1 LARAVEL Framework - Tutorial 04 - Creación de Aplicación Álbum Fotográfico en Laravel

1.1 Sumario

2 Configuración inicial de la aplicación Enfocalia (álbum fotográfico)

- **ATENCION: Damos por supuesto que ya tenemos instalado un servidor web, php, mysql, compose, usuario MySQL y Base de Datos.**

2.1 Instalación de Laravel

```
# Crear una carpeta para la aplicación.
mkdir album
cd album

# Instalamos Laravel
composer create-project laravel/laravel . dev-master
```

2.2 Configuración del entorno y proyecto de Laravel

- Vamos a configurar el espacio de nombres para esta aplicación que le llamaremos **Enfocalia**

```
php artisan app:name Enfocalia
#Application namespace set!
```

- Editamos el fichero de configuración **.env** con **DEBUG a true** y los **datos de conexión de base de datos**:

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=WIlqbeKBoomeAOnF5ivw8ysm6oynoFJU

DB_HOST=localhost
DB_DATABASE=c2base2
DB_USERNAME=c2base2
DB_PASSWORD=xxxxxxxxx

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
```

3 Creación de los modelos

- **Estructura de las tablas y relaciones entre ellas**:

```
Album
#id
nombre
descripcion
usuario_id

"Un Album pertenece a un Usuario."
"Un Album tiene muchas fotos."

Foto
#id
```

```
nombre
descripcion
ruta
album_id

"Una Foto pertenece a un Album."

Usuario
#id
nombre
email
password
pregunta
respuesta

"Un usuario poseerá muchos Álbumes."
```

- **Crearemos a continuación los modelos y sus relaciones.**

3.1 Modelo Usuario

- Aprovechamos el modelo **app/User.php** y lo renombramos a **app/Usuario.php** y cambiamos también el nombre de la clase a **Usuario**.
- Contenido de **app/Usuario.php**:

```
<?php namespace Enfocalia;

use Illuminate\Auth\Authenticatable;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Auth\Passwords\CanResetPassword;
use Illuminate\Contracts\Auth\Authenticatable as AuthenticatableContract;
use Illuminate\Contracts\Auth\CanResetPassword as CanResetPasswordContract;

class Usuario extends Model implements AuthenticatableContract, CanResetPasswordContract {

use Authenticatable, CanResetPassword;

/**
         * The database table used by the model.
         *
         * @var string
         */
protected $table = 'usuarios';

/**
         * The attributes that are mass assignable.
         *
         * @var array
         */
protected $fillable = ['id','nombre', 'email', 'password','pregunta','respuesta'];

/**
         * The attributes excluded from the model's JSON form.
         *
         * @var array
         */
protected $hidden = ['password', 'remember_token'];


// Definimos las relaciones utilizando funciones.
// "Un usuario posee uno o varios albumes"
public function albumes()
{
return $this->hasMany('Enfocalia\Album');
}
}
```

3.2 Modelo Album

- Creamos el modelo con php artisan:

```
php artisan make:model Album

# Se ha creado el modelo y la migración
Model created successfully.
Created Migration: 2015_04_28_175836_create_albums_table
```

- Contenido de **app/Album.php**:

```
<?php namespace Enfocalia;

use Illuminate\Database\Eloquent\Model;

class Album extends Model {

protected $table="albumes";
protected $fillable=['id','nombre','descripcion','usuario_id'];

// Definimos las relaciones utilizando funciones.
// "Un álbum posee muchas fotos"
public function fotos()
{
return $this->hasMany('Enfocalia\Foto');
}

// "Un álbum pertenece a un propietario"
public function propietario()
{
return $this->belongsTo('Enfocalia\Usuario');
}
}
```

3.3 Modelo Foto

- Creamos el modelo con php artisan:

```
php artisan make:model Foto

# Se ha creado el modelo y la migración
Model created successfully.
Created Migration: 2015_04_28_180148_create_fotos_table
```

- Contenido de **app/Foto.php**:

```
<?php namespace Enfocalia;

use Illuminate\Database\Eloquent\Model;

class Foto extends Model {

protected $table="fotos";
protected $fillable=['id','nombre','descripcion','ruta','album_id'];

// Definimos las relaciones utilizando funciones.
// "Una foto pertenece a un album"
public function album()
{
return $this->belongsTo('Enfocalia\Album');
}
}
```

4 Creación inicial de los Controladores

- Vamos a crear los controladores (se encargan de interactuar entre los modelos y las vistas).
- Además los controladores hacen un mapeo de las rutas
- Accedemos a la carpeta **app/Http/Controllers** y vamos a **renombrar** la carpeta **Auth** a **Validacion**.
- Accedemos a la carpeta **app/Http/Controllers/Validacion** y vamos a **renombrar** AuthController **por** ValidacionController**.**
- Accedemos a la carpeta **app/Http/Controllers/Validacion** y vamos a **eliminar PasswordController.php** ya que nosotros para recuperar la contraseña lo haremos en base a una **pregunta** y una **respuesta** y no enviando por correo electrónico.

4.1 ValidacionController.php

- Editamos el fichero **ValidacionController** para programar nuestros propios métodos de validación y registro de usuarios.
- Contenido de **app/Http/Controllers/Validacion/ValidacionController.php**:

```
<?php namespace Enfocalia\Http\Controllers\Validacion;

use Enfocalia\Http\Controllers\Controller;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

class ValidacionController extends Controller {
protected $auth;
protected $registrar;

public function __construct(Guard $auth, Registrar $registrar)
{
$this->auth = $auth;
$this->registrar = $registrar;

// Sustituir getLogout por getSalida
$this->middleware('guest', ['except' => 'getSalida']);
}

public function getRegistro()
{
return "formulario de creación de cuentas.";
}

/**
         * Handle a registration request for the application.
         *
         * @param  \Illuminate\Http\Request  $request
         * @return \Illuminate\Http\Response
         */
public function postRegistro(Request $request)
{
$validator = $this->registrar->validator($request->all());

if ($validator->fails())
{
$this->throwValidationException(
$request, $validator
);
}

$this->auth->login($this->registrar->create($request->all()));

return redirect($this->redirectPath());
}

public function getInicio()
{
return 'Mostrando formulario Inicio Sesión';
}

/**
         * Handle a login request to the application.
         *
         * @param  \Illuminate\Http\Request  $request
         * @return \Illuminate\Http\Response
         */
public function postInicio(Request $request)
{
$this->validate($request, [
'email' => 'required|email', 'password' => 'required',
]);

$credentials = $request->only('email', 'password');

if ($this->auth->attempt($credentials, $request->has('remember')))
```

```
{
return redirect()->intended($this->redirectPath());
}

return redirect($this->loginPath())
->withInput($request->only('email', 'remember'))
->withErrors([
'email' => $this->getFailedLoginMessage(),
]);
}

protected function getFailedLoginMessage()
{
return 'E-mail o contraseña incorrectos.';
}

public function getSalida()
{
$this->auth->logout();

return redirect(property_exists($this, 'redirectAfterLogout') ? $this->redirectAfterLogout : '/');
}

/**
         * Get the post register / login redirect path.
         *
         * @return string
         */
public function redirectPath()
{
if (property_exists($this, 'redirectPath'))
{
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->redirectTo : '/inicio';
}

public function loginPath()
{
return property_exists($this, 'loginPath') ? $this->loginPath : '/validacion/inicio';
}

public function getRecuperar()
{
return "recuperar contraseña";
}

public function postRecuperar()
{
return "recuperando contraseña";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

4.2 InicioController.php

- Renombramos el fichero **HomeController.php** a **InicioController.php**.
- InicioController se encarga de mostrar la información para los usuarios que se acaban de autenticar.

- Contenido de **app**/**Http**/**Controllers**/**InicioController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class InicioController extends Controller {
```

```
/*
        |--------------------------------------------------------------------------
        | Home Controller
        |--------------------------------------------------------------------------
        |
        | This controller renders your application's "dashboard" for users that
        | are authenticated. Of course, you are free to change or remove the
        | controller as you wish. It is just here to get your app started!
        |
        */

/**
         * Create a new controller instance.
         *
         * @return void
         */
public function __construct()
{
$this->middleware('auth');
}

/**
         * Show the application dashboard to the user.
         *
         * @return Response
         */
public function getIndex()
{
return "pagina de inicio de usuario validado";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

4.3 BienvenidaController.php

- Renombramos el fichero **WelcomeController** a **BienvenidaController**.
- Este controlador se encarga de mostrar la información para los usuarios que no han iniciado sesión.
- Contenido de **app**/**Http**/**Controllers**/**BienvenidaController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class BienvenidaController extends Controller {

/**
         * Create a new controller instance.
         *
         * @return void
         */
public function __construct()
{
$this->middleware('guest');
}

/**
         * Show the application welcome screen to the user.
         *
         * @return Response
         */
public function getIndex()
{
return "Página de bienvenida a la aplicación.";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
```

```
}
}
```

4.4 FotoController.php

- Creamos un nuevo fichero **FotoController.php**
- Hacemos una copia del fichero HomeController.php y pegamos el contenido en FotoController.php y lo editamos.
- Contenido de **app**/**Http**/**Controllers**/**FotoController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class FotoController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
return "Mostrando las fotos del usuario.";
}

public function getCrearFoto()
{
return "Formulario de creación fotos.";
}

public function postCrearFoto()
{
return "Almacenando fotos...";
}

public function getActualizarFoto()
{
return "Formulario de actualización de fotos.";
}

public function postActualizarFoto()
{
return "Actualizando foto...";
}

public function getEliminarFoto()
{
return "Formulario de eliminación de fotos.";
}

public function postEliminarFoto()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

4.5 AlbumController.php

- Creamos un nuevo fichero **AlbumController.php**
- Hacemos una copia del contenido de **FotoController.php** y pegamos el contenido en **AlbumController.php** y lo editamos.
- Contenido de **app**/**Http**/**Controllers**/**AlbumController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class AlbumController extends Controller {
```

```
public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
return "Mostrando álbumes del usuario.";
}

public function getCrearAlbum()
{
return "Formulario de creación Albumes.";
}

public function postCrearAlbum()
{
return "Almacenando Albumes...";
}

public function getActualizarAlbum()
{
return "Formulario de actualización de Albumes.";
}

public function postActualizarAlbum()
{
return "Actualizando Album...";
}

public function getEliminarAlbum()
{
return "Formulario de eliminación de Albumes.";
}

public function postEliminarAlbum()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

4.6 UsuarioController.php

- Creamos un nuevo fichero **UsuarioController.php**
- Hacemos una copia del contenido de **FotoController.php** y pegamos el contenido en **UsuarioController.php** y lo editamos.
- Contenido de **app**/**Http**/**Controllers**/**UsuarioController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class UsuarioController extends Controller {

// Para el tema de contraseñas se encargará el controlador de Validación

public function __construct()
{
$this->middleware('auth');
}

public function getEditarPerfil()
{
return "Mostrando formulario de perfil.";
}

public function postEditarPerfil()
{
```

```
return "Generando actualización de perfil...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

5 Rutas y Middleware

5.1 Modificación de routes

- Configuramos las rutas para que llamen a los controladores que hemos definido anteriormente.
- Para ello modificaremos el fichero app/Htpp/Routes.php.

- Contenido del fichero **app**/**Htpp**/**Routes.php**:

```
<?php

/*
|--------------------------------------------------------------------------
| Application Routes
|--------------------------------------------------------------------------
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

// Definimos las llamadas a todos los controladores que hemos creado anteriormente.
// Es importante el orden de más restrictivo a más genérico.

