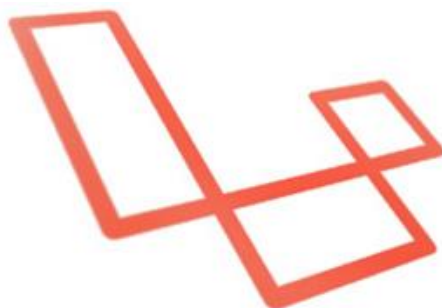


Desarrollo de Aplicaciones Empresariales

LABORATORIO N° 05

Laravel - Requests

CODIGO DEL CURSO:



laravel

Alumno(s)		Nota
Grupo		
Ciclo		
Fecha de entrega		

I.- OBJETIVOS:

- Crear funcionalidad en la aplicación del Gestor de Imágenes.

II.- SEGURIDAD:**Advertencia:**

En este laboratorio está prohibida la manipulación del hardware, conexiones eléctricas o de red; así como la ingestión de alimentos o bebidas.

III.- FUNDAMENTO TEÓRICO:

Revise sus diapositivas del tema antes del desarrollo del laboratorio.

IV.- NORMAS EMPLEADAS:

No aplica

V.- RECURSOS:

- En este laboratorio cada alumno trabajará con un equipo con Windows 8.

VI.- METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:

- El desarrollo del laboratorio es individual.

VII.- PROCEDIMIENTO:**Nota:**

Las secciones en cursivas son demostrativas, pero sirven para que usted pueda instalar las herramientas de desarrollo en un equipo externo.

CREANDO LA FUNCIONALIDAD: ACTUALIZAR PERFIL DE USUARIO

Ahora procederemos a crear la funcionalidad para actualizar los campos del usuario, es decir, darle el derecho de actualizar el perfil.

1. Cree una carpeta dentro de la carpeta de vistas (views) con el nombre “usuario”.
2. Dentro de la carpeta creada, cree un archivo de nombre “actualizar.blade.php”.
3. A este archivo copie el código de la vista **Registro**.
4. Pero antes, ubique la función **getEditarPerfil()**, en el controlador del usuario y modifíquelo para que se haga uso de la vista que hemos creado.

```

13
14     public function getEditarPerfil(){
15         return view('usuario.actualizar');
16     }
17     public function postEditarPerfil(){

```

5. Abra el archivo de la vista principal “app.blade.php”, y agregue la siguiente línea para darle opción de editar al usuario en caso de que éste se encuentre loggeado.

```

44 <li class="dropdown">
45     <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-expanded="false">{{ A
46     <span class="caret"></span></a>
47     <ul class="dropdown-menu" role="menu">
48         <li><a href="{{ url('/validado/usuario/editar-perfil') }}">Actualizar Perfil</a></li>
49         <li><a href="{{ url('/validacion/salida') }}">Salir</a></li>
50     </ul>
</li>

```

6. Ahora modifique la vista actualizar.blade.php para que quede de la siguiente manera:

```

6     <div class="col-md-8 col-md-offset-2">
7         <div class="panel panel-default">
8             <div class="panel-heading">Actualizar perfil</div>
9             <div class="panel-body">
10                 @if (count($errors) > 0)

```

```

20
21 <form class="form-horizontal" role="form" method="POST" action="/validado/usuario/editar-perfil">
22 <input type="hidden" name="_token" value="{{ csrf_token() }}">
23
24 <div class="form-group">
25 <label class="col-md-4 control-label">Nombre</label>
26 <div class="col-md-6"> +
27 <input type="text" class="form-control" name="nombre" value="{{ Auth::user()->nombre }}">
28 </div>
29 </div>
30
31 <div class="form-group">
32 <label class="col-md-4 control-label">Contraseña</label>
33 <div class="col-md-6">
34 <input type="password" class="form-control" name="password">
35 </div>

```

Borrar el campo email

```

43 </div>
44
45 <div class="form-group">
46 <label class="col-md-4 control-label">Pregunta</label>
47 <div class="col-md-6">
48 <input type="text" class="form-control" name="pregunta" value="{{ Auth::user()->pregunta }}">
49 </div>
50 </div>
51
52 <div class="form-group">
53 <label class="col-md-4 control-label">Respuesta</label>
54 <div class="col-md-6">
55 <input type="text" class="form-control" name="respuesta" value="{{ Auth::user()->respuesta }}">
56 </div>
57 </div>
58

