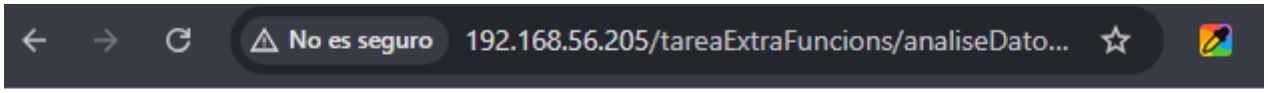
Exercicio 4: Análise de datos de alumnado con funcións de frecha e anónimas

- **Obxectivo:** Practicar o uso de funcións anónimas (`closures`) e funcións de frecha (`fn`) xunto con funcións de orde superior como `array_filter` e `array_map` para procesar conxuntos de datos.
- **Enunciado:** Dado o seguinte array de datos de estudantes, realiza as seguintes operacións en orde, encadeando as funcións cando sexa posible:
    0. Filtra o array para obter só os estudantes que aprobaron (nota maior ou igual a 5). Para este paso, utiliza unha **función anónima tradicional**.
    1. Do resultado anterior, extrae unicamente os nomes completos dos estudantes.
    2. Formatea os nomes para que aparezan como "Apelido, Nome" (por exemplo, "Outeiro Freire, Antía"). Para este paso, utiliza unha **función de frecha**.
    3. Finalmente, imprime por pantalla a lista de nomes formatados.
- **Datos de partida:**

```
<?php
$alumnado = [
    ['nome' => 'Antía', 'apelidos' => 'Outeiro Freire', 'nota' => 7.5],
    ['nome' => 'Breixo', 'apelidos' => 'Muiños Seoane', 'nota' => 4.9],
    ['nome' => 'Sabela', 'apelidos' => 'Castro Dopico', 'nota' => 9.1],
    ['nome' => 'Iago', 'apelidos' => 'Ferreiro Figueiras', 'nota' => 3.2],
    ['nome' => 'Xoel', 'apelidos' => 'Vázquez Regueiro', 'nota' => 6.0],
];

// O código para resolver o exercicio vai aquí
?>
```

```php
<?php

$alumnado = [
  ['nome' => 'Antía', 'apelidos' => 'Outeiro Freire', 'nota' => 7.5],
  ['nome' => 'Breixo', 'apelidos' => 'Muiños Seoane', 'nota' => 4.9],
  ['nome' => 'Sabela', 'apelidos' => 'Castro Dopico', 'nota' => 9.1],
  ['nome' => 'Iago', 'apelidos' => 'Ferreiro Figueiras', 'nota' => 3.2],
  ['nome' => 'Xoel', 'apelidos' => 'Vázquez Regueiro', 'nota' => 6.0],
];

$aprobados = array_filter($alumnado, function ($alumno) {
  return $alumno['nota'] >= 5;
});

$nombresCompletos = [];

foreach ($aprobados as $alumno) {
  $nombresCompletos[] = [
    'nome' => $alumno['nome'],
    'apelidos' => $alumno['apelidos']
  ];
}
```

```php
$nomesFormateados = array_map(fn($alumno) =>
    $alumno['apelidos'] . ', ' . $alumno['nome'],
  $nombresCompletos);

  foreach($nomesFormateados as $nome){
    echo $nome. "</br>";
  }

?>
```



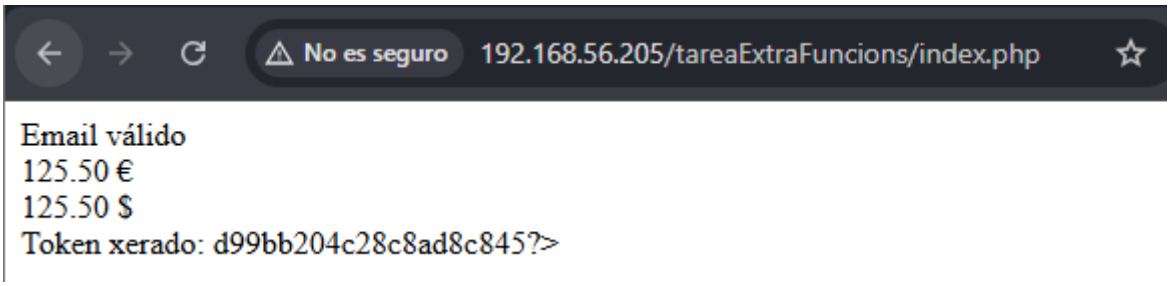
Outeiro Freire, Antía
Castro Dopico, Sabela
Vázquez Regueiro, Xoel