Route::controllers([
'validacion' => 'Validacion\ValidacionController',
'validado/fotos' => 'FotoController',
'validado/albumes' => 'AlbumController',
'validado/usuario' => 'UsuarioController',
'validado' => 'InicioController',
'/'=> 'BienvenidaController'
]);
```

5.2 Modificación de Middleware Authenticate.php

- Si intentamos entrar en /**validado** o en /**validado**/**fotos** vemos que nos redirecciona a **auth**/**login** por lo que tenemos que corregir ese fallo.
- Tenemos que modificar la redirección a **auth**/**login** a **validacion**/**inicio** en la **línea 4**3.

- Contenido del fichero **app**/**Htpp**/**Middleware**/**Authenticate.php**:

```
<?php namespace Enfocalia\Http\Middleware;

use Closure;
use Illuminate\Contracts\Auth\Guard;

class Authenticate {

/**
         * The Guard implementation.
         *
         * @var Guard
         */
protected $auth;

/**
         * Create a new filter instance.
         *
```

```
         * @param  Guard  $auth
         * @return void
         */
public function __construct(Guard $auth)
{
$this->auth = $auth;
}

/**
         * Handle an incoming request.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Closure  $next
         * @return mixed
         */
public function handle($request, Closure $next)
{
if ($this->auth->guest())
{
if ($request->ajax())
{
return response('Unauthorized.', 401);
}
else
{
return redirect()->guest('validacion/inicio');
}
}

return $next($request);
}

}
```

- Una vez que hemos probado que funciona comentaremos desde las líneas 35 a la 45, para desactivar esta comprobación y así poder programar la sección de usuarios registrados cómodamente.

- Contenido del fichero **app/Htpp/Middleware/Authenticate.php**:

```
<?php namespace Enfocalia\Http\Middleware;

use Closure;
use Illuminate\Contracts\Auth\Guard;

class Authenticate {

/**
         * The Guard implementation.
         *
         * @var Guard
         */
protected $auth;

/**
         * Create a new filter instance.
         *
         * @param  Guard  $auth
         * @return void
         */
public function __construct(Guard $auth)
{
$this->auth = $auth;
}

/**
         * Handle an incoming request.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Closure  $next
         * @return mixed
         */
public function handle($request, Closure $next)
```

```
{
/*              if ($this->auth->guest())
                {
                        if ($request->ajax())
                        {
                                return response('Unauthorized.', 401);
                        }
                        else
                        {
                                return redirect()->guest('validacion/inicio');
                        }
                }
*/
return $next($request);
}

}
```

- Modificaremos la **línea 38** del fichero **RedirectIfAuthenticated.php** para que apunte a /validado cuando un usuario está autenticado.
- Contenido del fichero **app/Htpp/Middleware/RedirectIfAuthenticated.php**:

```
<?php namespace Enfocalia\Http\Middleware;

use Closure;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Http\RedirectResponse;

class RedirectIfAuthenticated {

/**
         * The Guard implementation.
         *
         * @var Guard
         */
protected $auth;

/**
         * Create a new filter instance.
         *
         * @param  Guard  $auth
         * @return void
         */
public function __construct(Guard $auth)
{
$this->auth = $auth;
}

/**
         * Handle an incoming request.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Closure  $next
         * @return mixed
         */
public function handle($request, Closure $next)
{
if ($this->auth->check())
{
return new RedirectResponse(url('/validado'));
}

return $next($request);
}

}
```

6 Creación de algunas vistas

- Por defecto vienen 3 vistas: **app**, **home** y **welcome**.

6.1 Vista app.blade.php

- Esta vista es como el esqueleto de nuestra aplicación (lo que hemos visto en otros tutoriales como layouts/master.blade.php.
- Editaremos la vista **resources/views/app.blade.php** para adaptar las rutas a nuestra aplicación:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laravel</title>

<link href="{{ asset('/css/app.css') }}" rel="stylesheet">


<link href='//fonts.googleapis.com/css?family=Roboto:400,300' rel='stylesheet' type='text/css'>




</head>
<body>
        <nav class="navbar navbar-default">
                <div class="container-fluid">
                        <div class="navbar-header">
                                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-exampl
                                        <span class="sr-only">Toggle Navigation</span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                </button>
                                <a class="navbar-brand" href="#">Enfocalia</a>
                        </div>

                        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                                <ul class="nav navbar-nav">
                    /') }}">Inicio</a></li><li><a href="{{ url('
                                </ul>

                                <ul class="nav navbar-nav navbar-right">
                                        @if (Auth::guest())
                    /validacion/inicio') }}">Iniciar Sesión</a></li>
                    /validacion/registro') }}">Registrarse</a></li>
                                        @else
                                                <li class="dropdown">
                                                        <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" ari
                                                        <ul class="dropdown-menu" role="menu">
                    /validacion/salida') }}">Salir</a></li>    <li><a href="{{ url('
                                                        </ul>
                                                </li>
                                        @endif
                                </ul>
                        </div>
                </div>
        </nav>

        @yield('content')


<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.1/js/bootstrap.min.js"></script>
</body>
</html>
```

6.2 Vista home.blade.php

- Vamos a renombrar la vista a **inicio.blade.php**.
- Editaremos la vista **resources/views/inicio.blade.php**:

```
@extends('app')

@section('content')
<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Bienvenido (usuario)
</div>
</div>
</div>
</div>
</div>
@endsection
```

6.3 Vista welcome.blade.php

- Vamos a renombrar la vista a **bienvenida.blade.php**.
- **Copiaremos** el código de la vista **inicio.blade.ph**p.
- **Pegamos** el código anterior en la vista **bienvenida.blade.php**.
- **Editaremos** la vista **resources/views/bienvenida.blade.php**:

```
@extends('app')

@section('content')
<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Por favor inicie sesión para acceder al sistema.
</div>
</div>
</div>
</div>
</div>
@endsection
```

6.4 Adaptar los controladores a las nuevas Vistas

6.4.1 BienvenidaController

- Contenido del fichero **app/Http/Controllers/BienvenidaController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class BienvenidaController extends Controller {

/**
         * Create a new controller instance.
         *
         * @return void
         */
public function __construct()
{
$this->middleware('guest');
}

/**
         * Show the application welcome screen to the user.
         *
         * @return Response
         */
public function getIndex()
```

```
{
//return "Página de bienvenida a la aplicación.";
return view('bienvenida');
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

6.4.2 InicioController

- Contenido del fichero **app/Http/Controllers/InicioController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

class InicioController extends Controller {

/*
        |--------------------------------------------------------------------------
        | Home Controller
        |--------------------------------------------------------------------------
        |
        | This controller renders your application's "dashboard" for users that
        | are authenticated. Of course, you are free to change or remove the
        | controller as you wish. It is just here to get your app started!
        |
        */

/**
         * Create a new controller instance.
         *
         * @return void
         */
public function __construct()
{
$this->middleware('auth');
}

/**
         * Show the application dashboard to the user.
         *
         * @return Response
         */
public function getIndex()
{
//return "pagina de inicio de usuario validado";
return view('inicio');
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Ya podremos probar si funcionan las URL accediendo a http://www.nuestrodominio.local

6.4.3 ValidacionController

- Necesitamos corregir ahora los mensajes que aparecen cuando pulsamos en las opciones del menú: Iniciar Sesión y Registrarse.

- Contenido del fichero **app/Http/Controllers/Validacion/ValidacionController.php**:

```
<?php namespace Enfocalia\Http\Controllers\Validacion;

use Enfocalia\Http\Controllers\Controller;
```

```
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

class ValidacionController extends Controller {
protected $auth;
protected $registrar;

public function __construct(Guard $auth, Registrar $registrar)
{
$this->auth = $auth;
$this->registrar = $registrar;

// Sustituir getLogout por getSalida
$this->middleware('guest', ['except' => 'getSalida']);
}

public function getRegistro()
{
//return "formulario de creación de cuentas.";
return view('validacion.registro');
}

/**
         * Handle a registration request for the application.
         *
         * @param  \Illuminate\Http\Request  $request
         * @return \Illuminate\Http\Response
         */
public function postRegistro(Request $request)
{
$validator = $this->registrar->validator($request->all());

if ($validator->fails())
{
$this->throwValidationException(
$request, $validator
);
}

$this->auth->login($this->registrar->create($request->all()));

return redirect($this->redirectPath());
}

public function getInicio()
{
//return 'Mostrando formulario Inicio Sesión';
return view('validacion.inicio');
}

/**
         * Handle a login request to the application.
         *
         * @param  \Illuminate\Http\Request  $request
         * @return \Illuminate\Http\Response
         */
public function postInicio(Request $request)
{
$this->validate($request, [
'email' => 'required|email', 'password' => 'required',
]);

$credentials = $request->only('email', 'password');

if ($this->auth->attempt($credentials, $request->has('remember')))
{
return redirect()->intended($this->redirectPath());
}

return redirect($this->loginPath())
->withInput($request->only('email', 'remember'))
->withErrors([
```

```
'email' => $this->getFailedLoginMessage(),
]);
}

protected function getFailedLoginMessage()
{
return 'E-mail o contraseña incorrectos.';
}

public function getSalida()
{
$this->auth->logout();

return redirect(property_exists($this, 'redirectAfterLogout') ? $this->redirectAfterLogout : '/');
}

/**
         * Get the post register / login redirect path.
         *
         * @return string
         */
public function redirectPath()
{
if (property_exists($this, 'redirectPath'))
{
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->redirectTo : '/inicio';
}

public function loginPath()
{
return property_exists($this, 'loginPath') ? $this->loginPath : '/validacion/inicio';
}

public function getRecuperar()
{
return "recuperar contraseña";
}

public function postRecuperar()
{
return "recuperando contraseña";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

6.5 Vista inicio.blade.php

- **Renombramos la carpeta resources/views/auth a resources/views/validacion**.
- Entramos en la carpeta **resources/views/validacion** y **renombramos** el fichero **login.blade.php** a **inicio.blade.php**.
- Contenido del fichero **resources/views/validacion/inicio.blade.php**:

```
@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">Iniciar Sesión</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
```

```
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validacion/inicio') }}">
<input type="hidden" name="_token" value="{{ csrf_token() }}">

<div class="form-group">
<label class="col-md-4 control-label">Correo Electrónico</label>
<div class="col-md-6">
<input type="email" class="form-control" name="email" value="{{ old('email') }}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Password</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<div class="checkbox">
<label>
<input type="checkbox" name="remember"> Recordarme
</label>
</div>
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">Iniciar Sesión</button>

<a class="btn btn-link" href="{{ url('/validacion/recuperar') }}">¿Olvidó su contraseña?</a>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

6.6 Vista registro.blade.php

- Entramos en la carpeta **resources/views/validacion** y **renombramos** el fichero **register.blade.php** a **registro.blade.php**.
- Contenido del fichero **resources/views/validacion/inicio.blade.php**:

```
@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">Registrarse</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
```

```
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validacion/registro') }}">
<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">Correo Electrónico</label>
<div class="col-md-6">
<input type="email" class="form-control" name="email" value="{{ old('email') }}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Confirmar Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password_confirmation">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Pregunta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="pregunta">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Respuesta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="respuesta">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">
Registrarse
</button>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

- **Eliminamos** los ficheros **resources/views/validacion/password.blade.php** y **resources/views/validacion/reset.blade.php** ya que no necesitaremos esas vistas.

6.7 Vista recuperar.blade.php

- Entramos en la carpeta **resources/views/validacion** y **creamos** el fichero **recuperar.blade.php**.
- Copiamos el contenido del fichero de registro.blade.php y lo pegamos en el nuevo fichero.

- Contenido del fichero **resources/views/validacion/recuperar.blade.php**:

```
@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">Recuperar contraseña</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validacion/recuperar') }}">
<input type="hidden" name="_token" value="{{ csrf_token() }}">

<div class="form-group">
<label class="col-md-4 control-label">Correo Electrónico</label>
<div class="col-md-6">
<input type="email" class="form-control" name="email" value="{{ old('email') }}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Nueva Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Confirmar Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password_confirmation">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Pregunta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="pregunta">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Respuesta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="respuesta">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">
Recuperar Contraseña
</button>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Editamos el controlador **app/Http/Controllers/Validacion/ValidacionController.php** para que use la nueva vista **validacion.recuperar**:

```
.....

public function getRecuperar()
{
//return "recuperar contraseña";
return view('validacion.recuperar');
}
.....
```

7 Creación de migraciones para los modelos

- Cuando **creamos los modelos con PHP Artisan automáticamente se crearon automáticamente las plantillas de migraciones para cada modelo**.
- En el caso de que no estuvieran creadas se podrían hacer con PHP Artisan.
- Creación de las migraciones con PHP Artisan (si fuera necesario)

```
# Es muy importante el orden de creación de las migraciones.
# Primero tendrían que estar los álbumes y luego las fotos, ya que las fotos tienen una clave foránea sobre álbumes.
# En cada migración aparece la fecha_hora_nombre.. Si queremos cambiar el orden simplemente modificando la fecha_hora (lo que nos se

php artisan make:migration --create="albumes" CrearTablaAlbumes
#Created Migration: 2015_04_29_143326_CrearTablaAlbumes

php artisan make:migration --create="fotos" CrearTablaFotos
Created Migration: 2015_04_29_143830_CrearTablaFotos
```

- **Borramos** la migración **database/migrations/create_password_resets_table.php**
- Editamos la migración **database/migrations/2014_10_12_000000_create_users_table.php**.

