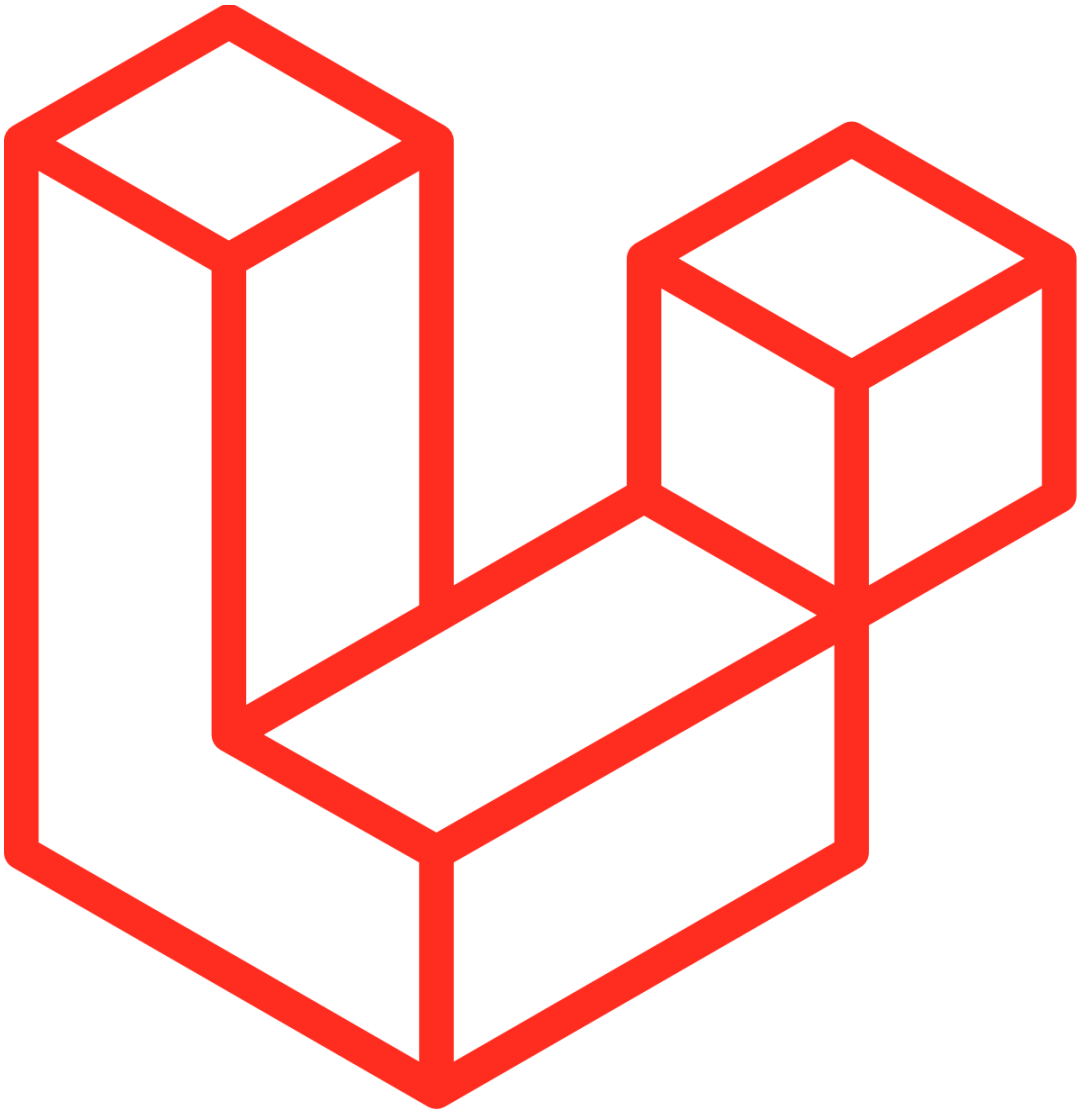
Laravel

Especialidad PHP



Felipe Izquierdo Romero

Índice

1. Descripción del problema

Crear una web de registro de películas. En ella se podrá visualizar una lista de todas las películas. Cada película tendrá nombre, género, año del estreno, puntuación del espectador y sinopsis.

En el listado general aparecerán nombre género y año de estreno. En la vista especifica de una peli se visualizarán los demás datos.

