Tarefa 3: Control de stock intelixente Imos aplicar as condicions a un problema típico do comercio electronico: xestionar o stock dun produto.

Obxectivo: Crear un script que, en funcion dunha variable de stock, mostre unha mensaxe diferente ao usuario.

Pasos:

1. Declara unha variable $stock e asígnalle un valor numerico enteiro. Proba con diferentes valores (ex: 20, 5, 0, -1) para ver como cambia a mensaxe.

2. Escribe unha estrutura if-elseif-else que cubra os seguintes casos:

- Se o stock e maior que 10, mostra unha mensaxe en cor verde: "Disponibilidade alta".
- Se o stock e maior que 0 pero menor ou igual que 10, mostra unha mensaxe en cor laranxa: "Ultimas unidades!".
- Se o stock e exactamente 0, mostra unha mensaxe en cor vermella: "Produto esgotado".
- Para calquera outro caso (ex: un numero negativo, que sería un erro), mostra unha mensaxe de "Erro: O stock non pode ser negativo."

Consello: Para as cores, podes usar HTML directamente no echo. Por exemplo: echo ' Dispoñibilidade alta ';


```php
<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tarefa3CarlosML</title>
</head>
<body>
  <?php
  $stock = 20;

  if ($stock > 0): ?>
    <h1 style="color:green;">Dispoñibilidade alta</h1>

  <?php elseif ($stock === 0): ?>
    <h1 style="color:red;">Producto esgotado</h1>

  <?php elseif ($stock > 0 && $stock <= 10): ?>
    <h1 style="color:orange;">Últimas unidades!</h1>

  <?php else: ?>
    <h1>Erro: O stock non pode ser negativo.</h1>

  <?php endif; ?>
</body>
</html>
```

Tarefa 4: Mensaxes de Estado HTTP

Imos usar a estrutura match para crear un "tradutor" de codigos de estado HTTP, algo moi comun no desenvolvemento web.

Obxectivo: Dado un codigo de estado HTTP nunha variable numerica, usar match para asignar unha mensaxe descritiva a outra variable e mostrala.

Pasos: 1. Declara unha variable $codigo_http e asígnalle un valor numerico (proba con 200, 404, 500, etc.).

2. Usa unha expresion match que avalíe a variable $codigo_http.

3. A expresion match debe devolver un valor de texto que asignaras a unha nova variable chamada $significado.

4. Define os seguintes casos:

o Para os codigos 200, 201 e 204, a mensaxe debe ser "Exito (Success)". (Nota: podes agrupalos nunha soa lina!).

o Para o codigo 400, a mensaxe debe ser "Peticion incorrecta (Bad Request)".

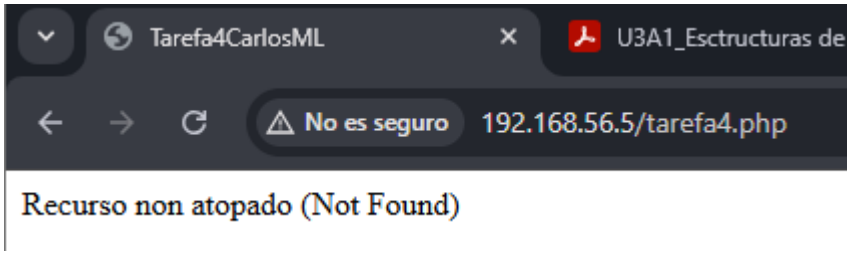
o Para o codigo 404, a mensaxe debe ser "Recurso non atopado (Not Found)".

o Para o codigo 500, a mensaxe debe ser "Erro interno do servidor (Internal Server Error)".

o Engade un caso default para calquera outro codigo, coa mensaxe "Codigo de estado desconecido".