7.1 Migración de Usuarios

- Contenido de **database/migrations/2014_10_12_000000_create_users_table.php**:

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration {

/**
         * Run the migrations.
         *
         * @return void
         */
public function up()
{
Schema::create('usuarios', function(Blueprint $table)
{
$table->increments('id');
$table->string('nombre');
$table->string('email')->unique();
$table->string('password', 60);
$table->string('pregunta');
$table->string('respuesta');
$table->rememberToken();
$table->timestamps();
});
}

/**
         * Reverse the migrations.
         *
         * @return void
         */
public function down()
```

```
{
Schema::drop('usuarios');
}

}
```

7.2 Migración de Albumes

- Contenido de **database/migrations/2015_04_28_175836_create_albums_table.php**:

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAlbumsTable extends Migration {

/**
         * Run the migrations.
         *
         * @return void
         */
public function up()
{
Schema::create('albumes', function(Blueprint $table)
{
$table->increments('id');
$table->string('nombre');
$table->string('descripcion');
$table->integer('usuario_id')->unsigned();

// Relaciones con las otras tablas:
$table->foreign('usuario_id')->references('id')->on('usuarios');

$table->timestamps();
});
}

/**
         * Reverse the migrations.
         *
         * @return void
         */
public function down()
{
Schema::drop('albumes');
}

}
```

7.3 Migración de Fotos

- Contenido de **database/migrations/2015_04_28_180148_create_fotos_table**:

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateFotosTable extends Migration {

/**
         * Run the migrations.
         *
         * @return void
         */
public function up()
{
Schema::create('fotos', function(Blueprint $table)
{
```

```
$table->increments('id');
$table->string('nombre');
$table->string('descripcion');
$table->string('ruta');
$table->integer('album_id')->unsigned();

// Relaciones con las otras tablas:
$table->foreign('album_id')->references('id')->on('albumes');
$table->timestamps();
});
}

/**
         * Reverse the migrations.
         *
         * @return void
         */
public function down()
{
Schema::drop('fotos');
}

}
```

8 Seeding de las tablas

- A continuación vamos a rellenar las tablas con datos automáticos para hacer algunas pruebas.
- Configuramos el fichero de llamadas a los Seeder en **database/seeds/DatabaseSeeder.php**:

8.1 DatabaseSeeder

- Contenido del fichero **database/seeds/DatabaseSeeder.php**:

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

// Hacemos uso de los modelos de nuestra aplicación
use Enfocalia\Foto;
use Enfocalia\Album;
use Enfocalia\Usuario;



class DatabaseSeeder extends Seeder {

/**
         * Run the database seeds.
         *
         * @return void
         */
public function run()
{
// Para que no verifique el control de claves foráneas al hacer el truncate haremos:
DB::statement('SET FOREIGN_KEY_CHECKS=0');

// Primero hacemos un truncate de las tablas para que no se estén agregando datos
// cada vez que ejecutamos el seeder.
Foto::truncate();
Album::truncate();
Usuario::truncate();

// Es importante el orden de llamada.
$this->call('UsuariosSeeder');
$this->call('AlbumesSeeder');
$this->call('FotosSeeder');
}

}
```

8.2 Seeder de Usuarios

- Duplicamos el fichero **database/seeds/DatabaseSeeder.php** con el nombre **database/seeds/UsuariosSeeder.php**.
- Contenido del fichero **database/seeds/UsuariosSeeder.php**:

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

// Hacemos uso de los modelos de nuestra aplicación
use Enfocalia\Foto;
use Enfocalia\Album;
use Enfocalia\Usuario;

class UsuariosSeeder extends Seeder {

/**
         * Run the database seeds.
         *
         * @return void
         */
public function run()
{
for ($i=0;$i<50;$i++)
{
Usuario::create(
[
'nombre'=> "usuario$i",
'email' => "email$i@test.com",
'password' => bcrypt("pass$i"), // bcrypt es una function helper de Hash::make
//'password' => Hash::make("pass$i")
'pregunta' => "preg$i",
"respuesta" => "resp$i"
]);
}
}
}
```

8.3 Seeder de Albumes

- Duplicamos el fichero **database/seeds/DatabaseSeeder.php** con el nombre **database/seeds/AlbumesSeeder.php**.
- Contenido del fichero **database/seeds/AlbumesSeeder.php**:

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

// Hacemos uso de los modelos de nuestra aplicación
use Enfocalia\Foto;
use Enfocalia\Album;
use Enfocalia\Usuario;

class AlbumesSeeder extends Seeder {

/**
         * Run the database seeds.
         *
         * @return void
         */
public function run()
{
$usuarios = Usuario::all();
$contador=0;

foreach ($usuarios as $usuario)
{
$cantidad = mt_rand(0,15);
for ($i=0;$i< $cantidad;$i++)
```

```
{
$contador++;
Album::create(
[
'nombre' => "Nombre Album_$contador",
'descripcion' => "Descripcion Album_$contador",
'usuario_id' => $usuario->id
]);
}
}
}
}
```

8.4 Seeder de Fotos

- Duplicamos el fichero **database/seeds/AlbumesSeeder.php** con el nombre **database/seeds/FotosSeeder.php**.
- Contenido del fichero **database/seeds/FotosSeeder.php**:

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;

// Hacemos uso de los modelos de nuestra aplicación
use Enfocalia\Foto;
use Enfocalia\Album;
use Enfocalia\Usuario;

class FotosSeeder extends Seeder {

/**
         * Run the database seeds.
         *
         * @return void
         */
public function run()
{
$albumes = Album::all();
$contador=0;

foreach ($albumes as $album)
{
$cantidad = mt_rand(0,5);
for ($i=0;$i< $cantidad;$i++)
{
$contador++;
Foto::create(
[
'nombre' => "Nombre Foto_$contador",
'descripcion' => "Descripcion Foto_$contador",
'ruta' => '/img/test.png',
'album_id' => $album->id
]);
}
}
}
}
```

9 Control de excepciones en Laravel

- Veamos como controlar los típicos errores que se producen cuando surge alguna excepción en Laravel.
- El típico mensaje de **"Whoops, looks like something went wrong."**.
- Para evitar que se muestren todos los detalles de los errores (**en entornos de producción**) tendríamos que modificar en el fichero **.env**:

```
APP_DEBUG=false
```

- En este caso lo dejamos como está a **APP_DEBUG=true**.
- En el fichero **app/Exceptions/Handler.php** se gestiona el control de excepciones.

- Dentro de ese fichero, el método **report** se encarga de guardar un log en el archivo de logs con cada uno de los errores.
- El método **render** se encarga de mostrar cada uno de los errores.

- Editaremos el fichero **app/Exceptions/Handler.php** para gestionar nuestras excepciones. Lo que queremos hacer es que si por ejemplo falla el token csrf o se genere una excepción que que muestre un error de "Algo salió mal".

- Contenido del fichero **app/Exceptions/Handler.php**:

```
<?php namespace Enfocalia\Exceptions;

use Exception;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

// Añadimos esta clase:
use Illuminate\Session\TokenMismatchException;

class Handler extends ExceptionHandler {

/**
         * A list of the exception types that should not be reported.
         *
         * @var array
         */
protected $dontReport = [
'Symfony\Component\HttpKernel\Exception\HttpException'
];

/**
         * Report or log an exception.
         *
         * This is a great spot to send exceptions to Sentry, Bugsnag, etc.
         *
         * @param  \Exception  $e
         * @return void
         */
public function report(Exception $e)
{
return parent::report($e);
}

/**
         * Render an exception into an HTTP response.
         *
         * @param  \Illuminate\Http\Request  $request
         * @param  \Exception  $e
         * @return \Illuminate\Http\Response
         */
public function render($request, Exception $e)
{
// Si la excepción es una instancia de TokenMismatchException
if ($e instanceof TokenMismatchException)
{
// Redireccionamos a la URL de dónde proviene la petición, mandando un mensaje en una variable de sesión "csrf".
// La URL de dónde proviene la petición suelen ser los formularios.
// Tendremos que modificar las vistas correspondientes: validacion/inicio.blade.php, bienvenida.blade.php e inicio.blade.php
return redirect($request->url())->with('csrf','Pasó mucho tiempo, inténtalo de nuevo.');
}

// Renderizará el error cuando estemos en entorno local (APP_DEBUG=true)
if (config('app.debug'))
{
return parent::render($request,$e);
}

// En otro caso lo mandamos a la dirección de inicio.
return redirect('/')->with('error','Algo salió mal.');
}

}
```

- Tenemos que **modificar las vistas** para mostrar el error **csrf** o **error**.

- Contenido de **resources/views/validacion/inicio.blade.php**:

```
@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">Iniciar Sesión</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

@if (Session::has('csrf'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('csrf')}}
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validacion/inicio') }}">
<input type="hidden" name="_token" value="{{ csrf_token() }}">

<div class="form-group">
<label class="col-md-4 control-label">Correo Electrónico</label>
<div class="col-md-6">
<input type="email" class="form-control" name="email" value="{{ old('email') }}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Password</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<div class="checkbox">
<label>
<input type="checkbox" name="remember"> Recordarme
</label>
</div>
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">Iniciar Sesión</button>

<a class="btn btn-link" href="{{ url('/validacion/recuperar') }}">¿Olvidó su contraseña?</a>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Contenido de **resources/views/bienvenida.blade.php**:

```
@extends('app')

@section('content')


@if (Session::has('error'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('error')}}
</div>
@endif

<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Por favor inicie sesión para acceder al sistema.
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Contenido de **resources/views/inicio.blade.php**:

```
@extends('app')

@section('content')



@if (Session::has('error'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('error')}}
</div>
@endif

<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Bienvenido (usuario)
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Si desactivamos en **.env APP_DEBUG=false** veremos todos los mensajes de errores personalizados.
- Para el resto del curso pondremos en el fichero **.env** la opción **APP_DEBUG=true** para tener más información sobre los fallos.

10 Validación de usuarios

- Acuérdate de poner en el fichero **.env APP_DEBUG=true**.
- **Si probamos a validarnos veremos que nos va a dar un mensaje de error** y eso es por que tenemos que realizar todavía algunos cambios en el sistema de validación.
- **Por defecto Laravel viene preparado para utilizar la tabla Users y nosotros lo hemos cambiado para usar la tabla Usuarios.**
- Así que tendremos que realizar los siguientes cambios en los siguientes ficheros.

10.1 app/services/Registrar.php

- En este fichero Laravel lleva el control del sistema de registro automático de usuarios y de las **reglas de validación** que deben cumplir los campos antes de grabarlos en la tabla.
- Tenemos que adaptarlo a las características de la nueva tabla de usuarios.

- Contenido del fichero **app/Services/Registrar.php**:

```
<?php namespace Enfocalia\Services;

use Enfocalia\Usuario;
use Validator;
use Illuminate\Contracts\Auth\Registrar as RegistrarContract;

class Registrar implements RegistrarContract {

/**
         * Get a validator for an incoming registration request.
         *
         * @param  array  $data
         * @return \Illuminate\Contracts\Validation\Validator
         */
public function validator(array $data)
{
return Validator::make($data, [
'nombre' => 'required|max:255',
'email' => 'required|email|max:255|unique:usuarios',
'password' => 'required|confirmed|min:6',
'pregunta' => 'required|max:255',
'respuesta' => 'required|max:255'
]);
}

/**
         * Create a new user instance after a valid registration.
         *
         * @param  array  $data
         * @return User
         */
public function create(array $data)
{
return Usuario::create([
'nombre' => $data['nombre'],
'email' => $data['email'],
'password' => bcrypt($data['password']),
'pregunta' => $data['pregunta'],
'respuesta' => $data['respuesta']
]);
}
}
```

10.2 config/auth.php

- En este fichero Laravel configura los parámetros de autenticación para la aplicación..
- Tenemos que adaptarlo a las características de la nueva tabla de usuarios.

