

## 3º Proyecto : Instagram laravel

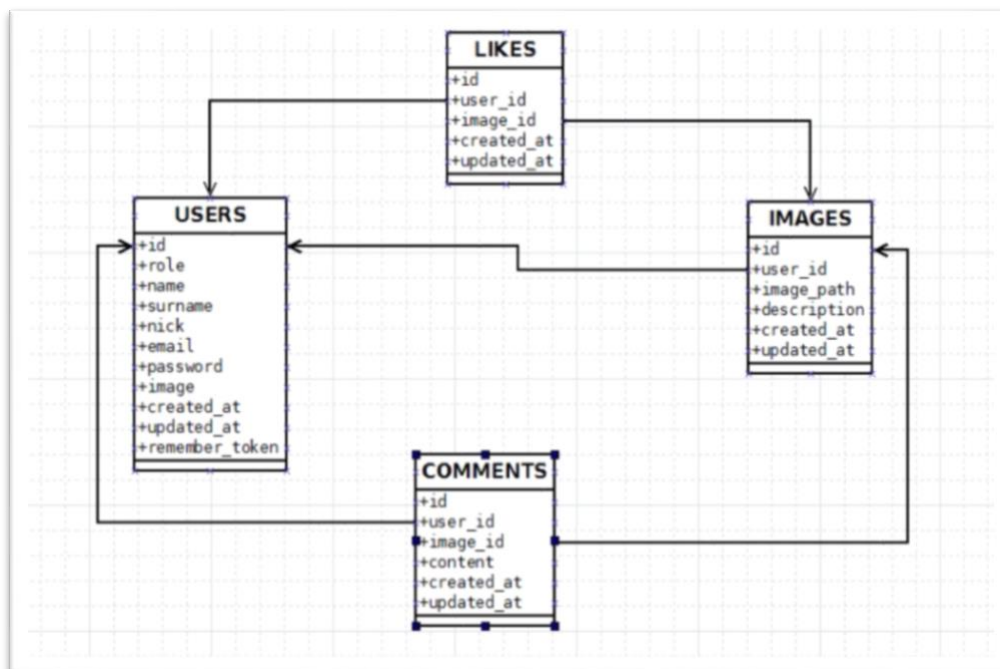
En este proyecto vamos a implementar un Instagram utilizando el framework de Laravel

Los pasos para seguir son los siguientes

1. Diseño de la Base Datos y crear proyecto
2. Migraciones, modelos y seeders
3. Relaciones del modelo
4. Login y registro de usuario
5. Configuración de usuarios
6. Imágenes de la aplicación
7. Sistema de comentarios
8. Sistema de likes
9. Perfiles de usuarios
10. Edición y borrado de imágenes
11. Gente y buscador

### 1. Diseñar la Base de datos y crear proyecto

- Creamos nuestro proyecto en Laravel utilizando composer
  - Creamos nuestra BD **instagram**
  - Realizamos la conexión con la BD .env
- Nota:** Si has utilizado host virtual para acceder al proyecto modifica la línea APP\_URL=tu\_dominio
- Esquema de la BD



## 2. Migraciones, modelo y seeder

- Utilizando el fichero de SQL comprueba los tipos de campo y realiza 3 migraciones para cada una de las tablas en el siguiente orden (**images, comments, likes**) para que no haya problemas con las claves foráneas. La migración Users ya esta creada modifícala con los campos que indique el SQL e investiga como se añade al campo **image** que pueda ser null.

**Nota:** Otra forma de resolverlo es crear una migración extra solamente con las claves foráneas una vez creadas las tablas con las migraciones.

- Crea los 3 modelos en singular (**Image, Comment, Like**). El **modelo User** ya está creado.

Ejemplo tabla images

```
Schema::create('images', function (Blueprint $table) {  
    $table->increments('id');  
    $table->Integer('user_id')->unsigned();  
    $table->string('image_path',255);  
    $table->text('description');  
    $table->foreign('user_id')->references('id')->on('users');  
    $table->timestamps();  
});
```

- Asegúrate que todas las claves primarias de las 4 tablas sea **increments**.
- Las claves foráneas sean **Integer sin tamaño y unsigned()**.

- Realiza 1 seeder para insertar los datos en todas las tablas siguiendo los insert del SQL. Utiliza Eloquent ORM para insertar los datos insertados previamente en un array. Importa la librería de todos los modelo

## 3. Relaciones del modelo

Vamos a crear dentro de los modelos unos métodos para que nos resulte más fácil acceder a las tablas que están relacionadas 1:M o M:1.