Exercicio 5: Creación dunha libraría de utilidades e xestión de ficheiros

- **Obxectivo:** Entender a importancia de organizar o código en ficheiros e utilizar `require_once` para incluír dependencias esenciais de forma segura.
- **Enunciado:**
    0. Crea un ficheiro chamado `utilidades.php`. Dentro del, define as seguintes tres funcións:
        - `validarEmail(string $email): bool`: Devolve `true` se o email contén un carácter `@` e `false` no caso contrario (unha validación sinxela é suficiente).
        - `formatarMoeda(float $cantidade, string $simbolo = '€'): string`: Devolve un string coa cantidade formatada con dous decimais e o símbolo da moeda ao final (ex: `125.50 €`). O símbolo debe ser un parámetro opcional.
        - `xerarTokenSeguro(int $lonxitude = 16): string`: Xera unha cadea aleatoria de texto da lonxitude especificada. Podes investigar a función `bin2hex(random_bytes($lonxitude / 2))` para isto.
    1. Crea un segundo ficheiro chamado `index.php`.
    2. En `index.php`, inclúe o ficheiro `utilidades.php` usando a instrución máis segura para dependencias críticas.
    3. Proba cada unha das funcións da túa libraría con datos de exemplo e imprime os resultados para verificar que funcionan correctamente.

```php
<?php
function validarEmail($email){
  if(str_contains($email,'@')){
    return true;
  }else{
    return false;
  }
}

function formatarMoeda(float $cantidade, string $simbolo = '€'): string {
  $formateada = number_format($cantidade, 2, '.', '');
  return $formateada . ' ' . $simbolo;
}

function xerarTokenSeguro($lonxitude=16){
  return bin2hex(random_bytes($lonxitude / 2));
}

?>
```

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

$emailExemplo = "usuario@exemplo.com";
$cantidadeExemplo = 125.5;
$tokenLonxitude = 20;

echo validarEmail($emailExemplo) ? "Email válido<br>" : "Email non válido<br>";
echo formatarMoeda($cantidadeExemplo);
echo "<br>";
echo formatarMoeda($cantidadeExemplo, '$');
echo "<br>";
echo "Token xerado: " . xerarTokenSeguro($tokenLonxitude);
?>
```

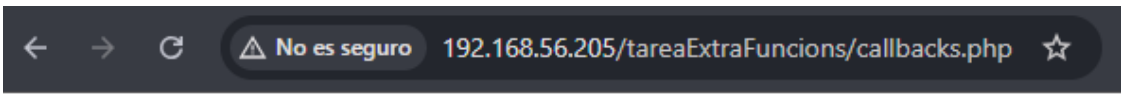


Exercicio 6: Procesador xenérico con Callbacks

- **Obxectivo:** Deseñar unha función xenérica que acepte un *callback* para delegar unha parte da súa lóxica, reforzando o concepto de "inversión de control".
- **Enunciado:**
    0. Crea unha función principal chamada `procesarArray(array $datos, callable $operacion)`. Esta función debe percorrer cada elemento do array `$datos`, aplicarlle a función `$operacion` a cada un, e devolver un novo array cos resultados.
    1. Define tres funcións diferentes que poidan ser usadas como *callbacks*:
        - `elevarAoCadrado(int $numero): int`: Devolve o cadrado do número.
        - `converterAMaiusculas(string $cadea): string`: Converte o texto a maiúsculas.
        - `obterLonxitude(string $cadea): int`: Devolve a lonxitude do texto.
    2. Declara dous arrays de exemplo: un con números e outro con textos.
    3. Chama á túa función `procesarArray` varias veces para:
        - Elevar ao cadrado todos os elementos do array de números.
        - Converter a maiúsculas todos os elementos do array de textos.
        - Obter a lonxitude de cada elemento do array de textos.
    4. Imprime os resultados de cada chamada para amosar a flexibilidade da túa función principal.
- **Pista:** Asegúrate de usar o *type hint* `callable` na definición de `procesarArray` para garantir que o segundo argumento sexa sempre unha función executable.