- Contenido del fichero **config/auth.php**:

```
<?php

return [

/*
        |--------------------------------------------------------------------------
        | Default Authentication Driver
        |--------------------------------------------------------------------------
        |
        | This option controls the authentication driver that will be utilized.
        | This driver manages the retrieval and authentication of the users
```

```
        | attempting to get access to protected areas of your application.
        |
        | Supported: "database", "eloquent"
        |
        */

    'driver' => 'eloquent',

    /*
        |--------------------------------------------------------------------------
        | Authentication Model
        |--------------------------------------------------------------------------
        |
        | When using the "Eloquent" authentication driver, we need to know which
        | Eloquent model should be used to retrieve your users. Of course, it
        | is often just the "User" model but you may use whatever you like.
        |
        */

    'model' => 'Enfocalia\Usuario',

    /*
        |--------------------------------------------------------------------------
        | Authentication Table
        |--------------------------------------------------------------------------
        |
        | When using the "Database" authentication driver, we need to know which
        | table should be used to retrieve your users. We have chosen a basic
        | default value but you may easily change it to any table you like.
        |
        */

    'table' => 'usuarios',

    /*
        |--------------------------------------------------------------------------
        | Password Reset Settings
        |--------------------------------------------------------------------------
        |
        | Here you may set the options for resetting passwords including the view
        | that is your password reset e-mail. You can also set the name of the
        | table that maintains all of the reset tokens for your application.
        |
        | The expire time is the number of minutes that the reset token should be
        | considered valid. This security feature keeps tokens short-lived so
        | they have less time to be guessed. You may change this as needed.
        |
        */

    'password' => [
    'email' => 'emails.password',
    'table' => 'password_resets',
    'expire' => 60,
    ],

    ];
```

10.3 Vista inicio.blade.php

- La vamos a modificar para que en el mensaje de Bienvenido(Usuario) ponga el nombre de ese usuario
- Contenido del fichero **resources/views/inicio.blade.php**:

```
@extends('app')

@section('content')



@if (Session::has('error'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('error')}}
```

```
</div>
@endif

<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Bienvenido ({{Auth::user()->nombre}})
</div>
</div>
</div>
</div>
</div>
@endsection
```

11 Los Requests en Laravel 5

- Si examinamos el código del controlador app/Http/Controllers/ValidacionController.php veremos que en el método postInicio hacemos una validación de datos recibidos para iniciar sesión.
- Pues bien con Laravel 5 podríamos crear una nueva clase de tipo Request y especificar allí dentro dichas reglas de validación, de tal forma que a la hora de acceder al método postInicio indicamos que debe cumplir esas reglas antes de ejecutar las instrucciones que están dentro de dicho método.

- Podemos crear una clase de tipo Request con:

```
php artisan make:request IniciarSesionRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/IniciarSesionRequest.php**:

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

use Enfocalia\Http\Requests\Request;

class IniciarSesionRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Devuelve verdadero o falso.
// Si devuelve falso es que el usuario no tiene permiso para realizar este tipo de operación.
// Pero en este caso para hacer el inicio de sesión todos podrán intentar dicho inicio.
// Si devolviéramos false al intentar ejecutar esta validación devuelve un error con código "403" con la respuesta "Forbidden"
// Devolvemos true en este caso.
return true;
}

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

- Contenido del fichero **app/Http/Controllers/Validacion/ValidacionController.php**:

```
<?php namespace Enfocalia\Http\Controllers\Validacion;

use Enfocalia\Http\Controllers\Controller;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

// Añadimos la clase Request;
use Illuminate\Http\Request;

// Añadimos el Request que hicimos para postInicio:
use Enfocalia\Http\Requests\IniciarSesionRequest;

class ValidacionController extends Controller {
protected $auth;
protected $registrar;

public function __construct(Guard $auth, Registrar $registrar)
{
$this->auth = $auth;
$this->registrar = $registrar;

// Sustituir getLogout por getSalida
$this->middleware('guest', ['except' => 'getSalida']);
}

public function getRegistro()
{
//return "formulario de creación de cuentas.";
return view('validacion.registro');
}

public function postRegistro(Request $request)
{
$validator = $this->registrar->validator($request->all());

if ($validator->fails())
{
$this->throwValidationException(
$request, $validator
);
}

$this->auth->login($this->registrar->create($request->all()));

return redirect($this->redirectPath());
}

public function getInicio()
{
//return 'Mostrando formulario Inicio Sesión';
return view('validacion.inicio');
}


public function postInicio(IniciarSesionRequest $request)
{
$credentials = $request->only('email', 'password');

if ($this->auth->attempt($credentials, $request->has('remember')))
{
return redirect()->intended($this->redirectPath());
}

return redirect($this->loginPath())
->withInput($request->only('email', 'remember'))
->withErrors([
'email' => $this->getFailedLoginMessage(),
]);
}
```

```
protected function getFailedLoginMessage()
{
return 'E-mail o contraseña incorrectos.';
}

public function getSalida()
{
$this->auth->logout();

return redirect(property_exists($this, 'redirectAfterLogout') ? $this->redirectAfterLogout : '/');
}

/**
         * Get the post register / login redirect path.
         *
         * @return string
         */
public function redirectPath()
{
if (property_exists($this, 'redirectPath'))
{
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->redirectTo : '/inicio';
}

public function loginPath()
{
return property_exists($this, 'loginPath') ? $this->loginPath : '/validacion/inicio';
}

public function getRecuperar()
{
//return "recuperar contraseña";
return view('validacion.recuperar');
}

public function postRecuperar()
{
return "recuperando contraseña";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

12 Recuperación de contraseñas

- Veamos como podemos implementar la opción de recuperar la contraseña en el controlador **ValidacionController**.
- Para ello nos vamos a crear un **Request** -> **RecuperarContrasenaRequest** para validar que los campos recibidos sean los correctos.
- Nos creamos primero el Request:

```
php artisan make:request RecuperarContrasenaRequest
#Request created successfully.
```

- Contenido del fichero **app**/**Http**/**Requests**/**RecuperarContrasenaRequest**:

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

use Enfocalia\Http\Requests\Request;

class RecuperarContrasenaRequest 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 [
'email' => 'required|email|exists:usuarios',
'password' => 'required|min:6|confirmed',
/*
                Confirmed indica que debe existir otro campo que se llame campo_confirmation
                y que contenga los mismos valores.
                En este caso se tiene que estar recibiendo también un password_confirmation con el mismo valor que password.
                */
'pregunta' => 'required',
'respuesta' => 'required'
];
}

}
```

- Contenido del fichero **app/Http/Controllers/Validacion/ValidacionController.php** usando la Request creada anteriormente en el **método postRecuperar()**:

```
<?php namespace Enfocalia\Http\Controllers\Validacion;

use Enfocalia\Http\Controllers\Controller;
use Illuminate\Contracts\Auth\Guard;
use Illuminate\Contracts\Auth\Registrar;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

// Añadimos la clase Request;
use Illuminate\Http\Request;

// Añadimos el Request que hicimos para postInicio:
use Enfocalia\Http\Requests\IniciarSesionRequest;

// Añadimos el Request que hicimos para postRecuperar:
use Enfocalia\Http\Requests\RecuperarContrasenaRequest;

// Necesitamos el modelo Usuario.
use Enfocalia\Usuario;


class ValidacionController extends Controller {
protected $auth;
protected $registrar;

public function __construct(Guard $auth, Registrar $registrar)
{
$this->auth = $auth;
$this->registrar = $registrar;

// Sustituir getLogout por getSalida
$this->middleware('guest', ['except' => 'getSalida']);
}

public function getRegistro()
{
//return "formulario de creación de cuentas.";
return view('validacion.registro');
}

public function postRegistro(Request $request)
{
$validator = $this->registrar->validator($request->all());
```

```
if ($validator->fails())
{
$this->throwValidationException(
$request, $validator
);
}

$this->auth->login($this->registrar->create($request->all()));

return redirect($this->redirectPath());
}

public function getInicio()
{
//return 'Mostrando formulario Inicio Sesión';
return view('validacion.inicio');
}


public function postInicio(IniciarSesionRequest $request)
{
$credentials = $request->only('email', 'password');

if ($this->auth->attempt($credentials, $request->has('remember')))
{
return redirect()->intended($this->redirectPath());
}

return redirect($this->loginPath())
->withInput($request->only('email', 'remember'))
->withErrors([
'email' => $this->getFailedLoginMessage(),
]);
}

protected function getFailedLoginMessage()
{
return 'E-mail o contraseña incorrectos.';
}

public function getSalida()
{
$this->auth->logout();

return redirect(property_exists($this, 'redirectAfterLogout') ? $this->redirectAfterLogout : '/');
}

/**
         * Get the post register / login redirect path.
         *
         * @return string
         */
public function redirectPath()
{
if (property_exists($this, 'redirectPath'))
{
return $this->redirectPath;
}

return property_exists($this, 'redirectTo') ? $this->redirectTo : '/inicio';
}

public function loginPath()
{
return property_exists($this, 'loginPath') ? $this->loginPath : '/validacion/inicio';
}

public function getRecuperar()
{
//return "recuperar contraseña";
return view('validacion.recuperar');
}
```

```
public function postRecuperar(RecuperarContrasenaRequest $request)
{
//return "recuperando contraseña";
$pregunta=$request->get('pregunta');
$respuesta=$request->get('respuesta');

$email=$request->get('email');
$usuario=Usuario::where('email','=',$email)->first();

if($pregunta=== $usuario->pregunta && $respuesta===$usuario->respuesta)
{
// Modificamos por lo tanto su contraseña con la nueva recibida.
$usuario->password=bcrypt($request->get('password'));

// Grabamos los cambios.
$usuario->save();

// Redireccionamos a Validacion/inicio pero sin valores y con un mensaje de actualizado.
return redirect('/validacion/inicio')
->with(['recuperada'=>'La contraseña se modificó correctamente. Inicie Sesión.']);
}

// En el caso de que no coincidan las preguntas y respuestas lo mandamos de nuevo a /validacion/recuperar.
return redirect('/validacion/recuperar')
->withInput($request->only('email','pregunta'))
->withErrors(['pregunta'=>'La pregunta y/o respuesta facilitadas no coinciden.']);
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Contenido del fichero **resources/views/validacion/inicio.blade.php**:

```
@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">Iniciar Sesión</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

@if (Session::has('csrf'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('csrf')}}
</div>
@endif

@if (Session::has('recuperada'))
<div class="alert alert-success">
{{Session::get('recuperada')}}
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validacion/inicio') }}">
<input type="hidden" name="_token" value="{{ csrf_token() }}">
```

```
<div class="form-group">
<label class="col-md-4 control-label">Correo Electrónico</label>
<div class="col-md-6">
<input type="email" class="form-control" name="email" value="{{ old('email') }}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Password</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<div class="checkbox">
<label>
<input type="checkbox" name="remember"> Recordarme
</label>
</div>
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">Iniciar Sesión</button>

<a class="btn btn-link" href="{{ url('/validacion/recuperar') }}">¿Olvidó su contraseña?</a>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

13 Edición de perfiles de usuario

- **Añadimos una opción al menú** para actualizar el perfil.
- Contenido del fichero **resources/views/app.blade.php**:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laravel</title>

<link href="{{ asset('/css/app.css') }}" rel="stylesheet">


<link href='//fonts.googleapis.com/css?family=Roboto:400,300' rel='stylesheet' type='text/css'>




</head>
<body>
        <nav class="navbar navbar-default">
                <div class="container-fluid">
                        <div class="navbar-header">
                                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-exampl
                                        <span class="sr-only">Toggle Navigation</span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
```

```
                        </button>
                        <a class="navbar-brand" href="#">Enfocalia</a>
                    </div>

                    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                        <ul class="nav navbar-nav">
                <li><a href="{{ url('
                /') }}">Inicio</a></li>
                        </ul>

                        <ul class="nav navbar-nav navbar-right">
                            @if (Auth::guest())
                <li><a href="{{ url('
                /validacion/inicio') }}">Iniciar Sesión</a></li>
                <li><a href="{{ url('
                /validacion/registro') }}">Registrarse</a></li>
                            @else
                                <li class="dropdown">
                                    <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" ari
                                    <ul class="dropdown-menu" role="menu">
                <li><a href="{{ url('
                /validado/usuario/editar-perfil') }}">Actualizar perfil</a></li>
                <li><a href="{{ url('
                /validacion/salida') }}">Salir</a></li>    <li><a href="{{ url('
                                    </ul>
                                </li>
                            @endif
                        </ul>
                    </div>
                </div>
            </nav>

            @yield('content')


        <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
        <script src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.1/js/bootstrap.min.js"></script>
        </body>
        </html>
```

- Creamos una nueva carpeta llamada **resources/views/usuario**.
- Dentro de la carpeta crearemos una nueva vista **actualizar.blade.php**.
- Copiaremos el código de la vista **resources/views/validacion/registro.blade.php** y lo pegaremos dentro de **resources/views/usuario/actualizar.blade.php**.

