SISTEMA DE SUBIDA Y MUESTRA DE IMÁGENES

En esta actividad vamos a implementar la funcionalidad de subida y listado de imágenes de nuestra red social. Dentro de esta actividad se engloban los siguientes apartados:

- Formulario de subida
- Subida y almacenamiento
- Listado de imágenes
- Paginación
- Mostar fecha de subida

Se van a proporcionar una serie de orientaciones, a partir de las cuales debes ir desarrollando y adaptando tu proyecto a las funcionalidades que se requieren.

En primer lugar debemos saber que todos los archivos que subamos se almacenarán en discos dentro del directorio storage.

Dentro del directorio config, en el archivo filesystems.php, encontramos todos los discos creados por defecto, donde se irán almacenando los archivos que subamos.

```php
<?php

return [

    /*
    |--------------------------------------------------------------------------
    | Default Filesystem Disk
    |--------------------------------------------------------------------------
    |
    | Here you may specify the default filesystem disk that should be used
    | by the framework. The "local" disk, as well as a variety of cloud
    | based disks are available to your application. Just store away!
    |
    */

    'default' => env( key: 'FILESYSTEM_DISK', default: 'local'),

    /*
    |--------------------------------------------------------------------------
    | Filesystem Disks
    |--------------------------------------------------------------------------
```

El disco 'local' es el que está por defecto, pero existen otros como por ejemplo 'public'.

Después modificaremos este archivo, pero el primer caso es generar el controlador con la consola de artisan. Debe contener al menos 3 métodos, uno para mostrar el formulario de subida, otro para almacenar las imágenes y el último para mostrarlas. Además, crearemos un constructor con el middleware de autenticación para poder asegurarnos de que se ha hecho login para poder acceder a la dirección donde se mostrarán las imágenes.

```
11  class ImageController extends Controller
12  {
13
14      public function __construct(){
15          $this->middleware( middleware: 'auth');
16      }
```

Recuerda cargar en el controlador la clase Storage y File que se encargan de la subida y manejo de archivos.

Dentro de la Storage tenemos el método put(), que recibe como primer parámetro el nombre del archivo y como segundo el contenido, que lo recogeremos mediante el método get() de la clase File.

Como hemos dicho anteriormente, por defecto se trabaja con el disco public, que si observamos almacena los archivos que subamos dentro de storage/app:

```
31      'disks' => [
32
33          'local' => [
34              'driver' => 'local',
35              'root' => storage_path( path: 'app'),
36              'throw' => false,
37          ],
```

Tenemos dos opciones, o bien cambiar el disco de almacenamiento por defecto en este archivo, o bien antes de llamar al método put():

```
Storage::disk( name: 'images')->put($image_path_name, File::get($image_path) );
```

Creamos el disco ‘images’, donde se almacenarán todas las imágenes de nuestra red social:

```
47        'images' => [
48            'driver' => 'local',
49            'root' => storage_path( path: 'app/images'),
50            'url' => env( key: 'APP_URL').'/images',
51            'visibility' => 'public',
52            'throw' => false,
53        ],
```

Para completar la implementación, crea el formulario y recoge el nombre del archivo y el propio archivo para almacenarlo en el disco ‘images’. Además, debes almacenar la información necesaria en la tabla ‘images’.

Sin embargos, para el almacenamiento de imágenes tenemos otra opción. Puedes emplear el método store() o storeAs(), si deseas especificar el nombre del archivo, en lugar de lo especificado anteriormente.

Recuerda especificar el nombre del disco donde se almacenarán los archivos, de lo contrario se almacenaran en el definido por defecto.

```
29        $file = $request->file( key: 'image');
30        $file->StoreAs();
```

Revisa la documentación oficial para ver algún ejemplo más.

