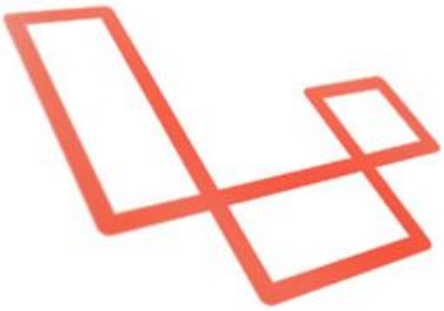
Desarrollo de Aplicaciones Empresariales

LABORATORIO N° 06

Laravel – Requests II y Archivos

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 trabará 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.

INSERTANDO ÁLBUMES EN LA BASE DE DATOS

1. Abra el controlador de los álbumes y ubique la función getCrearÁbum. Modifíquelo de la siguiente manera:

```
24      */
25     public function getCrearAlbum()
26     {
27         return view('albumes.crear-album');
28     }
29     public function postCrearAlbum(){
```

2. Ahora cree la vista de nombre: “crear-album.blade.php”. Y agregue el código para crear un álbum, puede ayudarse copiando el código de actualización de perfil de usuario:

```blade
@extends('app')
@section('content')
<div class="container-fluid">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Crear Álbum</div>
                <div class="panel-body">
                    @if (count($errors) > 0)
                        <div class="alert alert-danger">
                            <strong>Whoops!</strong> Al parecer algo está mal.<br><br>
                            <ul>
                                @foreach ($errors->all() as $error)
                                    <li>{{ $error }}</li>
                                @endforeach
                            </ul>
                        </div>
                    @endif
                    <form class="form-horizontal" role="form" method="POST" action="/validado/albumes/crear-album">
                        <input type="hidden" name="_token" value="{{ csrf_token() }}">
                        <div class="form-group">
                            <label class="col-md-4 control-label">Nombre</label>
                            <div class="col-md-6">
                                <input type="text" class="form-control" name="nombre" value="{{ old('nombre') }}">
                            </div>
                        </div>
                        <div class="form-group">
                            <label class="col-md-4 control-label">Descripcion</label>
                            <div class="col-md-6">
                                <input type="text" class="form-control" name="descripcion" value="{{ old('descripcion') }}">
                            </div>
                        </div>
                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <button type="submit" class="btn btn-primary">
                                    Crear
                                </button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection

```

3. Cree un Request de nombre: “CrearAlbumRequest”.
4. Referencie al request recientemente creado en el controlador de Álbum.
5. Modifique el request de la creación de álbum para que el código quede de la siguiente manera:

```php
<?php namespace GestorImagenes\Http\Requests;

use GestorImagenes\Http\Requests\Request;

class CrearAlbumRequest extends Request {

    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'nombre'=>'required',
            'descripcion'=>'required',
        ];
    }

}

```

6. Primero referencie en el controlador de Album a su Modelo (al modelo del álbum).

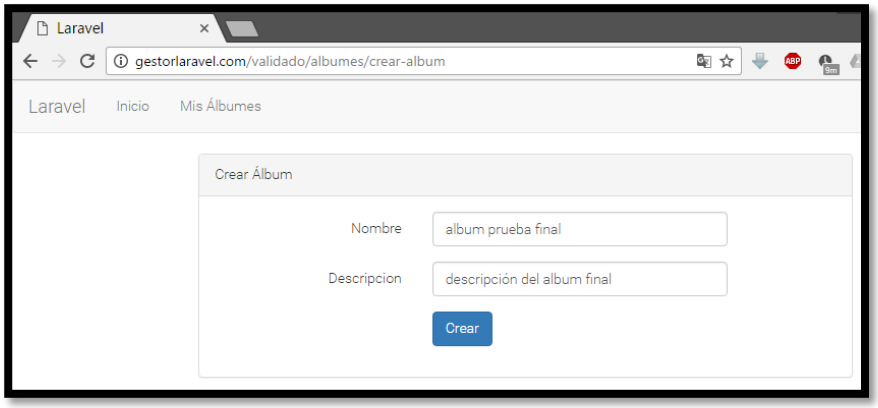
7. En el controlador de Album y nos ubicamos en el action postCrearAlbum(), lo modificamos de la siguiente manera:

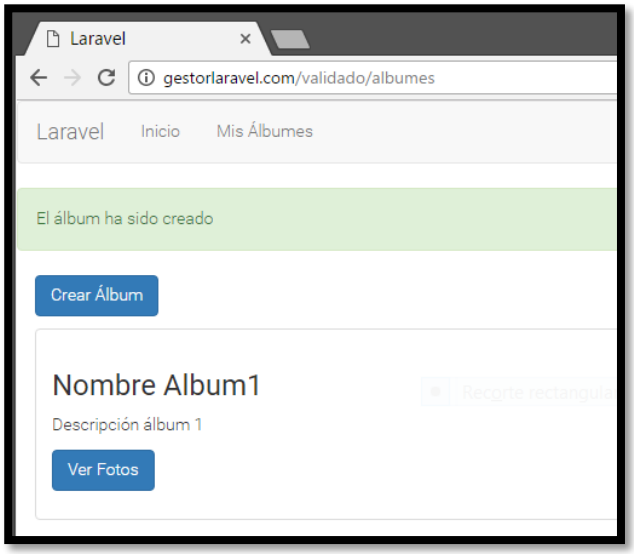
```
33  public function postCrearAlbum(CrearAlbumRequest $request){
34      $usuario=Auth::user();
35      Album::create
36      (
37          [
38              'nombre'=>$request->get('nombre'),
39              'descripcion'=>$request->get('descripcion'),
40              'usuario_id'=>$usuario->id,
41          ]
42      );
43      return redirect('/validado/albumes')->with('creado','El álbum ha sido creado');
44  }
```

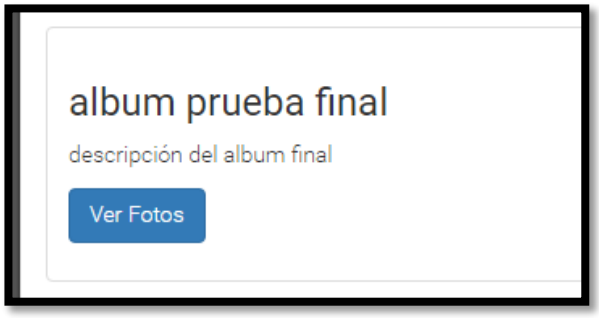
8. Para mejorar la vista luego de la creación de un álbum, inserte el código siguiente en la vista de Mostrar Álbum

```
1   @extends('app')
2
3   @section('content')
4
5   @if(Session::has('creado'))
6       <div class="alert alert-success">
7           <p>El álbum ha sido creado</p>
8       </div>
9   @endif
10
11  <div class="container-fluid">
12  <p><a href="/validado/albumes/crear-album" class="btn btn-primary" role=b
13  @if(sizeof($albumes)>0)
14      @foreach($albumes as $album)
15      <div class="row">
16          <div class="col-sm-6 col-md-12">
17              <div class="thumbnail">
18                  <div class="caption">
19                      <h3>{{$album->nombre}}</h3>
20                      <p>{{$album->descripcion}}</p>
```

9. Intente crear un nuevo álbum y verifique que se haya hecho lo correcto







INSERTANDO FOTOS EN LA BASE DE DATOS

Para crear una foto, ésta debe pertenecer a un álbum, de modo que incluso el método getCrearFoto(), debe recibir el id del álbum al que va a pertenecer. Este id requerido debería ser enviado por la vista que tiene la lista de fotos que un álbum contiene, es decir de la vista mostrar.blade.php (de la carpeta fotos - view).
Y para crear una nueva foto ésta vista mostrar.blade.php (de la carpeta fotos - view), debe enviar el id que recibió al método postCrearFoto

1. Abra el controlador de Foto. Modifique la función getCrearFoto()

```
30      */
31      public function getCrearFoto($id)
32      {
33          return view('fotos.crear-foto');
34      }
```

Y recibimos el id del álbum al que la nueva foto pertenecerá. Ahora modificaremos la función getIndex del controlador de fotos para que envíe la variable ID del álbum.

2. Modifique la función getIndex() del controlador de fotos para que quede de la siguiente manera:

```
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'=>$fotos,'id'=>$id]);
23
24      }
25
26      /**
27       * Show the application dashboard to the user.
28       *
```

enviamos el id del album desde la vista de mostrar fotos de un album.