```

```

58
59
60 <div class="form-group">
61 <div class="col-md-6 col-md-offset-4">
62 <button type="submit" class="btn btn-primary">
63 Actualizar
64 </button>
65 </div>

```

- Ahora ingrese al sistema, y escoja la opción actualizar perfil, el resultado debería ser parecido al siguiente:

The screenshot shows a web browser window with the URL 'gestorlaravel.com/validado/usuario/editar-perfil'. The page title is 'Laravel Inicio'. The main content is a form titled 'Actualizar perfil'. The form has the following fields: 'Nombre' with the value 'Jorge', 'Contraseña', 'Confirm Password', 'Pregunta' with the value 'preg', and 'Respuesta' with the value 'resp'. At the bottom of the form is a blue button labeled 'Registrarse'.

Hasta ahora hemos implementado la funcionalidad del verbo GET al momento de querer actualizar el perfil de usuario, falta la funcionalidad del verbo POST, lo haremos de la siguiente manera.

- Cree un Request nuevo con la consola de comandos de nombre: "EditarPerfilRequest".
- En el nuevo Request creado ponga en true el valor de retorno de la función authorize() y añada las reglas como se muestra en la imagen a continuación.

```

11     */
12     public function authorize()
13     {
14         return true;
15     }
16

```

```

22     public function rules()
23     {
24         return [
25             'nombre' => 'required',
26             'password' => 'min:6|confirmed',
27             'pregunta' => '',
28             'respuesta' => 'required_with:pregunta',
29         ];
30     }

```

10. En el controlador del usuario, importe el request creado en el punto anterior:

```

1  <?php namespace GestorImagenes\Http\Controllers;
2  use GestorImagenes\Http\Requests\EditarPerfilRequest;
3  class UsuarioController extends Controller {
4      /**
5       * Create a new controller instance

```

11. Referencie también a la clase Auth, ya que ésta contiene la información del usuario loggeado:

```

3  use GestorImagenes\Http\Requests\EditarPerfilRequest;
4  use Illuminate\Support\Facades\Auth;

```

12. Ahora modifique el contenido de su función postEditarPerfil(), para que quede de la siguiente manera:

```

20     public function postEditarPerfil(EditarPerfilRequest $request){
21
22         $usuario=Auth::user();
23         $nombre=$request->get('nombre');
24         $usuario->nombre=$nombre;
25
26         if ($request->has('password')) {
27             $usuario->password=bcrypt($request->get('password'));
28         }
29         if ($request->has('pregunta')) {
30             $usuario->pregunta=$request->get('pregunta');
31             $usuario->respuesta=$request->get('respuesta');
32         }
33         $usuario->save();
34         return redirect('/validado')->with('actualizado','Su perfil ha sido actualizado');
35     }

```

Hemos importado la librería Auth ya que ésta, contiene la información del usuario autenticado, de hecho la primera variable viene a ser del tipo usuario.

Luego recuperamos el nombre del request, ya que es requerido (y de ese trabajo de validaciones se encarga nuestro request).

El usuario puede o no cambiar el nombre, pero la única manera de comprobar o asegurarnos de que la información sea procesada de manera correcta es asignándole el nuevo nombre que recibimos del request al atributo nombre del usuario recibido (en un caso común, si el usuario no desea cambiar su nombre, lo dejará con el mismo valor y no tendremos inconvenientes).

El primer if, es para ver si es que nuestro request tiene lleno el campo de password, ya que en caso de que así fuera, asignemos el valor recibido, solo que encriptado al objeto usuario.

El segundo if, analiza si el request tiene lleno el campo de pregunta (no olvidemos que el request creado se encarga de analizar que si hay pregunta, tiene que haber respuesta), y ambos valores son asignados al objeto usuario.

Finalmente se hace uso de la función save(), para actualizar al objeto usuario en la base de datos y la redirección se hará validado que si observamos en nuestra hoja de rutas se traduce como InicioController.

