LARAVEL

CLASE 06

PAGINACIÓN

¿Cómo crear una paginación en Laravel?

Laravel nos brinda un método de paginado que podemos usar tanto a un query builder como a Eloquent.

```
// En este ejemplo estamos creando una paginación
// de 15 productos por página.


// Ejemplo con Eloquent
$productos = App\Product::paginate(15);


// Ejemplo con Query Builder
$productos = App\Product::where(price, '>', 100)->paginate(15);
```

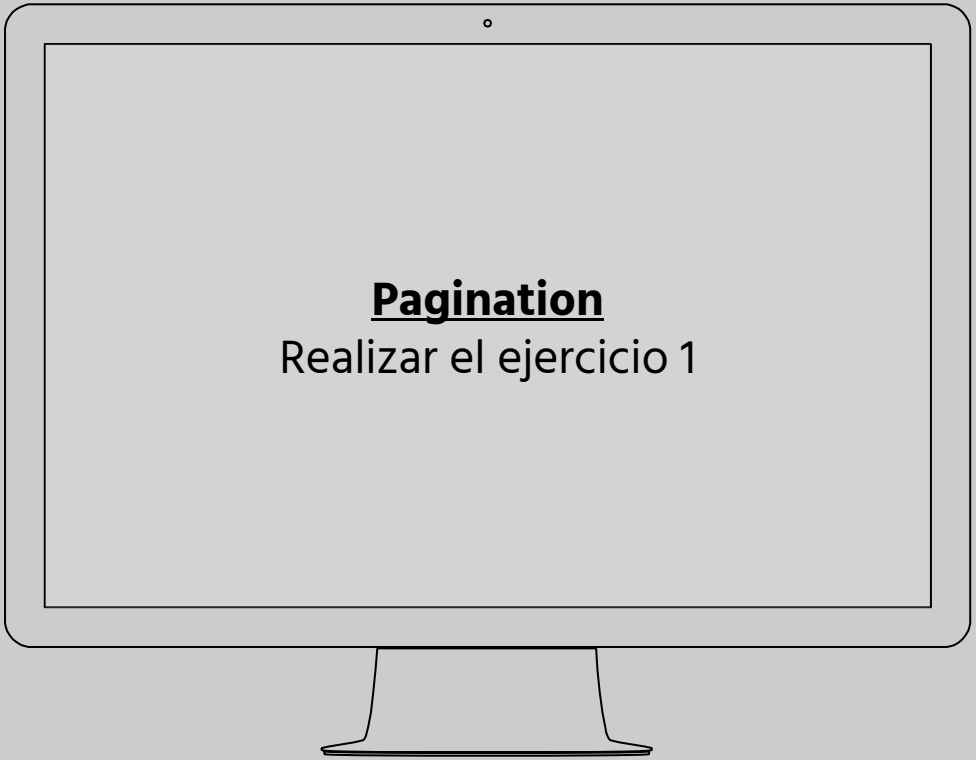
Para mostrar los resultado en una vista simplemente tenemos que iterar a través de la colección de

productos, tal como lo haríamos normalmente. Por último, para agregar los botones de paginado

usamos el método ->links();

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

    {{ $product->title }}

@endforeach


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

ES MOMENTO DE
PRACTICAR !



COLECCIONES

Uso de colecciones en Laravel

Muchos de nuestros resultados a la hora de utilizar eloquent, nos devuelven un objeto de tipo Collection.

Si bien a primera vista nos parece muy similar a los array, nos gusta decir que las colecciones son *"arrays con esteroides"*.

Siempre que tengamos un array, podremos obtener fácilmente a una colección, y así disponer de sus métodos.