2. Desarrollo

Primero veremos la migración tras ejecutar el comando:

```
php artisan make:migration create_peliculas_table --create=peliculas
```

Para crear el esqueleto de la migración y posteriormente definir las columnas de la tabla en la función créate. Cabe destacar que no uso un campo “YEAR” para guardar el año por la siguiente razón en el comentario de la función.

```php
public function up(): void
{
    Schema::create('peliculas', function (Blueprint $table) {
        $table->id();
        $table->timestamps();
        $table->string('nombre');
        $table->string('genero');
        $table->integer('anio');
        // year en mysql 1901 hasta 2155
        // Yo verifico entr 1888 y 2024
        $table->integer('puntuacion');
        $table->text('sinopsis');
    });
}
```

Ilustración 1- Migración tabla películas

A continuación, definimos el modelo de Película usando de nuevo un comando para crear la estructura.

```
php artisan make:model Película
```

Pondremos dentro del atributo “$filleable” cada atributo correspondiente a una columna de la base de datos que queramos gestionar.

```php
class Pelicula extends Model
{
    use HasFactory;

    protected $fillable =[
        'nombre',
        'genero',
        'anio',
        'puntuacion',
        'sinopsis'
    ];
}
```

Ilustración 2- Modelo Película

Pasaremos ahora a definir las vistas usando la herencia que te proporcionan las plantillas blade.

En la plantilla padre llamada “layout.blade.php” tendrá el esqueleto base de una plantilla html y haremos referencia al repositorio de Bootstrap para dar estilo a nuestra web.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Fonts -->
        <link rel="preconnect" href="https://fonts.bunny.net">
        <link href="https://fonts.bunny.net/css?family=figtree:400,600&display=swap" rel="stylesheet" />
        <!-- <link rel="stylesheet" href="{{ asset('css/styles.css') }}"> -->
        <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

        <title>@yield('title', 'Peliculas')</title>


    </head>
    <body>
        <div class='container'>
            @yield('content')
        </div>
    </body>
</html>
```

Ilustración 3- Plantilla padre

La plantilla principal “index.blade.php” que cargara las películas con la etiqueta “@extends(‘layout’)” recibe la herencia y con la etiqueta “@section(‘content’)” la sobrescribe.

Dentro de esta etiqueta hay una tabla para cargar todas las películas en un bucle foraech. En cada línea de la tabla el atributo “onclick” se asigna la función en JavaScript de verPelicula() que recibe por parámetro el id de cada película.

En la ultima columna de la tabla se añaden dos formularios con dos botones el de eliminar y el de modificar. Cada uno redirige a su rrespectivo Endpoint para realizar la acción.

Por último un botón debajo de la tabla para crear una película.

```
<div class='container'>
    <h1>Lista de peliculas disponibles</h1>
    <table class="table table-striped">
        <thead>
            <tr class="table-primary">
                <th>Nombre</th>
                <th>Género</th>
                <th>Año</th>
            </tr>
        </thead>

        <tbody>
            @isset($peliculas)
                @foreach($peliculas as $pelicula)
                    <tr class="darken-on-hover p-3" onclick="verPelicula({{ $pelicula->id }})">
                        <td class="table-light">{{ $pelicula->nombre }}</td>
                        <td class="table-light">{{ $pelicula->genero }}</td>
                        <td class="table-light d-flex botones_tabla" style="justify-content:space-between; align-items:center;">{{ $pelicula->anio }}
                            <form action="{{ route('eliminar', $pelicula->id) }}" method="POST" onsubmit="return confirm('¿Seguro que quieres eliminar esta película?');">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="btn btn-danger">Borrar</button>
                            </form>
                            <form action="{{ route('pre_modificar', $pelicula->id) }}" method="GET">
                                @csrf
                                <button type="submit" class="btn btn-warning">Modificar</button>
                            </form>
                        </td>

                    </tr>
                @endforeach
            @endisset
        </tbody>
    </table>
    <button class="btn btn-primary" onclick="crearPelicula()">Añadir pelicula</button>
</div>
```

Ilustración 4- Tabla de películas

```
    <script>
        function verPelicula(id)
        {
            console.log('ID de la película:', id);
            window.location.href = "{{ url('/peliculas') }}/" + id;
        }
        function crearPelicula()
        {
            window.location.href = "{{ url('/crear_pelicula') }}";
        }
    </script>
```

Ilustración 5- Script de la vista

Ahora veremos la plantilla del formulario de datos de una película “form.blade.php”. Sirve para crearla de cero o para cargar los datos de una existente y modificarlos.

Haciendo uso de la etiqueta “@isset()” y la etiqueta “@if()” y “@else()” compruebo si se pasó una película a esta vista es señal de que el formulario tendrá la función de modificar datos. En el caso contrario creará una película nueva.

Además, con un operador ternario cargo los valores en los inputs si son necesarios para modificar, de lo contrario al crear una película me daría error.

