

## **Estrutura de Pastas (XAMPP → htdocs/projeto\_credito)**

```
projeto_credito/
|
├── index.php    -> página inicial (front)
├── simular.php  -> processa a simulação (back)
├── resultado.php -> mostra resultado (front/back com sessão)
├── style.css     -> CSS simples
└── includes/
    └── funcoes.php -> funções de cálculo (back)
```

---

### **index.php (entrada de dados – FRONT)**

```
<?php
session_start();

// se já existe cookie com nome, pré-preenche
$nome = isset($_COOKIE['nome']) ? $_COOKIE['nome'] : "";
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Simulação de Crédito - Auto Popular</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h2>Simule seu Crédito</h2>
    <form action="simular.php" method="post">
        <label>Nome:</label>
        <input type="text" name="nome" value="<?php echo $nome; ?>" required><br><br>

        <label>Renda mensal (R$):</label>
        <input type="number" name="renda" required><br><br>

        <label>Valor do carro (R$):</label>
        <input type="number" name="valor" required><br><br>

        <label>Parcelas:</label>
        <select name="parcelas">
            <option value="12">12x</option>
            <option value="24">24x</option>
            <option value="36">36x</option>
        </select><br><br>

        <button type="submit">Simular</button>
    </form>
</body>
</html>
```

---

### **simular.php (processamento – BACK)**

```

<?php
session_start();
include "includes/funcoes.php";

// recebe dados do formulário
$nome = $_POST['nome'];
$renda = $_POST['renda'];
$valor = $_POST['valor'];
$parcelas = $_POST['parcelas'];

// cria cookie para lembrar nome por 1 dia
setcookie("nome", $nome, time() + 86400);

// usa função para calcular parcela
$parcela = calcularParcela($valor, $parcelas);

// regra simples: parcela não pode ultrapassar 30% da renda
$limite = $renda * 0.3;
$aprovado = ($parcela <= $limite);

// salva em sessão
$_SESSION['resultado'] = [
    "nome" => $nome,
    "valor" => $valor,
    "parcelas" => $parcelas,
    "parcela" => $parcela,
    "renda" => $renda,
    "aprovado" => $aprovado
];
;

// redireciona para resultado
header("Location: resultado.php");
exit;

```

---

#### **includes/funcoes.php (funções reutilizáveis)**

```

<?php
function calcularParcela($valor, $parcelas) {
    $juros = 0.02; // 2% ao mês fixo
    $valorComJuros = $valor * pow((1+$juros), $parcelas/12);
    return round($valorComJuros / $parcelas, 2);
}

```

---

#### **resultado.php (exibição – FRONT/BACK)**

```

<?php
session_start();

if (!isset($_SESSION['resultado'])) {
    echo "<p>Simulação expirada. Volte à página inicial.</p>";
    exit;
}

```

```

}

$res = $_SESSION['resultado'];
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Resultado da Simulação</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h2>Resultado da Simulação</h2>
    <p><b>Nome:</b> <?php echo $res['nome']; ?></p>
    <p><b>Valor do carro:</b> R$ <?php echo number_format($res['valor'],2,'.,.');?>"?></p>
    <p><b>Parcelas:</b> <?php echo $res['parcelas']; ?>x</p>
    <p><b>Valor de cada parcela:</b> R$ <?php echo number_format($res['parcela'],2,'.,.');?>"?></p>
    <p><b>Renda declarada:</b> R$ <?php echo number_format($res['renda'],2,'.,.');?>"?></p>
    <hr>
    <?php if ($res['aprovado']) { ?>
        <p style="color:green;">✓ Parabéns! Crédito aprovado.</p>
    <?php } else { ?>
        <p style="color:red;">✗ Infelizmente o crédito não foi aprovado.</p>
    <?php } ?>

    <a href="index.php">Nova simulação</a>
</body>
</html>

```

---

### style.css (bem simples)

```

body {
    font-family: Arial, sans-serif;
    margin: 40px;
    background: #f9f9f9;
}
h2 {
    color: #333;
}
form, .resultado {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0px 0px 6px rgba(0,0,0,0.1);
}
label {
    display: inline-block;
    width: 150px;
}
button {

```

```
padding: 8px 15px;  
background: #1976d2;  
color: white;  
border: none;  
border-radius: 5px;  
}  
button:hover {  
    background: #125a9c;  
}
```

---

#### Recursos trabalhados no exemplo:

- **Front-end separado do back-end** (index.php = entrada, simular.php = processamento, resultado.php = saída).
- **Funções em arquivo separado** (includes/funcoes.php).
- **Sessões** para carregar o resultado sem reenvio de formulário.
- **Cookies** para lembrar o nome do usuário.
- **Arrays** (resultado salvo em \$\_SESSION['resultado']).
- **HTML + CSS** para a interface simples.

ATENÇÃO:

**1) OPERADOR TERNÁRIO:**

```
$nome = isset($_COOKIE['nome']) ? $_COOKIE['nome'] : "";
```

Se você quiser reescrever com if e else, ficaria assim:

```
if (isset($_COOKIE['nome'])) {  
    $nome = $_COOKIE['nome'];  
} else {  
    $nome = "";  
}
```

👉 O resultado é exatamente o mesmo:

- Se existir o cookie nome, ele pega o valor.
- Caso contrário, define como string vazia "".