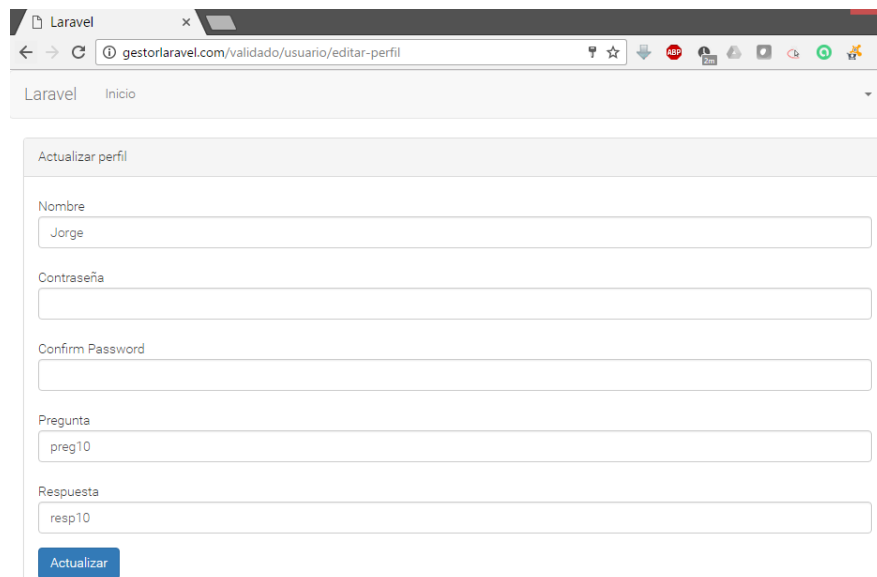
Cuando se hace la redirección, se envía un mensaje de: **Su perfil ha si.....**, encapsulado en la variable **actualizado** Para recibir e imprimir eso en la vista correspondiente debemos hacer lo siguiente:

13. Abra la vista Inicio y agregue el código para imprimir el mensaje de cambios realizados correctamente.

```
4
5 @if(Session::has('error'))
6     <div class="alert alert-danger">
7         <strong>Whoops!</strong> Al parecer algo está mal joven <br><br>
8         {{Session::get('error')}}
9     </div>
10 @endif
11 @if(Session::has('actualizado'))
12     <div class="alert alert-success">
13         <strong>Hecho!</strong> Cambios realizados <br><br>
14         {{Session::get('actualizado')}}
15     </div>
16 @endif
17
```

14. Para comprobar, inicie sesión, actualice su perfil como convenga y el resultado debería ser el siguiente:

EJEMPLO:



The screenshot shows a web browser window with the URL `gestorlaravel.com/validado/usuario/editar-perfil`. The page title is "Laravel Inicio". The main content area is titled "Actualizar perfil" and contains several input fields: "Nombre" (with the value "Jorge"), "Contraseña", "Confirm Password", "Pregunta" (with the value "preg10"), and "Respuesta" (with the value "resp10"). At the bottom of the form is a blue button labeled "Actualizar".



The screenshot shows the same web browser window after the profile update. The URL is now `gestorlaravel.com/validado`. A green alert box is displayed at the top of the page with the text "Hecho! Cambios realizados" and "Su perfil ha sido actualizado". Below the alert box, the page shows a navigation bar with "Inicio" and a welcome message "Bienvenido Jorge".

CREANDO LA FUNCIONALIDAD DE MOSTRAR ÁLBUMES

En esta sección vamos a crear la funcionalidad que tiene el sistema de mostrar los álbumes que posee un usuario.

IMPORTANTE: Existen algunos pasos, como creación de requests, importación de librerías, etc., que por el avance del curso se asume el participante domina, en tal sentido preste mucha atención a la hora de seguir los pasos del laboratorio.

1. Abra el controlador de Album Controller
2. Ubique la función por defecto (getIndex()).
3. Importaremos la definición de Auth para obtener los datos del usuario loggeado y modificaremos el action para que quede de la siguiente manera:

```
1  <?php namespace GestorImagenes\Http\Controllers;  
2  
3  use Illuminate\Support\Facades\Auth;  
4  
5  class AlbumController extends Controller {  
6      /**  
7       * Create a new controller instance.  
8       *  
9       * @return void  
10      */  
11     public function __construct()  
12     {  
13         $this->middleware('auth');  
14     }  
15     public function getIndex(){  
16         $usuario=Auth::user();  
17         $albumes=$usuario->albumes;  
18         return view('albumes.mostrar',['albumes'=>$albumes]);  
19     }  
20     /**  
21     * Show the application dashboard to the user
```

Destacamos que \$albumes es una colección de objetos de tipo álbum y que ahora se encuentran en la variable \$albumes. Además, se retorna este arreglo mediante un array asociativo con la variable álbumes a la vista **mostrar** dentro de una carpeta **álbumes** (que aún no hemos creado).