Ahora en la vista de Mostrar todas las fotos de un álbum, en el link de Crear nueva Foto, enviaremos el $id del álbum al que ésta foto va a pertenecer.

3. Abra la vista **mostrar.blade.php** de fotos y modifique el código de la siguiente manera:

```
1   @extends('app')
2
3   @section('content')
4   <div class="container-fluid">
5   <p><a href="/validado/fotos/crear-foto?id={{$id}}" class="btn btn-pr
6   @if(sizeof($fotos)>0)
7       @foreach($fotos as $foto)
8       <div class="row">
9           <div class="col-sm-6 col-md-12">
10              <div class="thumbnail">
11              <img src="{{$foto->ruta}}">
12                  <div class="caption">
13                      <h3>{{$foto->nombre}}</h3>
14                      <p>{{$foto->descripcion}}</p>
15                  </div>
```

De esta manera, cuando creemos una nueva foto el id del álbum estará siendo enviado desde esta vista =).

4. Cree un Resquest nuevo de nombre: “CrearFotoRequest”.
5. Modifique el Request creado de la siguiente manera:

```php
<?php namespace GestorImagenes\Http\Requests;

use GestorImagenes\Http\Requests\Request;
use GestorImagenes\Album;
use Illuminate\Support\Facades\Auth;

class CrearFotoRequest extends Request {

    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        $user=Auth::user();
        $id=$this->get('id');
        $album=$user->albumes()->find($id);
        if($album){
            return true;
        }
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'id'=>'required|exists:albumes,id',
            'nombre'=>'required',
            'descripcion'=>'required',
            'imagen'=>'required|image|max:20000'
        ];
    }

}

```

La función authorize será la misma que utilizamos en MostrarFotoRequest, ya que lo que queremos comprobar es que ese id de álbum le pertenece al usuario loggeado actualmente.
En las reglas destaca el campo imagen que es de tipo ***image*** y debe pesar como máximo 20Mb (aunque es mucho espacio para una fotografía.). Además del campo id que es requerido y tiene que existir en la tabla álbumes en su columna id.

Ahora vamos a crear la vista para crear la foto.

6. Cree la vista de nombre ***crear-foto.blade.php*** en la carpeta de fotos en vistas.
7. Puede apoyarse en el código de mostrar fotos para modificarlo y que finalmente quede de la siguiente manera:

```
@extends('app')

@section('content')
<div class="container-fluid">
    <form class="form-horizontal" role="form" method="POST" action="/validado/fotos/crear-foto?id={{id}}" enctype="multipart/form-data">
        <input type="hidden" name="_token" value="{{ csrf_token() }}" required>
        <div class="form-group required required">
            <label class="col-md-4 control-label">Nombre</label>
            <div class="col-md-6">
                <input type="text" class="form-control" name="nombre" value="{{old('nombre')}}" required>
            </div>
        </div>

        <div class="form-group required">
            <label class="col-md-4 control-label">Descripción</label>
            <div class="col-md-6">
                <textarea type="text" class="form-control" name="descripcion" rows="3" required> {{old('descripcion')}} </textarea>
            </div>
        </div>

        <div class="form-group required">
            <label class="col-md-4 control-label">Imagen max: 20Mb</label>
            <div class="col-md-6">
                <input type="file" class="form-control" name="imagen" required>
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-6 col-md-offset-4">
                <button type="submit" class="btn btn-primary">
                    Subir y crear foto
                </button>
            </div>
        </div>
    </form>
</div>
@endsection

```

8. Nuevamente modifique la función getCrearFoto del controlador de Fotos para que ahora no solo se retorne la vista si no también el id del álbum al que pertenecerá la nueva foto.

```
     */
    public function getCrearFoto(Request $request)
    {
        $id=$request->get('id');
        return view('fotos.crear-foto',['id'=>$id]);
    }
```

En este caso usamos un Request genérico (de la clase Request), porque de éste solo necesitamos el id que recibiremos.