- Contenido del fichero **resources/views/usuario/actualizar.blade.php**:

```
@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">Actualizar datos de perfil</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validado/usuario/editar-perfil') }}">
<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="{{ Auth::user()->nombre }}">
</div>
</div>
```

```
<div class="form-group">
<label class="col-md-4 control-label">Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Confirmar Contraseña</label>
<div class="col-md-6">
<input type="password" class="form-control" name="password_confirmation">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Pregunta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="pregunta" value="{{Auth::user()->pregunta}}">
</div>
</div>

<div class="form-group">
<label class="col-md-4 control-label">Respuesta</label>
<div class="col-md-6">
<input type="text" class="form-control" name="respuesta" value="{{Auth::user()->respuesta}}">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">
Actualizar Perfil
</button>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Editamos el controlador **app/Http/Controllers/UsuarioController.php** los métodos **getEditarPerfil()** y **PostEditarPerfil()**:

```
<?php namespace Enfocalia\Http\Controllers;

use Enfocalia\Http\Requests\EditarPerfilRequest;
use Auth;

class UsuarioController extends Controller {

// Para el tema de contraseñas se encargará el controlador de Validación

public function __construct()
{
$this->middleware('auth');
}

public function getEditarPerfil()
{
//return "Mostrando formulario de perfil.";
return view('usuario.actualizar');
}

public function postEditarPerfil(EditarPerfilRequest $request)
{
//return "Generando actualización de perfil...";
// Modificamos los datos del usuario autenticado.
$usuario = Auth::user();

$usuario->nombre=$request->get('nombre');
```

```
if($request->has('password'))
{
$usuario->password=bcrypt($request->get('password'));
}

if($request->has('pregunta'))
{
$usuario->pregunta=$request->get('pregunta');
$usuario->respuesta=$request->get('respuesta');
}

$usuario->save();

// Tendremos que modificar la vista de inicio.blade.php para incluir este mensaje.
return redirect('/validado')->with('actualizado','Su perfil ha sido actualizado correctamente.');
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Creamos el nuevo Request llamado **EditarPerfilRequest** que estamos usando en el controlador anterior **UsuarioController**.

```
php artisan make:request EditarPerfilRequest
#Request created successfully.
```

- Nos copiamos el contenido de **RecuperarContrasenaRequest** y lo pegamos en **EditarPerfilRequest**.
- Contenido del fichero **app**/**Http**/**Requests**/**EditarPerfilRequest**:

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

use Enfocalia\Http\Requests\Request;

class EditarPerfilRequest 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',
'password' => 'min:6|confirmed',  // No necesariamente es requerida. Debe ser confirmada si la contraseña está.
/*
                Confirmed indica que debe existir otro campo que se llame campo_confirmation
                y que contenga los mismos valores.
                En este caso se tiene que estar recibiendo también un password_confirmation con el mismo valor que password.
                */
'pregunta' => '',  // no tiene ninguna restriccion.
'respuesta' => 'required_with:pregunta' // La respuesta es requerida siempre y cuando recibamos una pregunta.
];
}

}
```

- Contenido del fichero **resources/views/inicio.blade.php** para incluir el mensaje de **actualizado**:

```
@extends('app')

@section('content')



@if (Session::has('error'))
<div class="alert alert-danger">
<strong>Whoops!</strong> Ha surgido un problema.<br><br>
{{Session::get('error')}}
</div>
@endif

@if (Session::has('actualizado'))
<div class="alert alert-success">
{{Session::get('actualizado')}}
</div>
@endif

<div class="container">
<div class="row">
<div class="col-md-10 col-md-offset-1">
<div class="panel panel-default">
<div class="panel-heading">Inicio</div>

<div class="panel-body">
Bienvenido ({{Auth::user()->nombre}})
</div>
</div>
</div>
</div>
</div>
@endsection
```

14 Visualizar álbumes y fotos

14.1 Mostrar álbumes

- Para gestionar los álbumes lo hacemos a través del controlador **AlbumController**.

- Contenido del fichero **app/Http/Controllers/AlbumController**:

```
<?php namespace Enfocalia\Http\Controllers;

use Auth;

class AlbumController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
//return "Mostrando álbumes del usuario.";

// Obtenemos cual es el usuario que está validado en el sistema.
$usuario =Auth::user();

// Álbumes de ese usuario.
$albumes=$usuario->albumes;

// Devolvemos la vista pasándole todos los álbumes que hemos obtenido.
return view('albumes.mostrar',['albumes'=>$albumes]);
}

public function getCrearAlbum()
```

```
{
return "Formulario de creación Albumes.";
}

public function postCrearAlbum()
{
return "Almacenando Albumes...";
}

public function getActualizarAlbum()
{
return "Formulario de actualización de Albumes.";
}

public function postActualizarAlbum()
{
return "Actualizando Album...";
}

public function getEliminarAlbum()
{
return "Formulario de eliminación de Albumes.";
}

public function postEliminarAlbum()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

- Añadimos a la aplicación una nueva opción "Mis Álbumes " dentro de la vista app.blade.php.
- Contenido del fichero **resources/views/app.blade.php**:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laravel</title>

<link href="{{ asset('/css/app.css') }}" rel="stylesheet">


<link href='//fonts.googleapis.com/css?family=Roboto:400,300' rel='stylesheet' type='text/css'>




</head>
<body>
        <nav class="navbar navbar-default">
                <div class="container-fluid">
                        <div class="navbar-header">
                                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-exampl
                                        <span class="sr-only">Toggle Navigation</span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                        <span class="icon-bar"></span>
                                </button>
                                <a class="navbar-brand" href="#">Enfocalia</a>
                        </div>

                        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                                <ul class="nav navbar-nav">
                    /') }}">Inicio</a></li><a href="{{ url('
```

```
                /validado/albumes')<li><a href="{{ url('Mis Álbumes</a></li>
                        </ul>

                        <ul class="nav navbar-nav navbar-right">
                                @if (Auth::guest())
                /validacion/inicio') }}">Iniciar Sesión</a></li>
                /validacion/registro') }}">Registrarse</a></li>
                                @else
                                        <li class="dropdown">
                                                <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" ari
                                                <ul class="dropdown-menu" role="menu">
                /validado/usuario/editar-perfil') }}">Actualizar perfil</a></li>
                /validacion/salida') }}">Salir</a></li>    <li><a href="{{ url('
                                                </ul>
                                        </li>
                                @endif
                        </ul>
                </div>
        </div>
</nav>

@yield('content')


<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.1/js/bootstrap.min.js"></script>
</body>
</html>
```

- Crearemos carpeta **resources/views/albumes**.
- Creamos vista **resources/views/albumes/mostrar.blade.php**.
- Contenido de **resources/views/albumes/mostrar.blade.php**.

```
@extends('app')

@section('content')
<div class="container-fluid">
<p><a href="/validado/albumes/crear-album" class="btn btn-primary" role="button">Crear Álbum</a></p>
@if (sizeof($albumes)>0)

@foreach ($albumes as $index=>$album)
@if ($index %3 ==0)
<div class="row">
@endif
<div class="col-sm-6 col-md-4">
<div class="thumbnail">
<div class="caption">
<h3>{{$album->nombre}}</h3>

<p>{{$album->descripcion}}</p>
<p><a href="/validado/fotos?id={{$album->id}}" class="btn btn-primary" role="button">Ver</a></p>
</div>
</div>
</div>
@if (($index+1)%3 == 0)
</div>
@endif

@endforeach
@else
<div class="alert alert-danger">
<p>No tienes álbumes de fotografías. Crea uno.</p>
</div>
@endif
</div>
@endsection
```

14.2 Mostrar Fotos

- Para gestionar la opción de ver Fotos lo hacemos a través del controlador **FotoController**.

- Tenemos que asegurarnos que a la hora de ver las fotos sean nuestras fotos y no las de otra persona, con lo que crearemos un Request **MostrarFotosRequest**.
- Crearemos el Request **MostrarFotosRequest**:

```
php artisan make:request MostrarFotosRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/MostrarFotosRequest**:

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

use Enfocalia\Http\Requests\Request;
use Auth;
use Enfocalia\Album;

class MostrarFotosRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
$user = Auth::user();

// Id del album recibido
$id = $this->get('id');

// Buscamos si ese usuario tiene un album con ese $id.
$album = $user->albumes()->find($id);

// Si ese album existe devolvemos true, en otro caso false (forbidden).
if ($album)
return true;
else
return false;
}

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

}
```

- Contenido del fichero **app/Http/Controllers/FotoController**:

```
<?php namespace Enfocalia\Http\Controllers;

use Enfocalia\Http\Requests\MostrarFotosRequest;
use Enfocalia\Album;
use Enfocalia\Foto;


class FotoController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex(MostrarFotosRequest $request)
{
```

```
//return "Mostrando las fotos del usuario.";
$id = $request->get('id');

// Obtenemos las fotos de ese album con ese id.
$fotos = Album::find($id)->fotos;

// Devolvemos una vista con las fotos de ese álbum.
return view('fotos.mostrar',['fotos'=>$fotos]);
}

public function getCrearFoto()
{
return "Formulario de creación fotos.";
}

public function postCrearFoto()
{
return "Almacenando fotos...";
}

public function getActualizarFoto()
{
return "Formulario de actualización de fotos.";
}

public function postActualizarFoto()
{
return "Actualizando foto...";
}

public function getEliminarFoto()
{
return "Formulario de eliminación de fotos.";
}

public function postEliminarFoto()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Nos falta crear la vista **mostrar.blade.php**.
- Creamos una carpeta **resources/views/fotos**.
- Creamos una vista **resources/views/fotos/mostrar.blade.php** con el contenido del mismo fichero pero de la carpeta **álbumes**.
- Contenido del fichero **resources/views/fotos/mostrar.blade.php**:

```
@extends('app')

@section('content')

<div class="container-fluid">
<p><a href="/validado/fotos/crear-foto?id={{$id}}" class="btn btn-primary" role="button">Crear Foto</a></p>
@if(sizeof($fotos) > 0)
@foreach($fotos as $index => $foto)
@if($index%4 == 0)
<div class="row">
@endif
  <div class="col-sm-6 col-md-3">
    <div class="thumbnail">
        <img src="{{$foto->ruta}}">
      <div class="caption">
        <h3>{{$foto->nombre}}</h3>
        <p>{{$foto->descripcion}}</p>
      </div>
      <p><a href="/validado/fotos/actualizar-foto/{{$foto->id}}" class="btn btn-primary" role="button">Editar Foto</a></p>
      <form action="/validado/fotos/eliminar-foto" method="POST">
```

```
<input type="hidden" name="_token" value="{{ csrf_token() }}" required>
<input type="hidden" name="id" value="{{$foto->id}}" required>
<input class="btn btn-danger" role="button" type="submit" value="Eliminar Foto"/>
</form>
    </div>
  </div>
@if(($index+1)%4 == 0)
</div>
@endif
@endforeach
@else
<div class="alert alert-danger">
<p>Al parecer este album no tiene fotos. Crea una.</p>
</div>
@endif
</div>
@endsection
```

15 Creación de Álbumes y Fotos

15.1 Creación de Álbumes

- Abrimos el controlador **AlbumController** y editamos el método **getCrearAlbum()**.
- Contenido del fichero **app/Http/Controllers/AlbumController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Auth;
use Enfocalia\Http\Requests\CrearAlbumRequest;
use Enfocalia\Album;

class AlbumController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
//return "Mostrando álbumes del usuario.";

// Obtenemos cual es el usuario que está validado en el sistema.
$usuario =Auth::user();

// Álbumes de ese usuario.
$albumes=$usuario->albumes;

// Devolvemos la vista pasándole todos los álbumes que hemos obtenido.
return view('albumes.mostrar',['albumes'=>$albumes]);
}

public function getCrearAlbum()
{
//return "Formulario de creación Albumes.";
return view('albumes.crear-album');
}

public function postCrearAlbum()
{
return "Almacenando Albumes...";

}

public function getActualizarAlbum()
{
return "Formulario de actualización de Albumes.";
}

public function postActualizarAlbum()
```

```
{
return "Actualizando Album...";
}

public function getEliminarAlbum()
{
return "Formulario de eliminación de Albumes.";
}

public function postEliminarAlbum()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

- Creamos la vista **resources/views/albumes/crear-album.blade.php**.
- Nos **copiamos** el contenido de **resources/views/usuario/actualizar.blade.php** y lo **pegamos** en el fichero anterior.
- Contenido del fichero **resources/views/albumes/crear-album.blade.php**:

```
@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($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/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">Descripción</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 Álbum
</button>
</div>
</div>
</form>
```

```
</div>
</div>
</div>
</div>
</div>
@endsection
```

- A continuación gestionamos el método **postCrearAlbum()**.
- Contenido del fichero **app/Http/Controllers/AlbumController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Auth;
use Enfocalia\Http\Requests\CrearAlbumRequest;
use Enfocalia\Album;

class AlbumController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
//return "Mostrando álbumes del usuario.";

// Obtenemos cual es el usuario que está validado en el sistema.
$usuario =Auth::user();

// Álbumes de ese usuario.
$albumes=$usuario->albumes;