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

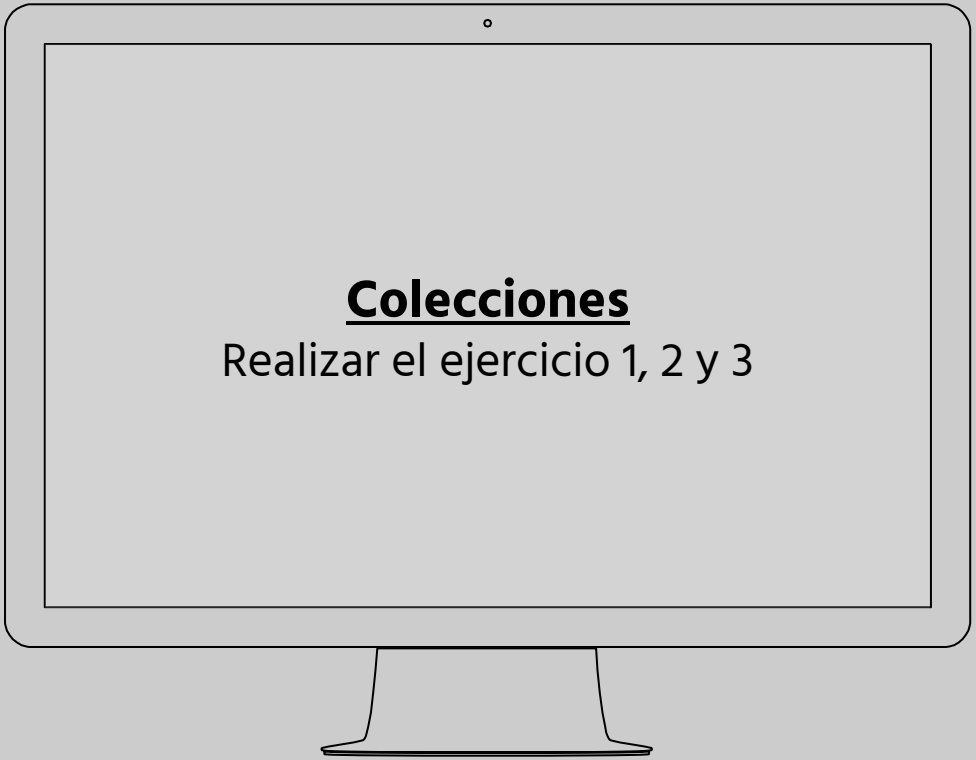
```
$collection = collect($array);
```

Métodos de colecciones:

```
$people = collect([
      ['name' => 'nick',      'age' => 32],
      ['name' => 'daniel',    'age' => 15],
      ['name' => 'francisco', 'age' => 22],
]);

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

//    Algunos ejemplos

$nameList = $people->implode('name', ', ');     // string: "nick, daniel, francisco"

$numbers->push(6);                               // Agregó el 6 al final: [1,2,3,4,5,6]

$last = $numbers->pop();                         // Devuelve 6 (el último)
                                                 // Y lo quita de la colección de $numbers



$peopleJson = $people->toJson();                 // string: '[{"name": "nick", "age": 32},{...'

$names = $people->pluck('name');                 // nueva Collection: ['nick', 'daniel', 'francisco']
```

```
//    Devuelve todos los elementos de una colección (el array “interno”)

$integers = $numbers->all();

//    Devuelve el primer/último elemento de una colección (no la modifica)

$one = $numbers->first();           // 1
$five = $numbers->last();           // 5

//    Ordena los elementos de una colección (alfabéticamente o ascendente)

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

//    Filtrar valores

$greaterThanTwo = $numbers->filter(function ($value) {      // nueva Collection: [3, 4, 5]
      Return $value > 2;
});

//    Hacer una acción para cada uno de todos los valores

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

ES MOMENTO DE
PRACTICAR !



STORAGE

Uso de storage en Laravel

La carpeta storage

Usamos la carpeta storage para guardar archivos que se generan por la aplicación:

- Archivos compilados por el framework
- Logs de errores
- Session / Cache
- Uploads

Uploads, storage y public

La carpeta storage no es públicamente visible. Para que un upload se pueda ver desde la web, tenemos que hacerlo visible:

```
php artisan storage:link
```

En Windows:

Almacenamiento de imágenes:

```php
public function store(Request $request)
{
    $user = $request->user();

    // carpeta en la que voy a guardar la imagen
    $folder = "avatars";

    // Laravel usará un nombre al azar y nos lo dará en $path
    $path = $request->file("avatar")->storePublicly($folder);

    // Debo guardarlo en base de datos...
    $user->avatar = $path;
    $user->save();
}
```

Almacenamiento de imágenes:

```php
public function store(Request $request)
{
    $user = $request->user();

    // Necesito el archivo en una variable esta vez
    $file = $request->file("avatar");

    // Armo un nombre único para este archivo
    $name = $user->id . "." . $file->extension();

    $folder = "avatars";
    $path = $file->storePubliclyAs($folder, $name);

    // Puedo igual guardarlo en base de datos...
    $user->avatar = $path;
    $user->save();
}
```

¡ HASTA LA PROXIMA !