No olvidarse de referenciar a la librería Request en la cabecera del documento:

```
use GestorImagenes\Http\Requests\MostrarFotosRequest;
use GestorImagenes\Http\Requests\CrearFotoRequest;
use Illuminate\Http\Request;
use Carbon\Carbon;
```

9. Ahora modifique el controlador de fotos en su action postCrearFoto para que quede de la siguiente manera:

```
38      public function postCrearFoto(CrearFotoRequest $request){
39          $id=$request->get('id');
40          $imagen=$request->file('imagen');
41          $ruta='/img/';
42          $nombre=sha1(Carbon::now()).".".$imagen->guessExtension();
43          $imagen->move(getcwd().$ruta,$nombre);
44          Foto::create(
45              [
46                  'nombre'=>$request->get('nombre'),
47                  'descripcion'=>$request->get('descripcion'),
48                  'ruta'=>$ruta.$nombre,
49                  'album_id'=>$id,
50              ]
51          );
52          return redirect("/validado/fotos?id=$id")->with('creada','La foto ha sido subida');
53      }
```

Explicación:
Línea 39: Recuperamos el id del álbum al que pertenecerá la foto.
Línea 40: Recuperamos del Request tmb, solo que éste es un archivo (es la forma de recuperar arhivos).
Línea 41: La ruta se compondrá de:

```
38      public function postCrearFoto(CrearFotoRequest $request){
39          $id=$request->get('id');
40          $imagen=$request->file('imagen');
41          $ruta='/img/';
42          $nombre=sha1(Carbon::now()).".".$imagen->guessExtension();
43          $imagen->move(getcwd().$ruta,$nombre);
44          Foto::create(
45              [
46                  'nombre'=>$request->get('nombre'),
47                  'descripcion'=>$request->get('descripcion'),
48                  'ruta'=>$ruta.$nombre,
49                  'album_id'=>$id,
50              ]
51          );
52          return redirect("/validado/fotos?id=$id")->with('creada','La foto ha sido subida');
53      }
```

Librería y función que nos da el momento actual en el que se realiza la operación.
extensión del archivo origen.
con el nombre que se copiará el archivo al servidor
obtiene el directorio de origen del archivo

Para hacer uso de la librería Carbon, no olvidarse de referenciarla en la cabecera del documento. Ésta librería sirve para obtener fechas y horas actuales y es nativa de Laravel.

```
6   use GestorImagenes\
7   use Carbon\Carbon;
8
```

¿Por qué cifrar?, solo para que si dos personas ponen una foto con el mismo nombre en el mismo instante el nombre sería el mismo, en este caso cuando aplicamos el cifrado sha1, el valor resultado nunca se repetirá.

Línea 42: La función getcwd(), obtiene el directorio actual de donde se está trabajando. En modo de producción estaríamos hablando de la carpeta public de nuestro proyecto, acompañado de los nombres de carpetas de nuestro servidor.

Línea 43: Y la función move, sirve para copiar el archivo con el que estamos trabajando a una ruta en nuestro servidor.

Líneas 44 – 49: Creamos un registro en la tabla Foto con los datos mencionados.

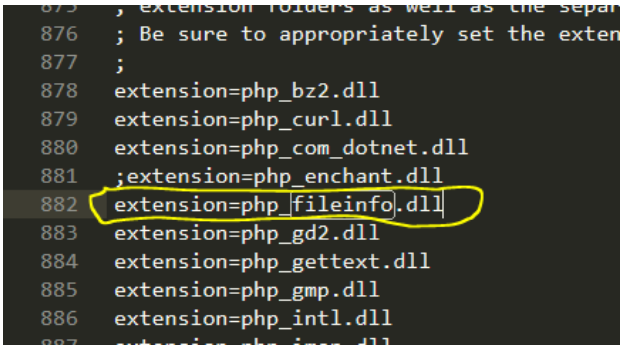
Línea 52: retornamos nuevamente a la página de fotos, no olvidándonos de enviar el id del álbum por si queremos crear otra foto. Además de una variable de sesión llamada ***creada*** con información de éxito al momento de guardar los cambios realizados.

ATENCIÓN: ESTE ES UN PASO OPCIONAL.