4. Antes ubique la vista principal (app.blade.php), para agregar el link de ver los álbumes de un usuario.

```
33  
34     <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">  
35         <ul class="nav navbar-nav">  
36             <li><a href="{{ url('/') }}">Inicio</a></li>  
37             <li><a href="{{ url('/validado/albumes') }}">Mis Álbumes</a></li>  
38         </ul>  
39  
40         <ul class="nav navbar-nav navbar-right">  
41             @if (Auth::request())
```

5. Ahora, cree la vista correspondiente para usar la funcionalidad mencionada. Cree la carpeta albumes y dentro de ella, la vista mostrar.blade.php. El código a crear será el siguiente:

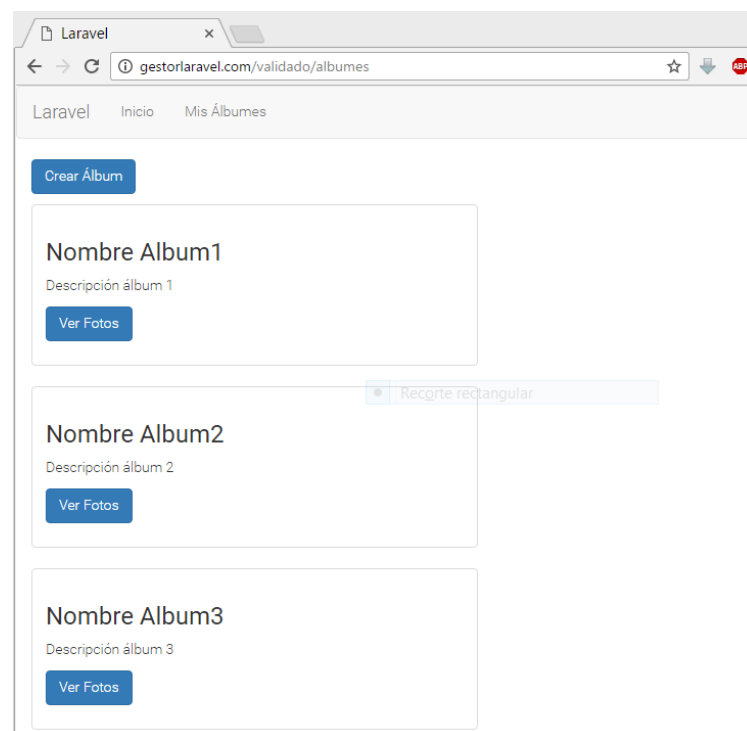
```
1 @extends('app')
2
3 @section('content')
4 <div class="container-fluid">
5 <p><a href="/validado/albumes/crear-album" class="btn btn-primary" role="button">Crear Álbum</a></p>
6 @if(sizeof($albumes)>0)
7     @foreach($albumes as $album)
8         <div class="row">
9             <div class="col-sm-6 col-md-12">
10                 <div class="thumbnail">
11                     <div class="caption">
12                         <h3>{{ $album->nombre }}</h3>
13                         <p>{{ $album->descripcion }}</p>
14                         <p><a href="/validado/fotos?id={{ $album->id }}" class="btn btn-primary" role="button">
15                             Ver Fotos</a></p>
16                     </div>
17                 </div>
18             </div>
19         @endforeach
20 @else
21 <div class="alert alert-danger">
22     <p>Al parecer no tienes álbumes. Crea uno!</p>
23 </div>
24 @endif
25 </div>
26 @endsection
```

Destacar que se usa la función sizeof para verificar que el arreglo que recibimos desde el controlador (\$albumes) sea mayor a 0.

Si es mayor a cero, recorreremos el arreglo y en cada iteración imprimimos el nombre del álbum, la descripción y un botón con dirección a fotos enviándole el ID del álbum.

En caso de que el arreglo esté vacío (lo que indica que el usuario tendría 0 álbumes), se imprimirá un mensaje para indicarle al usuario de que no tiene álbumes.

