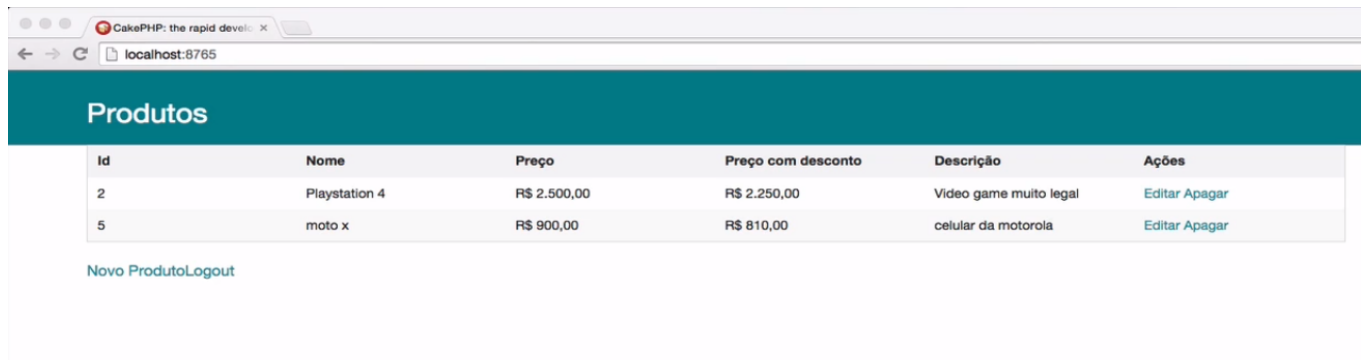


## Paginação

### Capítulo 1 - Paginação

Sejam bem-vindos ao nosso segundo curso de **Cake PHP**! Neste curso continuaremos o projeto do primeiro adicionando alguma funcionalidades que o framework nos disponibiliza para melhorar ainda mais.

Lembre-se que tínhamos uma listagem com dois produtos cadastrados:



Id	Nome	Preço	Preço com desconto	Descrição	Ações
2	Playstation 4	R\$ 2.500,00	R\$ 2.250,00	Video game muito legal	<a href="#">Editar</a> <a href="#">Apagar</a>
5	moto x	R\$ 900,00	R\$ 810,00	celular da motorola	<a href="#">Editar</a> <a href="#">Apagar</a>

[Novo Produto](#) [Logout](#)

Mas e se tivéssemos cadastrados 100 produtos? Neste momento, todos estariam em uma página apenas, seria difícil e confusa a visualização. Além disso todas essas informações podem fazer com que a página demore para carregar. A solução para isso é montar um sistema de paginação. Esta feature também deixará a página mais organizada.

Se queremos, então, fazer a paginação dos produtos, devemos mexer no "ProdutosController.php". O controle é feito pela função `index`, que nesse momento está assim:

```
public function index() {  
    $produtosTable = TableRegistry::get('Produtos');  
    $produtos = $produtosTable->find('all');  
    $this->set('produtos', $produtos);  
}
```

Precisamos escrever um método dentro do `this` indicando que na hora de mandar para a *view*, os produtos sejam paginados, o nome de é `paginate`:

```
$this->set('produtos', $this->paginate($produtos);
```

O `paginate` não vem habilitado por padrão, para habilitá-lo, escrevemos o método de sua inicialização:

```
class ProdutosController extends AppController {  
  
    public function initialize() {  
        parent::initialize();  
  
        $this->loadComponent('Paginator')  
    }  
}
```

Perceba que chamamos o método de inicialização da classe pai ( `AppController` ) e pedimos para carregar o componente que faz a paginação, o `Paginator` .

Como nós temos apenas dois produtos cadastrados, não fará muita diferença agora, até porque não indicamos quantos produtos queremos por página. Fazemos isso preenchendo um atributo dentro do `ProdutosController` :

```
class ProdutosController extends AppController {  
  
    public $paginate = [  
        'limit' => 1  
    ];  
}
```

Pedimos que apareça apenas um registro por página. Agora se formos na página:



De fato, só nos aparece um produto. Para ir para a segunda página devemos acrescentar o parâmetro `?page=2` na URL:

`localhost:8765/Produtos?page=2`

A mesma coisa para o primeiro: `?page=1` . Nossa paginação está funcionando. Porém, o usuário não saberá disso. É necessário criar uma lista com o número das páginas para navegar entre elas. Criemos, então, essa listagem. Para isso precisamos mexer no HTML, ou seja, vamos para "Template/Produtos/index.ctp" e adicionamos a parte de paginação no fim do código:

```
<?php  
echo $this->Paginator->prev('Voltar');  
echo $this->Paginator->next('Avançar');  
?>
```

Criamos um link para voltar (método `prev()` ) e avançar (método `next()` ) uma página. E agora sim, temos os links funcionando:



Vamos também mostrar quantas páginas temos para navegar:

```
<?php
    echo $this->Paginator->prev('Voltar');
    echo $this->Paginator->numbers();
    echo $this->Paginator->next('Avançar');
?>
```



Porém ainda não temos nenhum estilo. Vamos usar o Cake PHP para fazer isso adicionando duas classes que farão esse trabalho, `paginator` e `pagination` :

```
<div class="paginator">
    <ul class="pagination">
        <?php
            echo $this->Paginator->prev('Voltar');
            echo $this->Paginator->numbers();
            echo $this->Paginator->next('Avançar');
        ?>
    </ul>
</div>
```

Atualizando a página:



E é assim que fazemos paginação de dados utilizando Cake PHP.