Ejemplo: Para el **modelo Image** tenemos que crear un Atributo con el nombre de la tabla

```
protected $table='images';
```

y 3 métodos

- 1:M con la tabla Comments: Con hasMany podemos obtener todos los comentarios de una imagen (método se pone en plural **comments()**)
- 1:M con la tabla Likes
- M:1 con la tabla Users: Con belongsTo podemos obtener los datos del usuario de esa imagen (método se pone en singular **user()**)

```

<?php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Image extends Model
{
    protected $table = 'images';

    // Relación One To Many / de uno a muchos
    public function comments(){
        return $this->hasMany('App\Comment')->orderBy('id', 'desc');
    }

    // Relación de Muchos a Uno
    public function user(){
        return $this->belongsTo('App\User', 'user_id');
    }
}

```

Completa el método que falta en este modelo y completa métodos restantes.

#### 4. Login y registro de usuario

- Creamos con artisan toda la infraestructura de autenticación de usuario
- En el fichero de web.php se han creado automáticamente 2 líneas para generar todas las rutas necesarias. Modifica para que la ruta por defecto(/) llame al método index de HomeController (que es el formulario para identificarse). Elimina la ruta que te redirige a Welcome.

**Nota:** Para cambiar el idioma de los formularios puedes copiar los ficheros de **lang/en** a **lang/es** y cambiar los nombres. Configúralo para que utilicen esa carpeta.

```

// GENERALES
Auth::routes();
Route::get('/', 'HomeController@index')->name('home');

```

- Como hemos eliminado la ruta /home dentro del controlador RegisterController modifica la redirección

```
protected $redirectTo = '/';
```

- Más adelante también tendrás que modificar el método **HomeController@index** para que redirija a la vista principal cuando esté creada.
- **Modificar el formulario de registro (Views/Auth/register.blade.php)** como indica la imagen. Esto conlleva modificar también nuestro modelo

Register

Name

Surname

Nick

E-Mail Address

Password

Confirm Password

Una vez modificado la vista del formulario, modifica nuestro modelo **user.php y añadir los campos rellenables**

```
protected $fillable = [
    'role', 'name', 'surname', 'nick', 'email', 'password',
];
```

Y después modificar el método **validador en RegisterController@validator para validar** los campos que hemos incluidos nuevos y para terminar y que se almacenen en la base de datos modificamos el método **RegisterController@create** donde añadiremos los campos nuevos del formulario y **el campo Role tendrá siempre el valor user**

```
'role' => 'user',
```

- Nuestro **layout master** que heredarán todas las vistas es **layouts/app.blade.php** que se ha instalado automáticamente con el control de usuario. Vamos a incluir más elementos que utilizarán los **mismo estilos bootstrap** de las otras opciones incluidas en el fichero. Identifica las opciones que aparece como invitados (@guest) y las de logueado (else). Una vez dentro del else que indica que está logueado
  - Va a tener las siguientes opciones (**Inicio, gente, favorita y subir imagen**) -> **class="nav-item"**.
  - Y En el desplegable que parece el logout, introduce encima (**Mi perfil y Configuración**)-> **class= "dropdown-item"**

Inicio Gente Favoritas Subir imagen antonio ▾

Mi perfil

Configuración

Logout

- Modifica el estilos a tu gusto añadiendo header y footer en **app.blade.php**

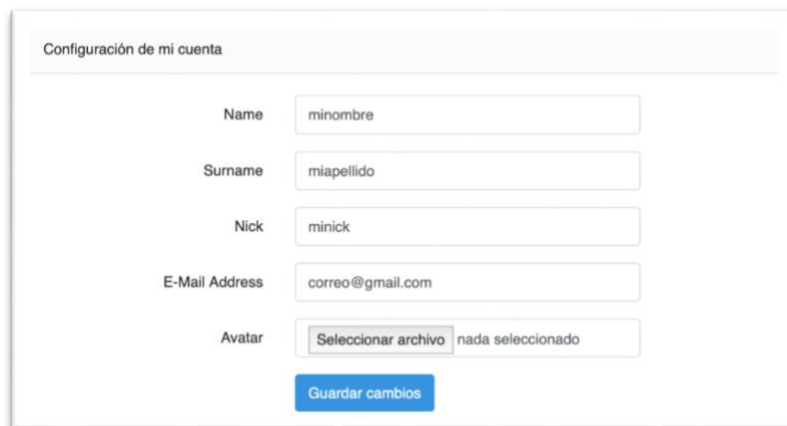
En los próximos apartados iremos creando las rutas de cada una de esas opciones. Vamos a intentar que todas las rutas tengan su **nombre**. **Ejemplo:**

**Route:: (.....)->name (user.create) y la llamada route ('user.create')**

Recuerda crear también las 4 carpetas para las vistas de cada una de las tablas

## 5. Configuración de usuarios

La opción de configuración de usuario nos permite modificar los datos de usuario y añadir un avatar. Para realizar la vista del formulario copiamos el código del formulario de registro heredando por supuesto de nuestro layout.