Por último, cabe destacar que en el campo user_id de la tabla ‘images’, debemos almacenar el id del usuario que ha subido la imagen. Para ello lo rescataremos gracias a la clase Auth, ya que para subir una imagen se ha debido hacer log in.

```php
use Illuminate\Support\Facades\Auth;

// Retrieve the currently authenticated user...
$user = Auth::user();

// Retrieve the currently authenticated user's ID...
$id = Auth::id();
```

LISTADO DE IMÁGENES

Ahora pasaremos a crear el método que nos permita mostrar los archivos que hay dentro de un determinado disco.

Para listar estas imágenes, vamos a ayudarnos de la clase Storage, que devolverá un array.

```php
foreach (Storage::disk( name: 'images')->files() as $file){
    var_dump($file);
```

Esto nos devuelve el nombre del archivo que hemos subido, pero con el método url() de la clase Storage obtenemos la ruta de la imagen:

```php
67      foreach (Storage::disk( name: 'images')->files() as $file){
68          $url= Storage::url($file);
69          dump($url);
```

```
"/storage/1674203131database_tip.PNG" // app\Http\Controllers\ImageController.php:69
```

Sin embargo, necesitamos la url completa y para generarla vamos a emplear el helper asset(), que generará una url a nuestro asset.

```php
$url = asset(Storage::disk( name: 'images')->url($file));
dump($url);
```

```
"http://localhost/storage/1674203131database_tip.PNG" // app\Http\Controllers\ImageController.php:72
```

Vemos que, si queremos mostrar la imagen con la anterior ruta, aparecería lo siguiente:

Esto es debido a que la ruta con la que está cargando la imagen no es accesible.

Para ello necesitamos configurar que los archivos que hay dentro de 'storage' sean accesibles. Para ello hacemos una conexión entre el directorio public y el directorio storage, donde hemos almacenado las imágenes.

```
PS C:\proyecto_rrss> php artisan storage:link

   INFO  The [C:\proyecto_rrss\public\storage] link has been connected to [C:\proyecto_rrss\storage\app/public].
```

Fíjate que únicamente el disco que hemos 'linkado' es el de public, pero nosotros estamos trabajando con el disco 'images'. Así pues, debemos generar también ese link, por lo que añadimos dentro del archivo filesystems.php lo siguiente:

```
80      'links' => [
81          public_path( path: 'storage') => storage_path( path: 'app/public'),
82          public_path( path: 'images') => storage_path( path: 'app/images'),
83
84      ],
```

LISTAR IMÁGENES DESDE LA BASE DE DATOS

Ahora vamos a listar las imágenes desde la base de datos, que será una tarea simple ya que el Eloquent facilita esta tarea en gran medida. Para ello, creamos un controlador Home, con el constructor que verifica si el usuario está logueado y con un método index() en el que listaremos todas las imágenes.

```php
public function index(){
    $images = Image::orderBy('id','desc')->get();

    return view( view: 'dashboard',[
        'images' => $images
        ]);
}
```

Por último, desde la vista, podemos ayudarnos del helper ‘asset’ para generar la url que permita mostrar las imágenes:

```php
@foreach($images as $image)
    {{$image->user->name.' '.$image->user->surname}}
    {{$image->description}} <br>
    <img src="{{asset('images/'.$image->image_path)}}"/>
@endforeach
```

PAGINACIÓN

Para poder hacer una paginación en Laravel, únicamente debemos sustituir el get() de obtener las imágenes por el método paginate(), pasando como parámetro el número de elementos por página. Después, en la vista, únicamente debemos mostrar los links, con el método links().

Revisa la documentación de Laravel para ver los diferentes tipos de paginación que puedes agregar.

MOSTRAR FECHA DE SUBIDA

Junto con la imagen aparecerá el tiempo que hace que se subió la imagen, en el formato “hace …”. Para ello nos vamos a ayudar de la librería Carbon. Consulta la documentación para mostrar la fecha en el formato adecuado.

Las función clave será la de longAbsoluteDiffForHumans().