5. Mostra o resultado final.

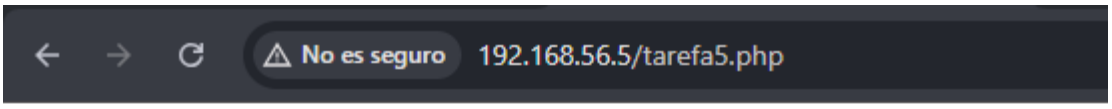
Resultado esperado na pantalla (se $codigo_http = 404): 17 / 30O código de estado 404 significa: Recurso non atopado (Not Found)




```php
<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tarefa4CarlosML</title>
</head>
<body>
  <?php $codigo_http=404;

  $significado = match ($codigo_http){
    200,201,204=> 'Exito (Success)',
    400=> 'Peticion incorrecta (Bad Request)',
    404=> 'Recurso non atopado (Not Found)',
    500=> 'Erro interno do servidor (Internal Server Error)',
    default => 'Codigo de estado desconecido'
  };
  echo $significado;
  ?>
</body>
</html>
```

Tarefa 5: Xerador de Cabeceiras HTML

E hora de poner en practica os bucles. Imos crear un script que use un bucle para xerar cabeceiras HTML de diferentes niveis. Podes resolvelo con for ou while.
Obxectivo: Crear unha paxina PHP que, utilizando un bucle, mostre 6 veces a frase “CURSO DE PHP”, cada vez dentro dunha etiqueta de cabeceira diferente, dende <h1> ata <h6>.
Pasos: 1. Inicia un bucle que se execute 6 veces (podes usar un contador dende 1 ata 6).
2. Dentro do bucle, utiliza a variable do contador para construír dinamicamente a etiqueta HTML de apertura e peche (ex: <h1>, <h2>, etc.).
3. Mostra a frase "CURSO DE PHP" envolvida nesas etiquetas



CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

CURSO DE PHP

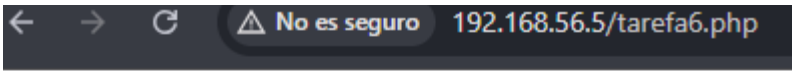
CURSO DE PHP

CURSO DE PHP

```php
<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <?php
  $contador=1;

  for($i;$i<=6;$i++){
    print "<h{$i}>CURSO DE PHP</h{$i}>";
  }

  while ($contador<=6) {
    print "<h{$contador}>CURSO DE PHP</h{$contador}>";
    $contador++;
  }
  ?>
</body>
</html>
```

Tarefa 6: O Catálogo de produtos
Imos combinar todo o aprendido para crear unha tarefa mais realista: mostrar un pequeno catalogo de produtos a partir dun array. 23 / 29Obxectivo: Crear un script que defina un array con nomes de produtos e despois use un bucle foreach para mostralos nunha lista ordenada dentro dunha paxina HTML.
Pasos: 1. Crea un array chamado $produtos que contena polo menos 5 nomes de produtos como cadeas de texto.
o Exemplo: $produtos = ["Portátil M2", "Rato USB", "Teclado Mecánico", "Monitor 4K", "Webcam HD"];

2. Escribe a estrutura HTML basica dunha paxina (non te preocupes polo CSS).
3. Usa echo para escribir a etiqueta de apertura dunha lista: <ol>.
4. Inicia un bucle foreach que percorra o teu array $produtos.
5. Dentro do bucle, por cada produto, mostra un elemento de lista <li> que contena o nome do produto.
6. Despois do bucle, usa echo para pechar a lista ordenada: . Resultado esperado na pantalla: 1. Portatil M2 2. Rato USB 3. Teclado mecanico 4. Monitor 4K 5. Webcam HD

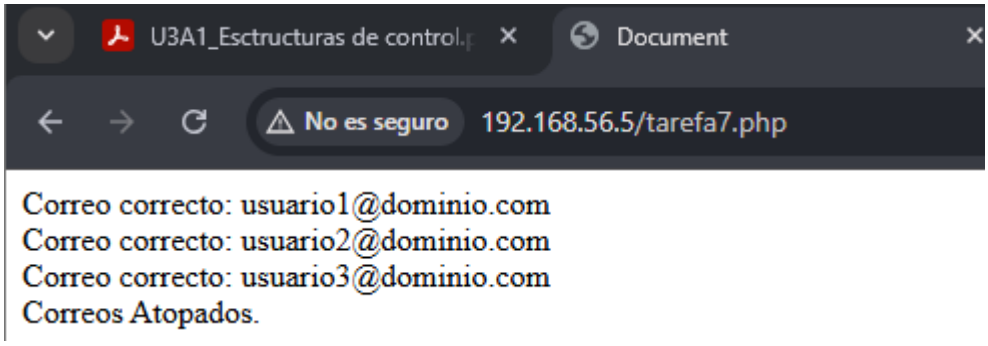


1. Portatil
2. Camara
3. Procesador
4. Ratón
5. Disco Duro

```
<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  <?php
  echo "<ol>";
    $productos=["Portatil","Camara","Procesador","Ratón","Disco Duro"];

    foreach($productos as $producto){
      echo "<li>$producto</li>";
    }
    echo "</ol>";
  ?>

</body>
</html>
```

