cuidado:** enviar dados sigilosos por POST **sem HTTPS** é tão prejudicial quanto pelo GET.

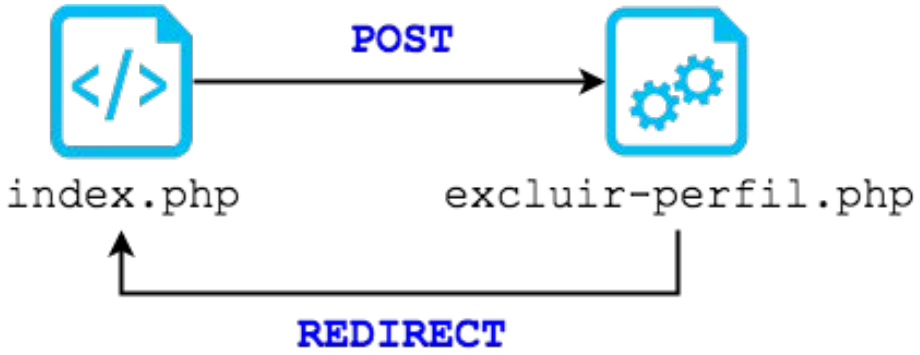
# Método POST (exe13.php)

Para capturar dados do POST no PHP, utilize o array **$_POST**

```php
<?php
    if (isset($_POST['senha'])) {
        $senha = $_POST['senha'];
        if ($senha == '123eja') {
            echo '<strong class="sucesso">Senha válida!</strong>';
        } else {
            echo '<strong class="erro">Senha inválida!</strong>';
        }
    }
?>
<form method="POST">
    <label for="senha">Senha:</label>
    <input name="senha" id="senha" type="password" />
    <button>Enviar</button>
</form>
```

# Redirect após controller (exe14.php)

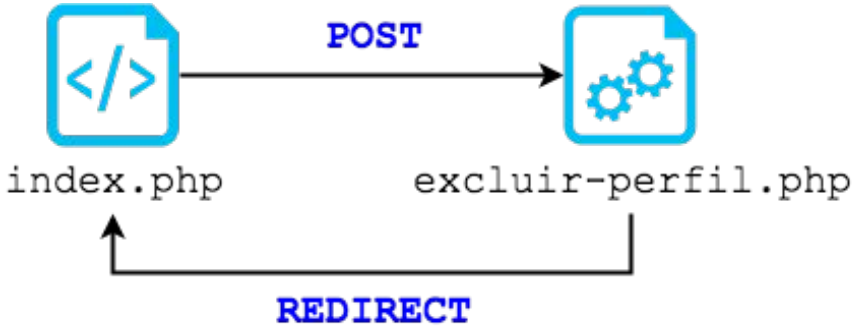
Outra prática que é comum no PHP é separar as lógica da controller em arquivos diferentes. Neste casos, após a execução do script, o servidor faz um redirecionamento para outra página da aplicação.




```php
if ($login != 'lgapontes' || $senha != '123eja') {
    header('Location: exe14.php?mensagem=0');
    die();
} else {
    header('Location: exe14.php?mensagem=1');
    die();
}
```

Use o método *header()* com a sintaxe *Location* para realizar o redirecionamento. **Sempre** use *die()* após este comando.

# Página/Controller (exe15/index.php)




Vamos agora ajustar a tela de manter Perfis com as boas práticas de envio de dados ao servidor, com os seguintes itens:

- Substituir os links de exclusão por formulários via POST
- Validando os dados enviados com *isset()*
- Criando scripts PHP separados por lógica que devem redirecionar para a página principal após a execução, exibindo a mensagem apropriada.

Passar informações de sucesso/falha por GET não é uma boa prática.
Veremos em breve outra forma de realizar isso (com SESSION).

# Página/Controller (exe15/index.php)

## Formulário para excluir perfis

```
<form method="POST" action="excluir-perfil.php">
    <input type="hidden" name="codigo"
        value="<?= $perfil->getCodigo() ?>">
    <button class="link">Excluir</button>
</form>
```

## Redirecionamento após exclusão do perfil

```
$resultadoExclusao = $repositorio->excluir($codigo);
if ($resultadoExclusao) {
    header("Location: index.php?mensagem=3");
    die();
} else {
    header("Location: index.php?mensagem=2");
    die();
}
```

# Obrigado!