```
@extends('layout')
@section('content')
@isset($pelicula)
    @if(is_null($pelicula))
        <form action="{{ route('formulario.enviar') }}" method="post">
    @else
        <form action="{{ route('modificar', $pelicula->id) }}" method="POST">
            @method('POST')
    @endif
@else
    <form action="{{ route('formulario.enviar') }}" method="post">
@endif
    @csrf
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" class="form-control" value="{{ isset($pelicula) ? $pelicula->nombre : '' }}" required>

    <label for="genero">Genero:</label>
    <input type="text" id="genero" name="genero" class="form-control" value="{{ isset($pelicula) ? $pelicula->genero : '' }}" required>

    <label for="anio">Año:</label>
    <input type="number" id="anio" name="anio" class="form-control" value="{{ isset($pelicula) ? $pelicula->anio : '' }}" required>

    <label for="puntuacion">Puntuación:</label><br>
    <div class="btn-group" role="group" aria-label="Puntuación">
        <button type="button" class="btn btn-secondary" data-value="1" onclick="clicado(1,this)">1</button>
        <button type="button" class="btn btn-secondary" data-value="2" onclick="clicado(2,this)">2</button>
        <button type="button" class="btn btn-secondary" data-value="3" onclick="clicado(3,this)">3</button>
        <button type="button" class="btn btn-secondary" data-value="4" onclick="clicado(4,this)">4</button>
        <button type="button" class="btn btn-secondary" data-value="5" onclick="clicado(5,this)">5</button>
        <input type="hidden" id="puntuacion" name="puntuacion" value="{{ isset($pelicula) ? $pelicula->puntuacion : '' }}" required>
    </div>
    <br>

    <label for="sinopsis">Sinopsis:</label>
    <input type="text" id="sinopsis" name="sinopsis" class="form-control" value="{{ isset($pelicula) ? $pelicula->sinopsis : '' }}" required>

    @isset($pelicula)
        <br><button type="submit" class="btn btn-primary">Actualizar</button>
    @else
        <br><button type="submit" class="btn btn-primary">Enviar</button>
    @endisset

</form>
<form action="{{ route('index') }}" method="get">
        <br><button type="submit" class="btn btn-primary">Volver</button>
</form>
```

Ilustración 6- Formulario Modificar/Crear

Esta vista además contiene scripts para guardar el input de puntuación pulsado a través de botones cuando se crea una película y un script para cargar en el input y el botón pulsado la puntuación al modificar una película.

```
<script>

    document.addEventListener('DOMContentLoaded', function() {
        var puntuacionInput = document.getElementById('puntuacion');
        var puntuacionValue = '{{ isset($pelicula) ? $pelicula->puntuacion : '' }}';

        // Si hay una puntuación preseleccionada, encontrar y activar el botón correspondiente
        if (puntuacionValue !== '') {
            var buttons = document.querySelectorAll('.btn-group button');
            buttons.forEach(function(btn) {
                if (btn.getAttribute('data-value') === puntuacionValue) {
                    btn.classList.add('active');
                }
            });
        }
    });

    function clicado(value, boton) {
        var input = document.getElementById('puntuacion');
        input.value = value;

        // Remover la clase 'active' de todos los botones
        var buttons = document.querySelectorAll('.btn-group button');
        buttons.forEach(function(btn) {
            btn.classList.remove('active');
        });

        // Agregar la clase 'active' solo al botón clicado
        boton.classList.add('active');
    }
</script>
```

Ilustración 7- Scripts Modificar/Crear

Para finalizar la descripción de las vistas esta la plantilla “show.blade.php” que simplemente carga los datos de la película en una lista.

```
@extends('layout')

@section('content')
    <div class="content">
        <ul class="list-group">
            @isset($pelicula)
            <li class="list-group-item" style="background-color:rgba(184,218,255,100);"><h1>{{$pelicula->nombre}}</h1></li>

            <li class="list-group-item">Genero: {{$pelicula->genero}}</li>
            <li class="list-group-item">Año de publicación: {{$pelicula->anio}}</li>
            <li class="list-group-item">Puntuacion: {{$pelicula->puntuacion}}</li>
            <li class="list-group-item">Sinopsis:<br>{{$pelicula->sinopsis}}</li>
            @endisset()
        </ul>
    </div>
    <form action="{{ route('index') }}" method="get">
        <br><button type="submit" class="btn btn-primary">Volver</button>
    </form>
@endsection()
```

Ilustración 8- Lista de datos de una película

El controlador que gestiona las llamadas y carga los datos se llama “PeliculaController.php” y contiene las funciones:

- Index: Carga todas las películas y las carga en la vista “index.blade.php”.