// Devolvemos la vista pasándole todos los álbumes que hemos obtenido.
return view('albumes.mostrar',['albumes'=>$albumes]);
}

public function getCrearAlbum()
{
//return "Formulario de creación Albumes.";
return view('albumes.crear-album');
}

public function postCrearAlbum(CrearAlbumRequest $request)
{
//return "Almacenando Albumes...";

// Obtenemos el usuario conectado.
$usuario = Auth::user();

// Creamos el álbum.
Album::create(
[
'nombre' => $request->get('nombre'),
'descripcion' => $request->get('descripcion'),
'usuario_id' => $usuario->id
]);

// Al hacer el redirect pasando variables esas variables se accederá a ellas a partir de
// las variables de sesión: @if(Session::has('creado'))  y  {{Session::get('creado')}}
return redirect('/validado/albumes')->with('creado','El álbum ha sido creado correctamente.');
}

public function getActualizarAlbum()
{
return "Formulario de actualización de Albumes.";
}

public function postActualizarAlbum()
{
return "Actualizando Album...";
}
```

```
public function getEliminarAlbum()
{
return "Formulario de eliminación de Albumes.";
}

public function postEliminarAlbum()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

- Creación del **Request CrearAlbumRequest**.

```
php artisan make:request CrearAlbumRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/CrearAlbumRequest.php**:

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

use Enfocalia\Http\Requests\Request;

class CrearAlbumRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// La autorización siempre será true por que cualquier usuario podrá
// crear álbumes y nosotros se los asociaremos al usuario
// que tiene la sesión creada.
return true;
}

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

}
```

- Modificación de **resources/views/albumes/mostrar-blade.php** para mostrar el mensaje de **Álbum creado correctamente**:

```
@extends('app')

@section('content')
@if(Session::has('creado'))
<div class="alert alert-success">
<p>{{Session::get('creado')}}</p>
</div>
@endif
```

```
<div class="container-fluid">
<p><a href="/validado/albumes/crear-album" class="btn btn-primary" role="button">Crear Álbum</a></p>
@if (sizeof($albumes)>0)

@foreach ($albumes as $album)
<div class="row">
<div class="col-sm-6 col-md-12">
<div class="thumbnail">
<div class="caption">
<h3>{{$album->nombre}}</h3>

<p>{{$album->descripcion}}</p>
<p><a href="/validado/fotos?id={{$album->id}}" class="btn btn-primary" role="button">Ver</a></p>
</div>
</div>
</div>
</div>

@endforeach
@else
<div class="alert alert-danger">
<p>No tienes álbumes de fotografías. Crea uno.</p>
</div>
@endif
</div>
@endsection
```

15.2 Creación de Fotos

- La gestión de las fotos las realizamos en **FotoController**.
- Tendremos que modificar los métodoso **getCrearFoto()** y **postCrearFoto()**.

- Contenido del fichero **app/Http/Controllers/FotoController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Enfocalia\Http\Requests\MostrarFotosRequest;
use Enfocalia\Http\Requests\CrearFotoRequest;
use Enfocalia\Album;
use Enfocalia\Foto;

use Illuminate\Http\Request;

// Utilidad para las fechas.
use Carbon\Carbon;

class FotoController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex(MostrarFotosRequest $request)
{
//return "Mostrando las fotos del usuario.";
$id = $request->get('id');

// Obtenemos las fotos de ese album con ese id.
$fotos = Album::find($id)->fotos;

// Devolvemos una vista con las fotos de ese álbum.
return view('fotos.mostrar',['fotos'=>$fotos,'id'=>$id]);
}

public function getCrearFoto(Request $request)
{
// Para este método se supone que a la hora de crear una foto necesitamos saber
// a qué album va a asociada, con lo que tenemos que recibir un id.
// En la vista fotos/mostrar.blade.php tenemos que añadir el ID a la URL <a href="/validado/albumes/crear-foto/{{$id}}"
```

```
$id = $request->get('id');

return view('fotos.crear-foto')->withId($id);
}

public function postCrearFoto(CrearFotoRequest $request)
{
//return "Almacenando fotos...";

$id = $request->get('id');

// Recibimos un archivo y generamos un nombre aleatorio en base a la fecha (Carbon) encriptada + la extensión.
$imagen = $request->file('imagen');

// Ruta por defecto en la carpeta public dónde se suben las imágenes.
$ruta='/img/';

// Nombre que le asignamos al fichero.
// Necesario activa extensión en xampp/php/php.ini: extension=php_fileinfo.dll
$nombre = sha1(Carbon::now()).'.'.$imagen->guessExtension();

// Movemos la imagen recibida a la ruta correspondiente.
$imagen->move(getcwd().$ruta,$nombre);

// Creamos la foto.
Foto::create(
[
'nombre'=>$request->get('nombre'),
'descripcion'=>$request->get('descripcion'),
'ruta'=>$ruta.$nombre,
'album_id'=>$id
]);

return redirect("/validado/fotos?id=$id")->with('creada','La foto ha sido subida correctamente.');
}

public function getActualizarFoto()
{
return "Formulario de actualización de fotos.";
}

public function postActualizarFoto()
{
return "Actualizando foto...";
}

public function getEliminarFoto()
{
return "Formulario de eliminación de fotos.";
}

public function postEliminarFoto()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Contenido del fichero **resources/views/fotos/mostrar.blade.php**:

```
@extends('app')

@section('content')

@if (Session::has('creada'))
<div class="alert alert-success">
{{Session::get('creada')}}
</div>
```

```
@endif

<div class="container-fluid">
<p><a href="/validado/fotos/crear-foto?id={{$id}}" class="btn btn-primary" role="button">Crear Foto</a></p>
@if(sizeof($fotos) > 0)
@foreach($fotos as $index => $foto)
@if($index%4 == 0)
<div class="row">
@endif
  <div class="col-sm-6 col-md-3">
    <div class="thumbnail">
        <img src="{{$foto->ruta}}">
      <div class="caption">
        <h3>{{$foto->nombre}}</h3>
        <p>{{$foto->descripcion}}</p>
      </div>
      <p><a href="/validado/fotos/actualizar-foto/{{$foto->id}}" class="btn btn-primary" role="button">Editar Foto</a></p>
      <form action="/validado/fotos/eliminar-foto" method="POST">
<input type="hidden" name="_token" value="{{ csrf_token() }}" required>
<input type="hidden" name="id" value="{{$foto->id}}" required>
<input class="btn btn-danger" role="button" type="submit" value="Eliminar Foto"/>
</form>
    </div>
  </div>
@if(($index+1)%4 == 0)
</div>
@endif
@endforeach
@else
<div class="alert alert-danger">
<p>Al parecer este album no tiene fotos. Crea una.</p>
</div>
@endif
</div>
@endsection
```

- Creamos el request **CrearFotoRequest**:

```
php artisan make:request CrearFotoRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/CrearFotoRequest.php**:

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

use Enfocalia\Http\Requests\Request;
use Illuminate\Support\Facades\Auth;

class CrearFotoRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Copiado de CrearFotoRequest.
$user = Auth::user();

// Id del album recibido
$id = $this->get('id');

// Buscamos si ese usuario tiene un album con ese $id.
$album = $user->albumes()->find($id);

// Si ese album existe devolvemos true, en otro caso false (forbidden).
if ($album)
return true;
else
```

```
return false;
}

/**
         * Get the validation rules that apply to the request.
         *
         * @return array
         */
public function rules()
{
return [
'id' => 'required|exists:albumes',
'nombre'=>'required',
'descripcion'=>'required',
'imagen'=>'required|image|max:20000', // Tipo Imagen y máximo de 20 MB.
];
}

}
```

- Creamos una nueva vista **resources/views/fotos/crear-foto.blade.php**.
- Contenido de la vista **resources/views/fotos/crear-foto.blade.php** - (copiado de fotos/mostrar.blade.php

```
@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() }}">

<div class="form-group 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 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 Imagen
</button>
</div>
</div>
</form>
</div>
@endsection
```

- **IMPORTANTE: Habilitar la extensión fileinfo.dll en Xampp.**
- Para que pueda funcionar correctamente la detección de tipos de archivo:

```
# Editar el fichero \Xampp\php\php.ini
# Buscar fileinfo y descomentarla.
extension=php_fileinfo.dll

# Grabar el fichero.
# Reiniciar Apache.
```

16 Edición de Álbumes y Fotos

16.1 Edición de Álbumes

- Trabajaremos sobre los métodos **getActualizarAlbum()** y **postActualizarAlbum()** del controlador **AlbumController**.
- Contenido del fichero **app/Http/Controllers/AlbumController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Auth;
use Enfocalia\Http\Requests\CrearAlbumRequest;
use Enfocalia\Http\Requests\ActualizarAlbumRequest;
use Enfocalia\Album;

class AlbumController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
//return "Mostrando álbumes del usuario.";

// Obtenemos cual es el usuario que está validado en el sistema.
$usuario =Auth::user();

// Álbumes de ese usuario.
$albumes=$usuario->albumes;

// Devolvemos la vista pasándole todos los álbumes que hemos obtenido.
return view('albumes.mostrar',['albumes'=>$albumes]);
}

public function getCrearAlbum()
{
//return "Formulario de creación Albumes.";
return view('albumes.crear-album');
}

public function postCrearAlbum(CrearAlbumRequest $request)
{
//return "Almacenando Albumes...";

// Obtenemos el usuario conectado.
$usuario = Auth::user();

// Creamos el álbum.
Album::create(
[
'nombre' => $request->get('nombre'),
'descripcion' => $request->get('descripcion'),
'usuario_id' => $usuario->id
]);

// Al hacer el redirect pasando variables esas variables se accederá a ellas a partir de
// las variables de sesión: @if(Session::has('creado'))  y  {{Session::get('creado')}}
return redirect('/validado/albumes')->with('creado','El álbum ha sido creado correctamente.');
}

public function getActualizarAlbum($id)
{
//return "Formulario de actualización de Albumes.";

$album = Album::find($id);

// Llamamos a la vista para actualizar el álbum pasándole el álbum a actualizar
return view('albumes.actualizar-album',['album'=>$album]);
}
```

```
public function postActualizarAlbum(ActualizarAlbumRequest $request)
{
//return "Actualizando Album...";

$album = Album::find($request->get('id'));
$album->nombre= $request->get('nombre');
$album->descripcion=$request->get('descripcion');

$album->save();

return redirect('/validado/albumes')->with('actualizado','El álbum se ha actualizado correctamente.');

}

public function getEliminarAlbum()
{
return "Formulario de eliminación de Albumes.";
}

public function postEliminarAlbum()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

- **Duplicamos** la vista **resources/views/albumes/crear-album.blade.php** y la grabamos con **actualizar-album.blade.php**.
- Contenido del fichero **resources/views/albumes/actualizar-album.blade.php**:

```
@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">Actualizar datos de perfil</div>
<div class="panel-body">
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Whoops!</strong> Hay errores en los campos de entrada.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form class="form-horizontal" role="form" method="POST" action="{{ url('/validado/albumes/actualizar-album') }}">
<input type="hidden" name="_token" value="{{ csrf_token() }}">
<input type="hidden" name="id" value="{{$album->id}}">

<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="{{ $album->nombre }}">
</div>
</div>



<div class="form-group">
<label class="col-md-4 control-label">Descripción</label>
<div class="col-md-6">
<input type="text" class="form-control" name="descripcion" value="{{$album->descripcion}}">
</div>
</div>
```

```
<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">
Actualizar Álbum
</button>
</div>
</div>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

- Le añadimos el **hiperenlace** de **Editar Álbum** a la vista de mostrar.
- Contenido del fichero **resources/views/albumes/mostrar.blade.php**:

```
@extends('app')

@section('content')
@if(Session::has('creado'))
<div class="alert alert-success">
<p>{{Session::get('creado')}}</p>
</div>
@endif

@if(Session::has('actualizado'))
<div class="alert alert-success">
<p>{{Session::get('actualizado')}}</p>
</div>
@endif

<div class="container-fluid">
<p><a href="/validado/albumes/crear-album" class="btn btn-primary" role="button">Crear Álbum</a></p>
@if (sizeof($albumes)>0)

@foreach ($albumes as $index=>$album)
@if ($index %3 ==0)
<div class="row">
@endif
<div class="col-sm-6 col-md-4">
<div class="thumbnail">
<div class="caption">
<h3>{{$album->nombre}}</h3>

<p>{{$album->descripcion}}</p>
<p><a href="/validado/fotos?id={{$album->id}}" class="btn btn-primary" role="button">Ver</a></p>
<p><a href="/validado/albumes/actualizar-album/{{$album->id}}" class="btn btn-primary" role="button">Editar Álbum</a></p>
</div>
</div>
</div>
@if ((($index+1)%3 == 0)
</div>
@endif

@endforeach
@else
<div class="alert alert-danger">
<p>No tienes álbumes de fotografías. Crea uno.</p>
</div>
@endif
</div>
@endsection
```

- Creación del **Request ActualizarAlbumRequest**.