Es posible que en sus servidores XAMP o WAMP, no tengan habilitado el permiso en PHP, de dar el acceso u obtener la información de los archivos que se manejen en las aplicaciones, ejm: la ruta de un archivo, la extensión, etc.
Y es probable también que, por este motivo, obtengan un error del siguiente tipo:

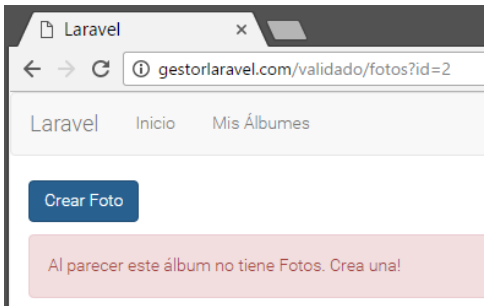
SOLUCIÓN:

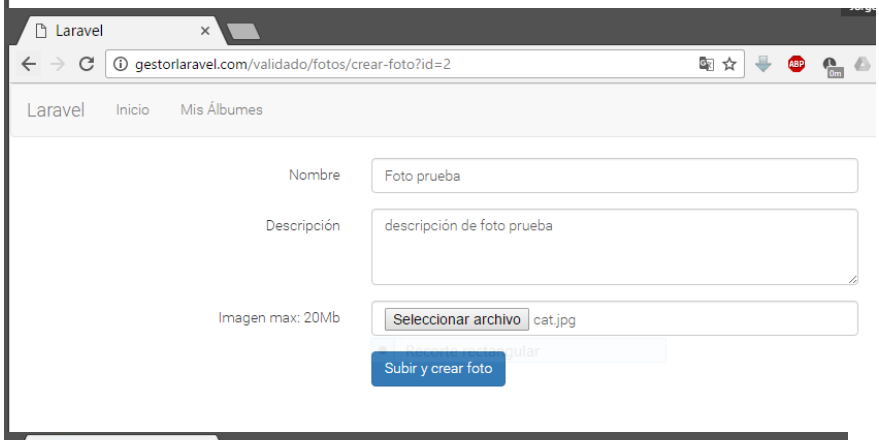
1. Abrir el archivo PHP.INI de nuestros servidores
2. Procedan a buscar el texto “fileinfo” y es una extensión que debería estar SIN COMENTARIO para que se pueda hacer uso de ella.

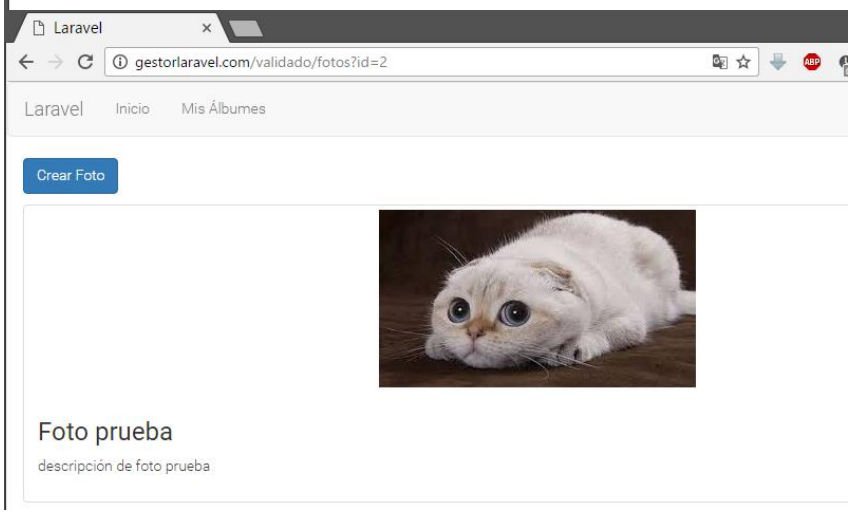


3. Una vez quitado el comentario en la línea, guardan los cambios y proceden a reiniciar el servidor =).

10. Para finalizar la sección, solo queda probar subir fotos a nuestros álbumes.


Laravel
gestorlaravel.com/validado/fotos?id=2
Laravel
Inicio
Mis Álbumes
Crear Foto
Al parecer este álbum no tiene Fotos. Crea una!


Laravel
gestorlaravel.com/validado/fotos/crear-foto?id=2
Laravel
Inicio
Mis Álbumes
Nombre
Foto prueba
Descripción
descripción de foto prueba
Imagen max: 20Mb
Seleccionar archivo
cat.jpg
Subir y crear foto


Laravel
gestorlaravel.com/validado/fotos?id=2
Laravel
Inicio
Mis Álbumes
Crear Foto
Foto prueba
descripción de foto prueba

WIN!