6. Ingrese al sistema con un usuario que tenga álbumes, escoja la opción de ver los álbumes del usuario y corrobore que la información resultado sea similar a la siguiente:



CREANDO LA FUNCIONALIDAD PARA MOSTRAR FOTOS

1. Cree un nuevo Request de nombre: “MostrarFotosRequest”.
2. En el controlador de las fotos, reference al request recientemente creado.
3. En el controlador de las fotos, referencia a los modelos Album y Foto
4. Modifique el código del request de la siguiente manera:

```
1 <?php namespace GestorImagenes\Http\Requests;
2
3 use GestorImagenes\Http\Requests\Request;
4 use Illuminate\Support\Facades\Auth;
5 use GestorImagenes\Album;
6
7 class MostrarFotosRequest extends Request {
8     /**
9      * Determine if the user is authorized to make this request.
10     *
11     * @return bool
12     */
13     public function authorize()
14     {
15         $user=Auth::user();
16         $id=$this->get('id');
17         $album=$user->albumes()->find($id);
18         if($album){
19             return true;
20         }
21         return false;
22     }
23
24     /**
25     * Get the validation rules that apply to the request.
26     *
27     * @return array
28     */
29     public function rules()
30     {
31         return [
32             'id'=>'required',
33         ];
34     }
35 }
36 }
```

En las reglas solo se indica que necesariamente se tiene que recibir un campo id.

La función authorize dice:

Primero se obtiene el objeto de tipo usuario que se encuentra loggeado.

Luego creamos la variable \$id para guardar el valor del id que ha venido a este request.

Creamos la variable \$album, que será obtenida de los álbumes que posee el usuario pasándole como parámetro el id del álbum que buscamos.

Si esta información es correcta y el álbum existe, se retornará true, en caso contrario, false.

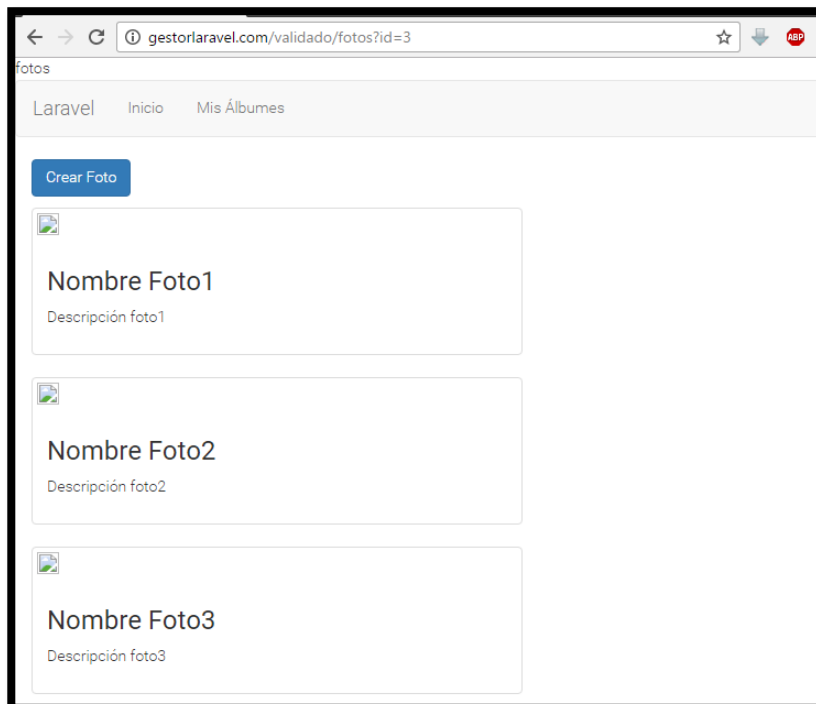
5. Regresando al controlador de las fotos modifique su función getIndex para que quede de la siguiente manera:

```
1 <?php namespace GestorImagenes\Http\Controllers;
2
3 use GestorImagenes\Album;
4 use GestorImagenes\Foto;
5 use GestorImagenes\Http\Requests\MostrarFotosRequest;
6
7 class FotoController extends Controller {
8     /**
9      * Create a new controller instance.
10     *
11     * @return void
12     */
13     public function __construct()
14     {
15         $this->middleware('auth');
16     }
17     public function getIndex(MostrarFotosRequest $request){
18
19         $id=$request->get('id');
20         $album=Album::find($id);
21         $fotos=$album->fotos;
22         return view('fotos.mostrar',['fotos'=>$fotos]);
23     }
24 }
25 }
```


6. Cree la carpeta fotos en las vistas y dentro de ella la vista mostrar.blade.php
7. Puede copiar el código de mostrar los álbumes de un usuario ya que es bastante similar, sin embargo modifique el código para que quede de la siguiente manera:

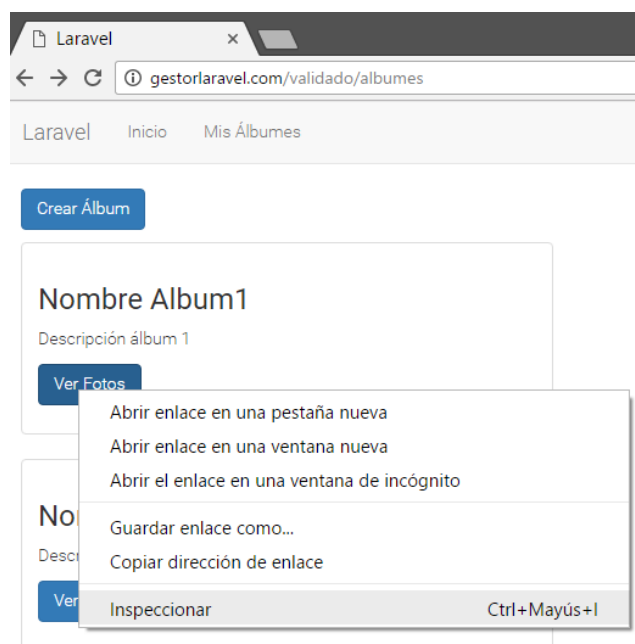
```
1  @extends('app')
2
3  @section('content')
4  <div class="container-fluid">
5  <p><a href="/validado/albumes/crear-foto" class="btn btn-primary" role="button">Crear Foto</a></p>
6  @if(sizeof($fotos)>0)
7      @foreach($fotos as $foto)
8          <div class="row">
9              <div class="col-sm-6 col-md-12">
10                 <div class="thumbnail">
11                     
12                     <div class="caption">
13                         <h3>{{ $foto->nombre }}</h3>
14                         <p>{{ $foto->descripcion }}</p>
15                     </div>
16                 </div>
17             </div>
18         </div>
19     @endforeach
20 @else
21 <div class="alert alert-danger">
22     <p>Al parecer este álbum no tiene Fotos. Crea una!</p>
23 </div>
24 @endif
25 </div>
26 @endsection
27 fotos
```

8. Verifique que un álbum contenga fotos en el sistema.

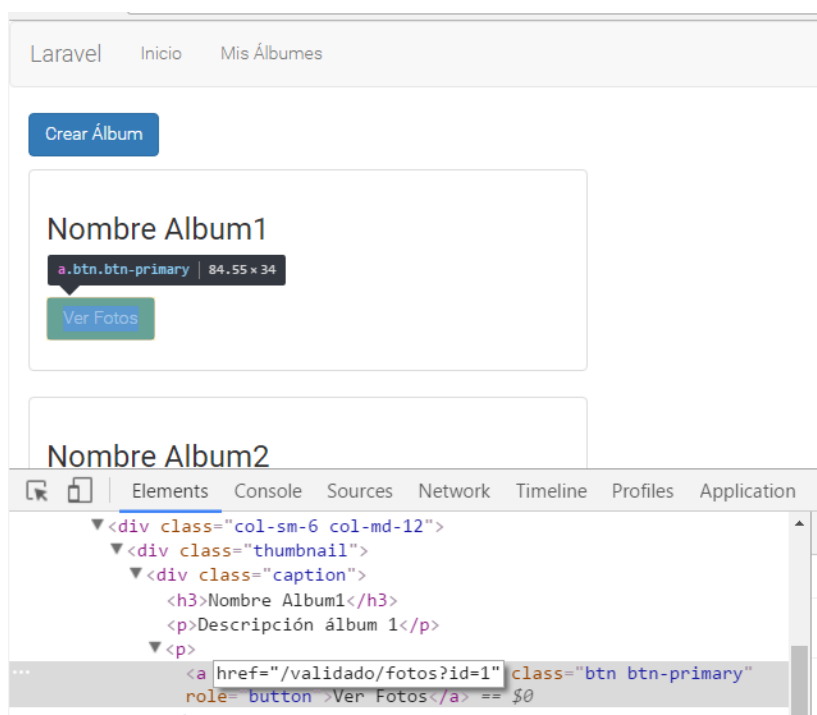


OJO: Las imágenes no se ven porque aún no hemos creado la ruta que tiene cada imagen, aunque es un tema secundario.

1. Tratemos de quebrar la seguridad del sistema, en el link de un álbum para ver sus fotos **click derecho e inspeccionar elemento**.



2. Ahora, modifique el valor del id que se le envía mediante url por un id diferente, por ejemplo 10(en este ejemplo el id será 1 por que pertenece al álbum 1).



3. Ahora, enter y de clic al link del botón del álbum editado, El resultado debería ser como el siguiente.

