

Abstract geometric shapes in the top-left corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a pink parallelogram, all overlapping and tilted at various angles.

# **LARAVEL**

**AULA 06**

Abstract geometric shapes in the top-right corner, including a green parallelogram, a blue parallelogram, an orange parallelogram, and a pink parallelogram, all overlapping and tilted at various angles.

The slide features a light gray background with abstract geometric shapes in the corners. On the left, there are overlapping shapes in shades of green, blue, orange, and purple. On the right, there are similar shapes in shades of green, blue, purple, and orange. The main content is centered on the slide.

# PAGINAÇÃO

Como criar uma paginação em Laravel?

Laravel oferece um método de paginação que pode ser usado tanto em query builders quanto em Eloquent.

```
// Neste exemplo, estamos criando uma paginação  
// de 15 produtos por página.
```

```
// Exemplo com Eloquent  
$produtos = App\Product::paginate(15);
```

```
// Exemplo com Query Builder  
$produtos = App\Product::where('price', '>', 100)->paginate(15);
```

Para mostrar o resultado em uma visualização, basta iterar através da coleção de produtos, assim como faríamos normalmente. Por último, para adicionar os botões de paginação, usamos o método `->links()`;

```
@foreach ($products as $product)
```

```
  {{ $product->title }}
```

```
@endforeach
```

```
  {{ $products->links() }}
```

Abstract geometric shapes in the top-left corner, including a green triangle, a blue parallelogram, an orange parallelogram, and a pink parallelogram.

# COLEÇÕES

Uso de coleções em Laravel

Abstract geometric shapes in the top-right corner, including a green triangle, a blue parallelogram, a pink parallelogram, and an orange parallelogram.

Quando utilizamos eloquent, muitos resultados retornam um objeto do tipo [Collection](#).

À primeira vista, as coleções são muito parecidas aos arrays, mas gostamos de dizer que as coleções são “*arrays bombados*”.

Sempre que existe um [array](#), podemos [obter](#) facilmente uma [coleção](#), e assim utilizar seus métodos.

```
$collection = collect($array);
```

```
Collection {#157 ▼  
  #items: array:5 [▼  
    0 => 1  
    1 => 2  
    2 => 3  
    3 => 4  
    4 => 5  
  ]  
}
```

# Métodos de coleções:

```
$pessoa = collect([  
    ['nome' => 'nick',      'idade' => 32],  
    ['nome' => 'daniel',    'idade' => 15],  
    ['nome' => 'francisco', 'idade' => 22],  
]);
```

```
$numeros = collect([1, 2, 3, 4, 5]);
```

// Alguns exemplos

```
$ListaDeNomes = $pessoa->implode('nome', ', '); // string: "nick, daniel, francisco"
```

```
$numeros->push(6); // Adicionou 6 ao final: [1,2,3,4,5,6]
```

```
$ultimo = $numeros->pop(); // Retorna 6 (o último)  
// E tira da coleção de $numbers
```

```
$pessoaJson = $pessoa->toJson(); // string: '[{"nome": "nick", "idade": 32},{...}'
```

```
$nomes = $pessoa->pluck('nome'); // nova Collection: ['nick', 'daniel', 'francisco']
```

```
// Retorna todos os elementos de uma coleção (o array “interno”)

$inteiros = $numeros->all();

// Retorna o primeiro/último elemento de uma coleção (sem modificar)

$um = $numeros->first();           // 1
$cinco = $numeros->last();          // 5

// Organiza os elementos de uma coleção (em ordem alfabética ou crescente)

$ordenar = collect([5, 3, 4, 1, 2])->sort();           // nova Collection: [1, 2, 3, 4, 5]

// Filtrar valores

$maiorQueDois = $numeros->filter(function ($value) { // nova Collection: [3, 4, 5]
    Return $value > 2;
});

// Realizar uma ação para cada um dos valores

$keyMaisValue = $numeros->map(function ($value, $key) { // nova Collection: [1, 3, 5, 7, 9]
    Return $key + $value;
});
```



The background features abstract, overlapping geometric shapes in various colors including light blue, green, cyan, magenta, orange, and red. These shapes are arranged in a way that creates a sense of depth and movement, framing the central text.

**ATÉ A  
PRÓXIMA!**