```php
public function index()
{
    $peliculas = Pelicula::all();
    return view('index',compact('peliculas'));
}
```

Ilustración 9- Función index

- Show: Carga una película y la pasa a “show.blade.php”.

```php
public function show($id)
{
    $pelicula = Pelicula::findOrFail($id);
    return view('show',compact('pelicula'));
}
```

Ilustración 10- Función show

- mostrarFormulario: Muestara la vista “form.blade.php”.

```php
public function mostrarFormulario()
{
    return view('form');
}
```

Ilustración 11- Función mostrar formulario

- crear: Valida los datos del formulario y da la señal para guardar la película.

```php
public function crear(Request $request)
{

    $validateData = $request->validate([
        'nombre' => 'required|string',
        'genero' => 'required|string',
        'anio' => ['required', new YearValidationRule],
        'puntuacion' => 'required',
        'sinopsis' => 'required|string',
    ]);

    $pelicula = Pelicula::create($validateData);

    return redirect()->route('index')->with('success', 'Formulario enviado correctamente');
}
```

Ilustración 12- Función crear

- eliminar: busca la película por su identificador y si existe la elimina. Redirigé al index en cualquier caso.

```php
public function eliminiar($id)
{
    $pelicula = Pelicula::find($id);

    if ($pelicula) {
        $pelicula->delete();
        return redirect()->route('index')->with('success', 'Película eliminada correctamente.');
    }

    return redirect()->route('index')->with('error', 'Película no encontrada.');
}
```

Ilustración 13- Función eliminar

- preModificar: busca una película y si la encuentra carga los datos en la plantilla "form.blade.php".

```php
public function pre_modificar($id)
{
    $pelicula = Pelicula::find($id);
    if ($pelicula) {
        return view('form', compact('pelicula'));
    }
}
```

Ilustración 14- Función preModificar

- modificar: verifica que los datos del formulario son correctos, luego busca la película por su identificador y le carga los valores recogidos y los guarda y redirige al index.

```php
public function modificar(Request $request, $id)
{

    $validateData = $request->validate([
        'nombre' => 'required|string',
        'genero' => 'required|string',
        'anio' => ['required', new YearValidationRule],
        'puntuacion' => 'required',
        'sinopsis' => 'required|string',
    ]);

    $pelicula = Pelicula::find($id);

    if ($pelicula) {
        $pelicula->nombre = $validateData['nombre'];
        $pelicula->genero = $validateData['genero'];
        $pelicula->anio = $validateData['anio'];
        $pelicula->puntuacion = $validateData['puntuacion'];
        $pelicula->sinopsis = $validateData['sinopsis'];

        $pelicula->save();

        return redirect()->route('index')->with('success', 'Película actualizada correctamente.');
    }

    return view('index');
}
```

Ilustración 15- Función modificar

Para finalizar el lab solo queda mostrar las rutas definidas en el “web.php” y el “ValidationRule” que sirve para validar que el año de la película esta en el rango definido como válido.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\IndexController;
use App\Http\Controllers\PeliculaController;


Route::get('/', [PeliculaController::class, 'index'])
    ->name('index');

Route::get('/peliculas/{id}', [PeliculaController::class, 'show'])
    ->name('show');

Route::get('/crear_pelicula', [PeliculaController::class, 'mostrarFormulario'])
    ->name('formulario.show');

Route::post('/crear_pelicula/enviar', [PeliculaController::class, 'crear'])
    ->name('formulario.enviar');

// Route::resource('/', [PeliculaController::class, 'borrar']);
Route::delete('/eliminar/{id}', [PeliculaController::class, 'eliminiar'])
    ->name('eliminar');

Route::get('/pre_modificar/{id}', [PeliculaController::class, 'pre_modificar'])
    ->name('pre_modificar');

Route::post('/modificar/{id}', [PeliculaController::class, 'modificar'])
    ->name('modificar');
```

Ilustración 16- Rutas definidas en "web.php"

```php
<?php

namespace App\Rules;

use Closure;
use Illuminate\Contracts\Validation\ValidationRule;

class YearValidationRule implements ValidationRule
{
    /**
     * Run the validation rule.
     *
     * @param  \Closure(string): \Illuminate\Translation\PotentiallyTranslatedString  $fail
     */
    public function validate(string $attribute, mixed $value, Closure $fail): void
    {
        if($value < 1888 || $value > 2024)
        {
            $fail('El año debe de estar entre 1888 y 2024.');
        }

    }

}
```

Ilustración 17- Validador de año