```
php artisan make:request ActualizarAlbumRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/ActualizarAlbumRequest.php**:

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

use Enfocalia\Http\Requests\Request;
use Illuminate\Support\Facades\Auth;

class ActualizarAlbumRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Copiado de CrearFotoRequest.
$user = Auth::user();

// Id del album recibido
$id = $this->get('id');

// Buscamos si ese usuario tiene un album con ese $id.
$album = $user->albumes()->find($id);

// Si ese album existe devolvemos true, en otro caso false (forbidden).
if ($album)
return true;
else
return false;
}

/**
         * Get the validation rules that apply to the request.
         *
         * @return array
         */
public function rules()
{
return [
'id'=>'required|exists:albumes,id',// Debe existir en tabla albumes campo id.
'nombre'=> 'required',
'descripcion' => 'required'
];
}

}
```

16.2 Edición de Fotos

- Contenido del fichero **app/Http/Controllers/FotoController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Enfocalia\Http\Requests\MostrarFotosRequest;
use Enfocalia\Http\Requests\CrearFotoRequest;
use Enfocalia\Http\Requests\ActualizarFotoRequest;

use Enfocalia\Album;
use Enfocalia\Foto;

use Illuminate\Http\Request;

// Utilidad para las fechas.
use Carbon\Carbon;

class FotoController extends Controller {

public function __construct()
{
$this->middleware('auth');
}
```

```
public function getIndex(MostrarFotosRequest $request)
{
//return "Mostrando las fotos del usuario.";
$id = $request->get('id');

// Obtenemos las fotos de ese album con ese id.
$fotos = Album::find($id)->fotos;

// Devolvemos una vista con las fotos de ese álbum.
return view('fotos.mostrar',['fotos'=>$fotos,'id'=>$id]);
}

public function getCrearFoto(Request $request)
{
// Para este método se supone que a la hora de crear una foto necesitamos saber
// a qué album va a asociada, con lo que tenemos que recibir un id.
// En la vista fotos/mostrar.blade.php tenemos que añadir el ID a la URL <a href="/validado/albumes/crear-foto/{{$id}}"
$id = $request->get('id');

return view('fotos.crear-foto')->withId($id);
}

public function postCrearFoto(CrearFotoRequest $request)
{
//return "Almacenando fotos...";

$id = $request->get('id');

// Recibimos un archivo y generamos un nombre aleatorio en base a la fecha (Carbon) encriptada + la extensión.
$imagen = $request->file('imagen');

// Ruta por defecto en la carpeta public dónde se suben las imágenes.
$ruta='/img/';

// Nombre que le asignamos al fichero.
// Necesario activa extensión en xampp/php/php.ini: extension=php_fileinfo.dll
$nombre = sha1(Carbon::now()).'.'.$imagen->guessExtension();

// Movemos la imagen recibida a la ruta correspondiente.
$imagen->move(getcwd().$ruta,$nombre);

// Creamos la foto.
Foto::create(
[
'nombre'=>$request->get('nombre'),
'descripcion'=>$request->get('descripcion'),
'ruta'=>$ruta.$nombre,
'album_id'=>$id
]);

return redirect("/validado/fotos?id=$id")->with('creada','La foto ha sido subida correctamente.');
}

public function getActualizarFoto($id)
{
//return "Formulario de actualización de fotos.";
$foto = Foto::find($id);

return view('fotos.actualizar-foto')->with('foto',$foto);
}

public function postActualizarFoto(ActualizarFotoRequest $request)
{
//return "Actualizando foto...";
// Buscamos la foto por el id.
$foto = Foto::find($request->get('id'));

// Actualizamos sus datos.
$foto->nombre=$request->get('nombre');
$foto->descripcion=$request->get('descripcion');

// Comprobamos si recibimos una imagen.
```

```
if ($request->hasFile('imagen'))
{
// Recibimos un archivo y generamos un nombre aleatorio en base a la fecha (Carbon) encriptada + la extensión.
$imagen = $request->file('imagen');

// Creamos el archivo.
// Ruta por defecto en la carpeta public dónde se suben las imágenes.
$ruta='/img/';

// Nombre que le asignamos al fichero.
// Necesario activa extensión en xampp/php/php.ini: extension=php_fileinfo.dll
$nombre = sha1(Carbon::now()).'.'.$imagen->guessExtension();

// Movemos la imagen recibida a la ruta correspondiente.
$imagen->move(getcwd().$ruta,$nombre);

// Borramos la imagen antigua.
$rutaAnterior = getcwd().$foto->ruta;

if (file_exists($rutaAnterior))
{
unlink (realpath($rutaAnterior));
}

// Actualizamos la ruta a la nueva ruta de la nueva foto.
$foto->ruta = $ruta.$nombre;
}


// Grabamos el registro.
$foto->save();

return redirect("validado/fotos?id=$foto->album_id")->with('editada','La foto fue editada correctamente.');

}

public function getEliminarFoto()
{
return "Formulario de eliminación de fotos.";
}

public function postEliminarFoto()
{
return "Eliminando foto...";
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- **Duplicamos resources/views/fotos/crear-foto.blade.php** con el nombre **resources/views/fotos/actualizar-foto.blade.php**
- Contenido del fichero **resources/views/fotos/actualizar-foto.blade.php**:

```
@extends('app')

@section('content')
<div class="container-fluid">
<form class="form-horizontal" role="form" method="POST" action="/validado/fotos/actualizar-foto" enctype="multipart/form-data">
<input type="hidden" name="_token" value="{{ csrf_token() }}" required>
<input type="hidden" name="id" value="{{ $foto->id }}" required>

<div class="form-group required">
<label class="col-md-4 control-label">Nombre</label>
<div class="col-md-6">
<input type="text" class="form-control" name="nombre" value="{{ $foto->nombre }}" required>
</div>
</div>


<div class="form-group required">
```

```
<label class="col-md-4 control-label">Descripción</label>
<div class="col-md-6">
<textarea class="form-control" name="descripcion" rows="3" required>{{$foto->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">
</div>
</div>

<div class="form-group">
<div class="col-md-6 col-md-offset-4">
<button type="submit" class="btn btn-primary">
Actualizar Imagen
</button>
</div>
</div>
</form>
</div>
@endsection
```

- Creación del **Request ActualizarFotoRequest**.

```
php artisan make:request ActualizarFotoRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/ActualizarFotoRequest.php**:

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

use Enfocalia\Http\Requests\Request;
use Illuminate\Support\Facades\Auth;

use Enfocalia\Album;
use Enfocalia\Foto;

class ActualizarFotoRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Copiado de CrearFotoRequest.
$user = Auth::user();

// Id de la foto
$id = $this->get('id');

// Buscamos si ese usuario tiene una foto con ese id.
$foto = Foto::find($id);

// Verificamos si existe un album del usuario que coincida con el album_id de la foto.
$album = $user->albumes()->find($foto->album_id);

// Si esa foto existe devolvemos true, en otro caso false (forbidden).
if ($album)
return true;
else
return false;
}

/**
         * Get the validation rules that apply to the request.
         *
         * @return array
         */
```

```
public function rules()
{
return [
'id'=>'required|exists:fotos,id',
'nombre'=>'required',
'descripcion'=>'required',
'imagen'=> 'image|max:20000'  // No es requerido pero si lo recibimos tendrá q cumplir esas condiciones.
];
}

}
```

- Añadimos a la vista de **fotos/mostrar.blade.php** el mensaje de **Editada correctamente**.
- Contenido del fichero **resources/views/fotos/mostrar.blade.php**:

```
@extends('app')

@section('content')

@if (Session::has('creada'))
<div class="alert alert-success">
{{Session::get('creada')}}
</div>
@endif

@if (Session::has('editada'))
<div class="alert alert-success">
{{Session::get('editada')}}
</div>
@endif

<div class="container-fluid">
<p><a href="/validado/fotos/crear-foto?id={{$id}}" class="btn btn-primary" role="button">Crear Foto</a></p>
@if(sizeof($fotos) > 0)
@foreach($fotos as $index => $foto)
@if($index%4 == 0)
<div class="row">
@endif
  <div class="col-sm-6 col-md-3">
    <div class="thumbnail">
        <img src="{{$foto->ruta}}">
      <div class="caption">
        <h3>{{$foto->nombre}}</h3>
        <p>{{$foto->descripcion}}</p>
      </div>
      <p><a href="/validado/fotos/actualizar-foto/{{$foto->id}}" class="btn btn-primary" role="button">Editar Foto</a></p>
      <form action="/validado/fotos/eliminar-foto" method="POST">
<input type="hidden" name="_token" value="{{ csrf_token() }}" required>
<input type="hidden" name="id" value="{{$foto->id}}" required>
<input class="btn btn-danger" role="button" type="submit" value="Eliminar Foto"/>
</form>
    </div>
  </div>
@if(($index+1)%4 == 0)
</div>
@endif
@endforeach
@else
<div class="alert alert-danger">
<p>Al parecer este album no tiene fotos. Crea una.</p>
</div>
@endif
</div>
@endsection
```

17 Borrado de Álbumes y Fotos

17.1 Borrado de Álbumes

- Solamente necesitamos el método **postEliminarAlbum()**, así que **eliminamos getEliminarAlbum()**.
- Contenido del fichero **app/Http/Controllers/AlbumController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Auth;
use Enfocalia\Http\Requests\CrearAlbumRequest;
use Enfocalia\Http\Requests\ActualizarAlbumRequest;
use Enfocalia\Http\Requests\EliminarAlbumRequest;
use Enfocalia\Album;


class AlbumController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex()
{
//return "Mostrando álbumes del usuario.";

// Obtenemos cual es el usuario que está validado en el sistema.
$usuario =Auth::user();

// Álbumes de ese usuario.
$albumes=$usuario->albumes;

// Devolvemos la vista pasándole todos los álbumes que hemos obtenido.
return view('albumes.mostrar',['albumes'=>$albumes]);
}

public function getCrearAlbum()
{
//return "Formulario de creación Albumes.";
return view('albumes.crear-album');
}

public function postCrearAlbum(CrearAlbumRequest $request)
{
//return "Almacenando Albumes...";

// Obtenemos el usuario conectado.
$usuario = Auth::user();

// Creamos el álbum.
Album::create(
[
'nombre' => $request->get('nombre'),
'descripcion' => $request->get('descripcion'),
'usuario_id' => $usuario->id
]);

// Al hacer el redirect pasando variables esas variables se accederá a ellas a partir de
// las variables de sesión: @if(Session::has('creado'))  y  {{Session::get('creado')}}
return redirect('/validado/albumes')->with('creado','El álbum ha sido creado correctamente.');
}

public function getActualizarAlbum($id)
{
//return "Formulario de actualización de Albumes.";

$album = Album::find($id);

// Llamamos a la vista para actualizar el álbum pasándole el álbum a actualizar
return view('albumes.actualizar-album',['album'=>$album]);
}

public function postActualizarAlbum(ActualizarAlbumRequest $request)
```

```
{
//return "Actualizando Album...";

$album = Album::find($request->get('id'));
$album->nombre= $request->get('nombre');
$album->descripcion=$request->get('descripcion');

$album->save();

return redirect('/validado/albumes')->with('actualizado','El álbum se ha actualizado correctamente.');

}

public function postEliminarAlbum(EliminarAlbumRequest $request)
{
//return "Eliminando album...";

$album = Album::find($request->get('id'));

// Obtenemos todas las fotos de ese album.
$fotos = $album->fotos;

// Recorremos todas las fotos y las borramos físicamente.
foreach ($fotos as $foto)
{
$ruta=getcwd().$foto->ruta;

if (file_exists($ruta))
{
unlink (realpath($ruta));
}

$foto->delete();
}

// Borramos el álbum ya que no tiene fotos.
$album->delete();

return redirect('/validado/albumes')->with('eliminado','El álbum fue eliminado correctamente.');
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}

}
```

- Creación del **Request EliminarAlbumRequest**.

```
php artisan make:request EliminarAlbumRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/EliminarAlbumRequest.php**:

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

use Enfocalia\Http\Requests\Request;
use Illuminate\Support\Facades\Auth;

class EliminarAlbumRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Copiado de ActualizarAlbumRequest.
$user = Auth::user();
```

```
// Id del album recibido
$id = $this->get('id');

// Buscamos si ese usuario tiene un album con ese $id.
$album = $user->albumes()->find($id);

// Si ese album existe devolvemos true, en otro caso false (forbidden).
if ($album)
return true;
else
return false;
}

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

}
```

