Já vimos como fazer um CRUD completo utilizando os Models do Laravel.
E aprendemos também o que devemos configurar para que o Framework possa relacionar um registro de uma tabela com a outra, sem a necessidade de ficar criando aqueles JOINS manualmente.

Sabendo disso, vamos aprender um truque bacana que o Laravel permite fazer facilmente utilizando o Model: a paginação.

1. Paginação

Muitas vezes você vai sentir a necessidade de apresentar apenas alguns registros de cada vez para o usuário, e não tudo em uma única página.

Isso é bastante útil quando quando você possui milhares de registros para serem apresentados. Ou quando as informações são acessadas através de um celular ou tablet, onde a velocidade da internet pode não ser tão agradável =)

Quando isso acontece, um recurso muito utilizado para contornar esse problema é a paginação. Ou seja, exibir uma determinada lista de registros separados por página.

Já pensou em fazer isso em ajuda do Laravel?
De alguma maneira você teria que armazenar qual foi o último registro, e quando for exibir outra página apresentar apenas aqueles que ainda não foram apresentados para o usuário.

Isso parece relativamente simples quando o usuário navega página por página na sequência.
Mas e se o usuário quiser acessar diretamente a página 3, por exemplo?
Ou então se ele resolver visualizar os registros usando outra ordem?

O Laravel, usando os modelos do Eloquent, já resolve isso tudo pra gente.

Primeiro precisamos utilizar o método `paginate` lá no controlador:

```
$produtos = Produto::paginate(15);
```

E depois lá na view do Blade basta colocar essa linha de código depois do seu loop (foreach, por exemplo):

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

Fazendo dessa maneira, por exemplo, o Laravel já saberá que cada página deverá ter 15 registros (ou quantos você indicar no parâmetro do `paginate`).

E além de calcular quantas páginas serão necessárias, ele também já vai incluir no final da sua lista os links necessários para que o usuário possa navegar entre eles.

2. Coleções

Lembra que o Laravel através dos modelos consegue acessar uma tabela do seu banco de dados e selecionar os registros que desejar?

Internamente o Framework utiliza a biblioteca chamada Eloquent para fazer isso acontecer.

E quando as informações da tabela do banco são localizadas o Laravel não devolve um array para que você possa utilizar, mas sim uma Collection (ou coleção).

A coleção de um array turbinado que possui vários métodos para facilitar a nossa vida.
Na verdade a coleção é um objeto, e não um array =)

Se você quiser também é possível converter um array em uma collection.
Para isso basta utilizar a função `collect`.
Olha só:

```php
// Este é um array com algumas pessoas
$pessoas = [
      ['nome' => 'nick',      'idade' => 32],
      ['nome' => 'daniel',    'idade' => 15],
      ['nome' => 'francisco', 'idade' => 22],
];

// Agora vamos transformar o array em uma collection
$colecao = collect($pessoas);
```

Se quiser conferir todos os métodos que uma coleção permite utilizar basta dar uma olhada na documentação do Laravel: https://laravel.com/docs/5.7/collections#available-methods

Vamos lembrar algumas delas aqui:

```php
$listaDeNomes = $pessoas->implode('nome', ', ');
// O resultado seria uma string com: "nick, daniel, francisco"

$arrayDeNomes = $pessoas->pluck('nome');
// O resultado seria uma nova collection apenas com os nomes:
['nick','daniel','francisco']

$ultimaPessoa = $pessoas->pop();
// Obtém a última pessoa do collection, e remove ao mesmo tempo essa pessoa

$pessoas->push(['nome' => 'joao', 'idade' => 10]);
// Adiciona também o 'joao' na collection de pessoas

$pessoasOrdenadas = $pessoas->sortBy('idade');
// Ordena a collection pessoas por idade
```