Tarefa 7: Validador de correos simplificado
Imos combinar break e continue para crear unha ferramenta que valide unha lista de correos electronicos.
Obxectivo: Percorrer unha lista de posibles correos electronicos. Debes ignorar as entradas non validas (continue) e deter o proceso unha vez que atopes 3 correos validos (break).
Pasos: 1. Define un array $lista_correos con varias cadeas de texto. Inclue correos validos, outros sen "@" e algunhas cadeas baleiras.
o Exemplo: ["usuario1@dominio.com", "texto-sen-arroba", "", "usuario2@dominio.com", "usuario3@dominio.com", "usuario4@dominio.com"]
2. Crea unha variable contador $validos_atopados = 0;.
3. Inicia un bucle foreach para percorrer a lista.
4. Dentro do bucle: o Primeiro, comproba se a cadea esta baleira ou non conten o caracter "@". Podes usar a funcion str_contains() para isto (if (! str_contains($correo, '@'))).
o Se non e un correo valido, usa continue para saltar ao seguinte elemento da lista.
o Se e valido, imprime unha mensaxe indicandoo e incrementa o contador $validos_atopados.
o Xusto despois, comproba se $validos_atopados chegou a
3. Se e así, usa break para deter o bucle.

Resultado esperado na pantalla (coa lista de exemplo): Correo válido atopado: usuario1@dominio.com Correo válido atopado: usuario2@dominio.com Correo válido atopado: usuario3@dominio.com Proceso rematado. Atopáronse 3 correos válidos.




```
<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <?php
```

```
    $lista_correos= ["usuario1@dominio.com", "texto-sen-arroba", "",
"usuario2@dominio.com", "usuario3@dominio.com", "usuario4@dominio.com"];
    $validos_atopados= 0;

    foreach($lista_correos as $correo){
      if (!str_contains($correo, '@')){
      continue;
      }else{
        echo "Correo correcto: $correo </br>";
        $validos_atopados++;
        if($validos_atopados===3){
          echo "Correos Atopados.";
          break;
        }
      }
    }

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

Tarefa 8: Comentando o teu código
Esta tarefa consiste en aplicar as boas practicas de documentacion a un exercicio anterior.
Obxectivo: Volve a "Tarefa 2: A Calculadora da cesta da compra" e engade comentarios claros e utiles.
Pasos: 1. Abre o ficheiro da Tarefa 2.
2. Engade un comentario de bloque na parte superior do ficheiro explicando o que fai o script (ex: "Este script calcula o prezo final dun produto aplicando o IVE.").
3. Usa comentarios de liña única (//) para explicar o proposito de cada unha das variables clave ($prezo_unidade, $taxa_ive, $total).
4. Engade un comentario explicando o calculo mais importante, por exemplo, o do prezo final. O obxectivo non e que o codigo cambie o seu funcionamento, senon que sexa moito mais facil de entender para calquera persoa que o lea.
Parabens, completaches os fundamentos de PHP

```
/* Este script
calcula o prezo final dun produto aplicando o IVE. */

<!DOCTYPE html>
<html lang="gl">
<head>
  <meta charset="UTF-8">
  <title>Tarefa 2: A Calculadora da cesta da compra</title>
</head>
<body>
  <?php

    $prezo_unidade = 25; //Prezo da unidade do producto
    $cantidade = 4;//Cantidade do producto
```

```
    $subtotal = $prezo_unidade * $cantidade; //Subtotal sin IVA
    $taxa_ive = 0.21; //Tasa do IVA
    $ive_calculado = $subtotal * $taxa_ive; //Calculamos o IVA sobre o prezo
    $total = $subtotal + $ive_calculado; //Calculamos prezo total

    echo "Prezo por unidade: " . $prezo_unidade . " €<br>";
    echo "Cantidade: " . $cantidade . "<br>";
    echo "Subtotal: " . $subtotal . " €<br>";
    echo "IVE (21%): " . $ive_calculado . " €<br>";
    echo "Total a pagar: " . $total . " €";

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