**Nota:** El campo avatar no es requerido

**(Subir nota)** Puedes añadir también la modificación del password añadiendo otro formulario debajo con la contraseña actual y la nueva contraseña

Método	URL	Acción	Vista
Get	/user/config	UserController@config	User.config
Post o put	/user/config/	UserController@update	
Get	/user/avatar/{filename}	UserController@getImage	

**Nota:** Para impedir accesos indebidos puedes proteger las rutas o los controladores con el método `__construct`

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

- En el método update vamos a recuperar la id del usuario autenticado a partir del objeto \$user: **\$user= Auth::user();**
- En laravel la validación de los formularios tiene mucha posibilidades que podeis ver en la documentación oficial. En nuestro caso podría utilizar el código de la validación del formulario de registro y modificarlo como indica la siguiente imagen

```
// Validación del formulario
$validate = $this->validate($request, [
    'name' => 'required|string|max:255',
    'surname' => 'required|string|max:255',
    'nick' => 'required|string|max:255|unique:users,nick, '.$id,
    'email' => 'required|string|email|max:255|unique:users,email, '.$id
]);
```

Fijaros en el Nick y email que comprueba que es único (unique) en la Base de datos, esta condición es perfecta en la creación, pero en la actualización tenemos que indicarle que si el Nick coincide con el de esa id lo permita.

- Tendremos que tener en cuenta que para que el formulario suba imágenes tiene que tener la propiedad **enctype="multipart/formdata"**  
En Laravel para guardar imágenes utiliza discos virtuales para tener protegido y organizados dichos archivos. La carpeta **/storage/app** crearemos una carpeta por cada disco virtual.

En principio vamos a crear 2 discos virtuales

- Las imágenes que suben los usuarios y su avatar (users)
- Las imágenes generales de la plataforma

Para crear un disco virtual tenemos que configurar **/config/filesystems.php**

```
'users' => [
    'driver' => 'local',
    'root' => storage_path('app/users'),
    'url' => env('APP_URL').'/storage',
    'visibility' => 'public',
],
```

- Para trabajar con imágenes en Laravel tendrás que importar las siguientes clases en el controlador

```
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\File;
```

- Dentro del método update se utilizará el siguiente código para subir la imagen a tu disco virtual users (/app/users)

```
// Subir la imagen
$image_path = $request->file('image_path');
if($image_path){
    // Poner nombre unico
    $image_path_name = time().$image_path->getClientOriginalName();

    // Guardar en la carpeta storage (storage/app/users)
    Storage::disk('users')->put($image_path_name, File::get($image_path));

    // Seteo el nombre de la imagen en el objeto
    $user->image = $image_path_name;
}
```

- Recuerda informar al usuario si todo ha ido correctamente en todos los formularios de tu aplicación web con las **variables de sesión**.  
Para este formulario si todo ha ido correctamente redijete al formulario y muestra un mensaje “ Tus datos han sido actualizado correctamente”.  
Utiliza las alertas de bootstrap para mostrar el mensaje de la siguiente manera.

```
<div class="row justify-content-center">
    <div class="col-md-8">
        @if(session('message'))
            <div class="alert alert-success">
                {{ session('message') }}
            </div>
        @endif
    </div>
</div>
```

- También se mostrará el avatar en el formulario al lado de su campo y arriba en el menú al lado del nombre de forma circular, para eso utilizaremos el método **getImage**.

```
public function getImage($filename){
    $file = Storage::disk('users')->get($filename);
    return new Response($file, 200);
}
```

Recuerda que el avatar es un campo no requerido y puede estar vacío.  
Antes de mostrar comprueba si tiene y si no muestra una imagen por defecto

## 6. Imágenes de la aplicación

## 7. Sistema de comentarios

**8. Sistema de likes**

**9. Perfiles de usuarios**

**10. Edición y borrado de imágenes**

**11. Gente y buscador**