```php
<?php
function procesarArray(array $datos, callable $operacion){
  $resultados = [];
  foreach($datos as $dato){
    $resultados[] = $operacion($dato);
  }
  return $resultados;
}
function elevarAoCadrado(int $numero): int{
  return $numero*$numero;
}
function converterAMaiusculas(string $cadea): string{
  return strtoupper($cadea);
}
function obterLonxitude(string $cadea): int{
  return strlen($cadea);
```

```php
}

$numeros=[1,2,3,4,5,6,7,8,9,10];
$textos=["Carlos","Hola","Servidor","Web"];

$cadrados=procesarArray($numeros,'elevarAoCadrado');
$maiusculas=procesarArray($textos,'converterAMaiusculas');
$lonxitudes=procesarArray($textos,'obterLonxitude');

echo "<h2>Números elevados: </h2>"."</br>";
foreach($cadrados as $cadrado){
  echo $cadrado."</br>";
}

echo "<h2>Textos en maiuscula: </h2>"."</br>";
foreach($maiusculas as $texto){
  echo $texto."</br>";
}

echo "<h2>Lonxitude do texto: </h2>"."</br>";
foreach($lonxitudes as $texto){
  echo $texto."</br>";
}

?>
```



Números elevados:

1
4
9
16
25
36
49
64
81
100

Textos en maiuscula:

CARLOS
HOLA
SERVIDOR
WEB

Lonxitude do texto:

6
4
8
3

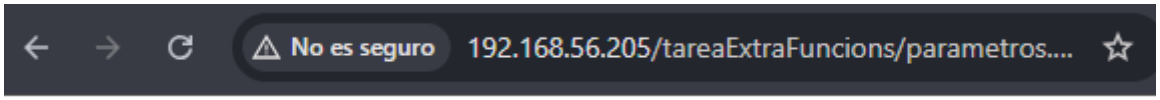
Exercicio 7: Parámetros por referencia para estatísticas

- **Obxectivo:** Comprender o uso práctico de pasar parámetros por referencia para que unha función poida modificar variables fóra do seu propio ámbito.
- **Enunciado:** Queres crear unha función que calcule varias estatísticas dunha lista de números sen necesidade de devolver un array con todos os resultados.
    0. Define unha función chamada `calcularEstatisticas(array $numeros, &$total, &$media)`.
    1. Esta función non debe devolver ningún valor coa instrución `return`.
    2. Dentro da función, calcula a suma total dos números e o seu valor medio.
    3. Asigna o resultado da suma á variable que se pasou por referencia como `$total`.
    4. Asigna o resultado do cálculo da media á variable que se pasou por referencia como `$media`.
    5. Fóra da función, inicializa as variables `$sumaTotal` e `$mediaCalculada` a `0`.
    6. Chama á función `calcularEstatisticas` pasándolle un array de números e as dúas variables que acabas de inicializar.
    7. Despois da chamada, imprime os valores de `$sumaTotal` e `$mediaCalculada` para comprobar que foron modificados correctamente pola función.
- **Datos de partida:**

```
<?php
// Define aquí a túa función calcularEstatisticas

$puntuacions = [8.5, 9.0, 7.2, 6.8, 9.5, 8.1];
$sumaTotal = 0;
$mediaCalculada = 0;

// Chama á función aquí

echo "A suma total das puntuacións é: " . $sumaTotal . "\n";
echo "A media calculada é: " . $mediaCalculada . "\n";
?>
```

```php
<?php
   // Define aquí a túa función calcularEstatisticas
   function calcularEstatisticas(array $numeros, &$total, &$media) {
     $total = array_sum($numeros);
     $media = $total / count($numeros);
   }
   $puntuacions = [8.5, 9.0, 7.2, 6.8, 9.5, 8.1];
   $sumaTotal = 0;
   $mediaCalculada = 0;

   // Chama á función aquí
   calcularEstatisticas($puntuacions, $sumaTotal, $mediaCalculada);
```

```php
    echo "A suma total das puntuacións é: " . $sumaTotal. "<br>";
    echo "A media calculada é: " . $mediaCalculada . "<br>";
?>
```



A suma total das puntuacións é: 49.1
A media calculada é: 8.1833333333333