**2) O fato de simular.php e funcoes.php não terem o fechamento ?> não é um erro.**

Na verdade, em **arquivos PHP que contêm apenas código PHP**, a recomendação oficial do próprio manual do PHP é **não usar o fechamento ?>**.

**Motivos:**

1. **Evita espaços/linhas em branco acidentais** depois do ?>, que poderiam gerar erros de *headers already sent* (muito comuns quando trabalhamos com session\_start(), setcookie() etc.).
2. **Boa prática de estilo**: frameworks como Laravel, WordPress e CodeIgniter seguem essa convenção.
3. **Mais seguro e limpo**: já que o arquivo termina no fim do código PHP, o parser entende o final automaticamente.

---

**Quando usar ?>**

- Quando o arquivo mistura **HTML + PHP** (ex.: index.php ou resultado.php), aí sim precisa do ?> para "voltar" ao HTML.
- Em arquivos só de lógica (ex.: simular.php, funcoes.php, config.php), a boa prática é **não fechar**.

---

👉 Ou seja, no projeto que te mostrei:

- **index.php e resultado.php** → precisam do fechamento porque alternam HTML e PHP.

- **similar.php** e **funcoes.php** → não precisam, e é até melhor sem.

### 3) **function calcularParcela(\$valor, \$parcelas)**

Vamos dissecar a função linha por linha para que fique bem claro:

```
function calcularParcela($valor, $parcelas) {
    $juros = 0.02; // 2% ao mês fixo
    $valorComJuros = $valor * pow((1+$juros), $parcelas/12);
    return round($valorComJuros / $parcelas, 2);
}
```

---

#### 1. Definição da função

```
function calcularParcela($valor, $parcelas) {
```

- A função recebe dois parâmetros:
    - **\$valor** → valor total do carro ou do financiamento.
    - **\$parcelas** → número de vezes que o valor será dividido.
- 

#### 2. Taxa de juros

```
$juros = 0.02; // 2% ao mês fixo
```

- Aqui define-se um **juros fixo de 2% ao mês** ( $0.02 = 2/100$ ).
  - Esse valor vai ser aplicado de forma composta no cálculo.
- 

#### 3. Cálculo do valor com juros

```
$valorComJuros = $valor * pow((1+$juros), $parcelas/12);
```

- **pow(base, expoente)** eleva a base a uma potência.
- A fórmula usada aqui é uma adaptação de **juros compostos**:

$$M = P \times (1+i)^t$$

Onde:

- **P** = valor inicial (principal).
- **i** = taxa de juros.
- **t** = tempo (período de aplicação).
- No código:
  - Base =  $1 + \$juros = 1.02$

- Expoente = \$parcelas / 12 → converte número de meses em anos (ex.: 24 meses = 2 anos).
- Então o montante final é o **valor financiado + acréscimo de juros compostos anuais**.

⚠ Observação: esse cálculo não é o mesmo usado em financiamentos reais (que seguem fórmulas de amortização como **Tabela Price** ou **SAC**). Aqui é uma simplificação didática.

---

#### 4. Cálculo da parcela

```
return round($valorComJuros / $parcelas, 2);
```

- O valor total com juros é dividido pela quantidade de parcelas.
  - **round(..., 2)** arredonda o valor para **2 casas decimais**, simulando centavos.
  - O resultado é a **parcela mensal** aproximada.
- 

#### Exemplo prático

Se chamarmos:

```
echo calcularParcela(30000, 24);
```

- Valor = R\$ 30.000,00
- Parcelas = 24 meses
- Juros = 2% ao mês (mas aplicado de forma anualizada pela fórmula)

O cálculo seria:

1.  $(1 + 0.02) = 1.02$
2.  $24 / 12 = 2 \rightarrow \text{expoente} = 2$  (2 anos)
3.  $\text{pow}(1.02, 2) = 1.0404$
4.  $30000 * 1.0404 = 31.212$
5.  $31.212 / 24 \approx 1300,50$

Ou seja, a parcela seria **R\$ 1.300,50**.

---

#### Resumo da função:

- Calcula uma **parcela mensal aproximada**, dividindo o valor com juros compostos pelo número de parcelas.
- É uma **simplificação didática** para praticar funções em PHP, mas não reflete cálculos bancários reais.

#### 4) funcionamento do trecho: `$aprovado = ($parcela <= $limite);`

##### 1. O que está acontecendo?

- O código está fazendo uma **comparação**:

`($parcela <= $limite)`

Ele verifica se o valor da **parcela** é **menor ou igual** ao **limite** permitido.

---

##### 2. Resultado da comparação

- Em PHP (e na maioria das linguagens), expressões condicionais retornam um valor **booleano**:
  - `true` (verdadeiro) se a condição é satisfeita.
  - `false` (falso) se a condição não é satisfeita.

Exemplo:

```
$parcela = 500;
```

```
$limite = 600;
```

```
$aprovado = ($parcela <= $limite);
```

```
// $aprovado vai receber true
```