- Nos falta incluir el **formulario de eliminar el Álbum** y el mensaje de **Eliminado correctamente**.
- Contenido del fichero **resources/views/albumes/mostrar.blade.php**:

```
@extends('app')

@section('content')
@if(Session::has('creado'))
<div class="alert alert-success">
<p>{{Session::get('creado')}}</p>
</div>
@endif

@if(Session::has('actualizado'))
<div class="alert alert-success">
<p>{{Session::get('actualizado')}}</p>
</div>
@endif

@if(Session::has('eliminado'))
<div class="alert alert-danger">
<p>{{Session::get('eliminado')}}</p>
</div>
@endif

<div class="container-fluid">
<p><a href="/validado/albumes/crear-album" class="btn btn-primary" role="button">Crear Álbum</a></p>
@if (sizeof($albumes)>0)

@foreach ($albumes as $index=>$album)
@if ($index %3 ==0)
<div class="row">
@endif
<div class="col-sm-6 col-md-4">
<div class="thumbnail">
<div class="caption">
<h3>{{$album->nombre}}</h3>

<p>{{$album->descripcion}}</p>
<p><a href="/validado/fotos?id={{$album->id}}" class="btn btn-primary" role="button">Ver</a></p>
<p><a href="/validado/albumes/actualizar-album/{{$album->id}}" class="btn btn-primary" role="button">Editar Álbum</a></p>

<form action='/validado/albumes/eliminar-album' method="POST">
<input type="hidden" name="_token" value="{{csrf_token()}}" required/>
<input type="hidden" name="id" value="{{$album->id}}" required/>
<input class="btn btn-danger" role="button" type="submit" value="Eliminar Álbum"/>
```

```
</form>

</div>
</div>
</div>
@if (($index+1)%3 == 0)
</div>
@endif

@endforeach
@else
<div class="alert alert-danger">
<p>No tienes álbumes de fotografías. Crea uno.</p>
</div>
@endif
</div>
@endsection
```

17.2 Borrado de Fotos

- Solamente necesitamos el método **postEliminarFoto()**, así que **eliminamos getEliminarFoto()**.
- Contenido del fichero **app/Http/Controllers/FotoController.php**:

```
<?php namespace Enfocalia\Http\Controllers;

use Enfocalia\Http\Requests\MostrarFotosRequest;
use Enfocalia\Http\Requests\CrearFotoRequest;
use Enfocalia\Http\Requests\ActualizarFotoRequest;
use Enfocalia\Http\Requests\EliminarFotoRequest;

use Enfocalia\Album;
use Enfocalia\Foto;

use Illuminate\Http\Request;

// Utilidad para las fechas.
use Carbon\Carbon;

class FotoController extends Controller {

public function __construct()
{
$this->middleware('auth');
}

public function getIndex(MostrarFotosRequest $request)
{
//return "Mostrando las fotos del usuario.";
$id = $request->get('id');

// Obtenemos las fotos de ese album con ese id.
$fotos = Album::find($id)->fotos;

// Devolvemos una vista con las fotos de ese álbum.
return view('fotos.mostrar',['fotos'=>$fotos,'id'=>$id]);
}

public function getCrearFoto(Request $request)
{
// Para este método se supone que a la hora de crear una foto necesitamos saber
// a qué album va a asociada, con lo que tenemos que recibir un id.
// En la vista fotos/mostrar.blade.php tenemos que añadir el ID a la URL <a href="/validado/albumes/crear-foto/{{$id}}"
$id = $request->get('id');

return view('fotos.crear-foto')->withId($id);
}

public function postCrearFoto(CrearFotoRequest $request)
{
//return "Almacenando fotos...";

$id = $request->get('id');
```

```
// Recibimos un archivo y generamos un nombre aleatorio en base a la fecha (Carbon) encriptada + la extensión.
$imagen = $request->file('imagen');

// Ruta por defecto en la carpeta public dónde se suben las imágenes.
$ruta='/img/';

// Nombre que le asignamos al fichero.
// Necesario activa extensión en xampp/php/php.ini: extension=php_fileinfo.dll
$nombre = sha1(Carbon::now()).'.'.$imagen->guessExtension();

// Movemos la imagen recibida a la ruta correspondiente.
$imagen->move(getcwd().$ruta,$nombre);

// Creamos la foto.
Foto::create(
[
'nombre'=>$request->get('nombre'),
'descripcion'=>$request->get('descripcion'),
'ruta'=>$ruta.$nombre,
'album_id'=>$id
]);

return redirect("/validado/fotos?id=$id")->with('creada','La foto ha sido subida correctamente.');
}

public function getActualizarFoto($id)
{
//return "Formulario de actualización de fotos.";
$foto = Foto::find($id);

return view('fotos.actualizar-foto')->with('foto',$foto);
}

public function postActualizarFoto(ActualizarFotoRequest $request)
{
//return "Actualizando foto...";
// Buscamos la foto por el id.
$foto = Foto::find($request->get('id'));

// Actualizamos sus datos.
$foto->nombre=$request->get('nombre');
$foto->descripcion=$request->get('descripcion');

// Comprobamos si recibimos una imagen.
if ($request->hasFile('imagen'))
{
// Recibimos un archivo y generamos un nombre aleatorio en base a la fecha (Carbon) encriptada + la extensión.
$imagen = $request->file('imagen');

// Creamos el archivo.
// Ruta por defecto en la carpeta public dónde se suben las imágenes.
$ruta='/img/';

// Nombre que le asignamos al fichero.
// Necesario activa extensión en xampp/php/php.ini: extension=php_fileinfo.dll
$nombre = sha1(Carbon::now()).'.'.$imagen->guessExtension();

// Movemos la imagen recibida a la ruta correspondiente.
$imagen->move(getcwd().$ruta,$nombre);

// Borramos la imagen antigua.
$rutaAnterior = getcwd().$foto->ruta;

if (file_exists($rutaAnterior))
{
unlink (realpath($rutaAnterior));
}

// Actualizamos la ruta a la nueva ruta de la nueva foto.
$foto->ruta = $ruta.$nombre;
}
```

```
// Grabamos el registro.
$foto->save();

return redirect("validado/fotos?id=$foto->album_id")->with('editada','La foto fue editada correctamente.');

}

public function postEliminarFoto(EliminarFotoRequest $request)
{
//return "Eliminando foto...";

// Obtenemos la foto.

$foto=Foto::find($request->get('id'));

$rutaAnterior = getcwd().$foto->ruta;

if (file_exists($rutaAnterior))
{
unlink (realpath($rutaAnterior));
}

$foto->delete();

return redirect("validado/fotos?id=$foto->album_id")->with('eliminada','La foto se ha eliminado correctamente.');
}

public function missingMethod($parameters = array())
{
// Disparamos un error 404.
abort(404);
}
}
```

- Creación del **Request EliminarFotoRequest**.

```
php artisan make:request EliminarFotoRequest
#Request created successfully.
```

- Contenido del fichero **app/Http/Requests/EliminarFotoRequest.php**:

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

use Enfocalia\Http\Requests\Request;
use Illuminate\Support\Facades\Auth;

use Enfocalia\Album;
use Enfocalia\Foto;

class EliminarFotoRequest extends Request {

/**
         * Determine if the user is authorized to make this request.
         *
         * @return bool
         */
public function authorize()
{
// Copiado de ActualizarFotoRequest.
$user = Auth::user();

// Id de la foto
$id = $this->get('id');

// Buscamos si ese usuario tiene una foto con ese id.
$foto = Foto::find($id);

// Verificamos si existe un album del usuario que coincida con el album_id de la foto.
$album = $user->albumes()->find($foto->album_id);

// Si esa foto existe devolvemos true, en otro caso false (forbidden).
```

```
if ($album)
return true;
else
return false;
}

/**
         * Get the validation rules that apply to the request.
         *
         * @return array
         */
public function rules()
{
return [
'id'=>'required|exists:fotos,id'
];
}
}
```

- Nos falta incluir el **formulario de eliminar el Álbum** y el mensaje de **Eliminado correctamente**.
- Contenido del fichero **resources/views/albumes/mostrar.blade.php**:

```
@extends('app')

@section('content')

@if (Session::has('creada'))
<div class="alert alert-success">
{{Session::get('creada')}}
</div>
@endif

@if (Session::has('editada'))
<div class="alert alert-success">
{{Session::get('editada')}}
</div>
@endif

@if (Session::has('eliminada'))
<div class="alert alert-success">
{{Session::get('editada')}}
</div>
@endif

<div class="container-fluid">
<p><a href="/validado/fotos/crear-foto?id={{$id}}" class="btn btn-primary" role="button">Crear Foto</a></p>
@if(sizeof($fotos) > 0)
@foreach($fotos as $index => $foto)
@if($index%4 == 0)
<div class="row">
@endif
  <div class="col-sm-6 col-md-3">
    <div class="thumbnail">
        <img src="{{$foto->ruta}}">
      <div class="caption">
        <h3>{{$foto->nombre}}</h3>
        <p>{{$foto->descripcion}}</p>
      </div>
      <p><a href="/validado/fotos/actualizar-foto/{{$foto->id}}" class="btn btn-primary" role="button">Editar Foto</a></p>
      <form action="/validado/fotos/eliminar-foto" method="POST">
<input type="hidden" name="_token" value="{{ csrf_token() }}" required>
<input type="hidden" name="id" value="{{$foto->id}}" required>
<input class="btn btn-danger" role="button" type="submit" value="Eliminar Foto"/>
</form>
    </div>
  </div>
@if(($index+1)%4 == 0)
</div>
@endif
@endforeach
@else
<div class="alert alert-danger">
<p>Al parecer este album no tiene fotos. Crea una.</p>
```

```
</div>
@endif
</div>
@endsection
```

18 Creación de Vistas para mensajes de Error

18.1 Error 404

- Cuando se produce un error 404 o lo disparamos con abort(404) se intenta mostrar el contenido de resources/views/errors/404.blade.php
- Podemos copiar el que viene de ejemplo 503.blade.php, editarlo y grabarlo como 404.blade.php

```
<html>
<head>
<link href='//fonts.googleapis.com/css?family=Lato:100' rel='stylesheet' type='text/css'>

<style>
body {
margin: 0;
padding: 0;
width: 100%;
height: 100%;
color: #B0BEC5;
display: table;
font-weight: 100;
font-family: 'Lato';
}

.container {
text-align: center;
display: table-cell;
vertical-align: middle;
}

.content {
text-align: center;
display: inline-block;
}

.title {
font-size: 72px;
margin-bottom: 40px;
}
</style>
</head>
<body>
<div class="container">
<div class="content">
<div class="title">La página que usted busca no se ha encontrado.</div>
</div>
</div>
</body>
</html>
```

19 Idiomas en Laravel 5

- Documentación Oficial sobre Localización en Laravel 5
- Con Laravel 5 podemos definir los mensajes de error en nuestro propio idioma.
- Podemos crear también nuestros propios ficheros de idioma.
- En la carpeta /**resources**/**lang**/**en** se encuentran ejemplos de ficheros de idioma en **Inglés**.
- Si quisiéramos ponerlos en español haríamos lo siguiente.
- Duplicar la carpeta **resources**/**lang**/**en** con el nombre de **resources**/**lang**/**es**

- **Traducir** el contenido de los ficheros.
- Indicarle a Laravel en el fichero **config**/**app.php** que use español:

```
'locale' => 'es',
```

- También **se podría modificar en tiempo real el idioma utilizado** con la instrucción:

```
App::setLocale('es');
```

- Para mostrar algún mensaje del fichero de idiomas se haría:

```
// echo trans('fichero.campo');
// Ejemplo:
echo trans('validation.required');

// O bien

echo Lang::get('validation.required');
```

- Ejemplo de un fichero de idioma **resources/lang/en/validation.php**:

```
<?php

return [

/*
        |--------------------------------------------------------------------------
        | Validation Language Lines
        |--------------------------------------------------------------------------
        |
        | The following language lines contain the default error messages used by
        | the validator class. Some of these rules have multiple versions such
        | as the size rules. Feel free to tweak each of these messages here.
        |
        */

"accepted"             => "The :attribute must be accepted.",
"active_url"           => "The :attribute is not a valid URL.",
"after"                => "The :attribute must be a date after :date.",
.....
```

20 Código fuente de la aplicación

Archivo:Album.zip

--Veiga (discusión) 23:15 3 may 2015 (CEST)