Examen Práctico de Laravel:

Gestión de Películas de Harry Potter y Directores de las mismas

“Las decisiones que tomamos muestran quiénes somos, mucho más que las habilidades que poseemos” (Albus Dumbledore)

Contenido

Objetivo:

Crear una aplicación básica para gestionar un listado de películas de Harry Potter, donde cada película esté asociada a un director. Los usuarios podrán:

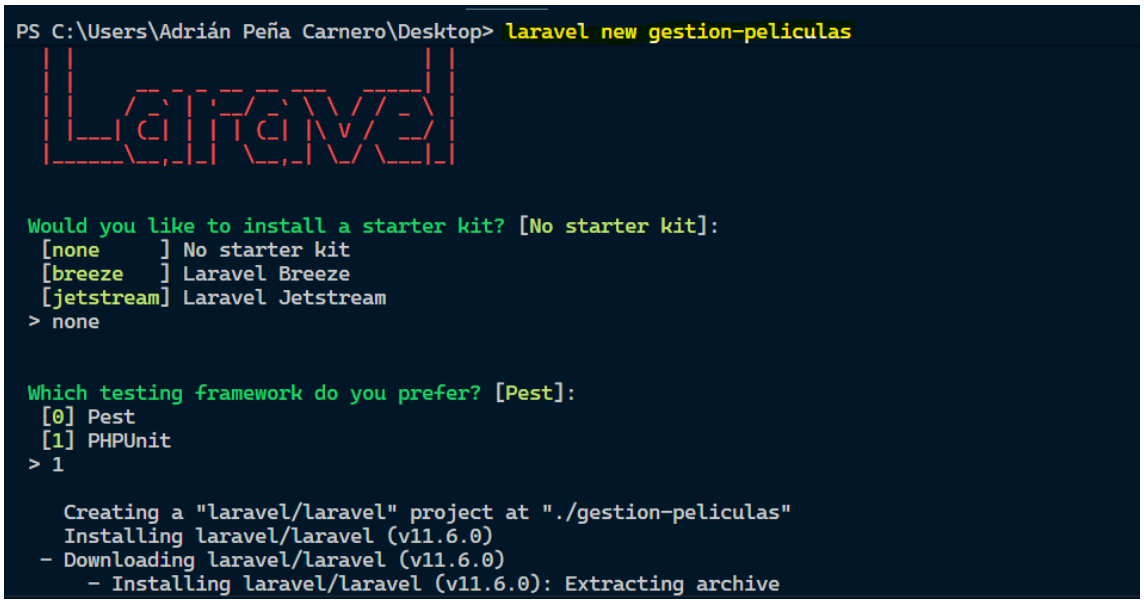
- Ver un listado de películas.
- Crear nuevas películas y asignarles un director.
- Editar y eliminar películas existentes.

---

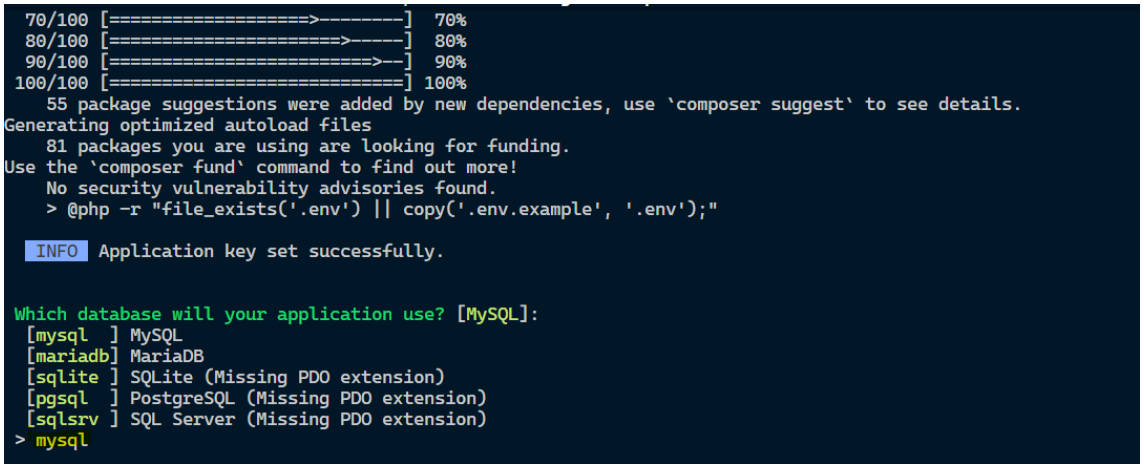
1. Instrucciones:

Crea un proyecto nuevo de Laravel utilizando:

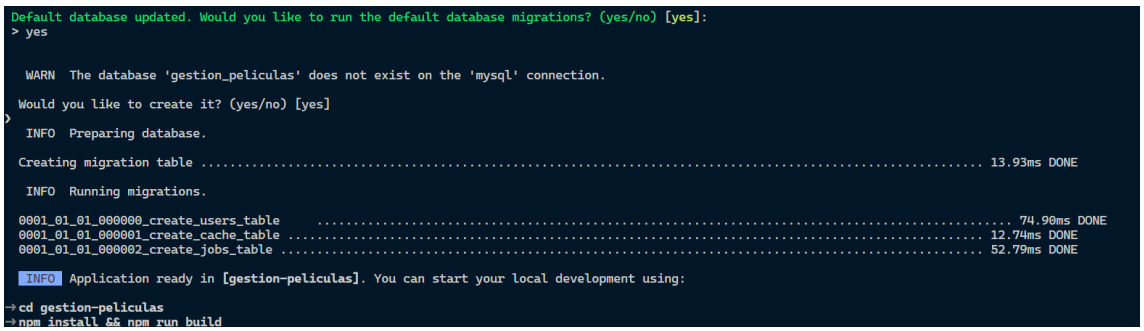
```
1. laravel new gestion-peliculas
```



Aquí nos preguntara que bd por defecto queremos utilizar:



Le decimos que nos añada las tablas por defecto también:

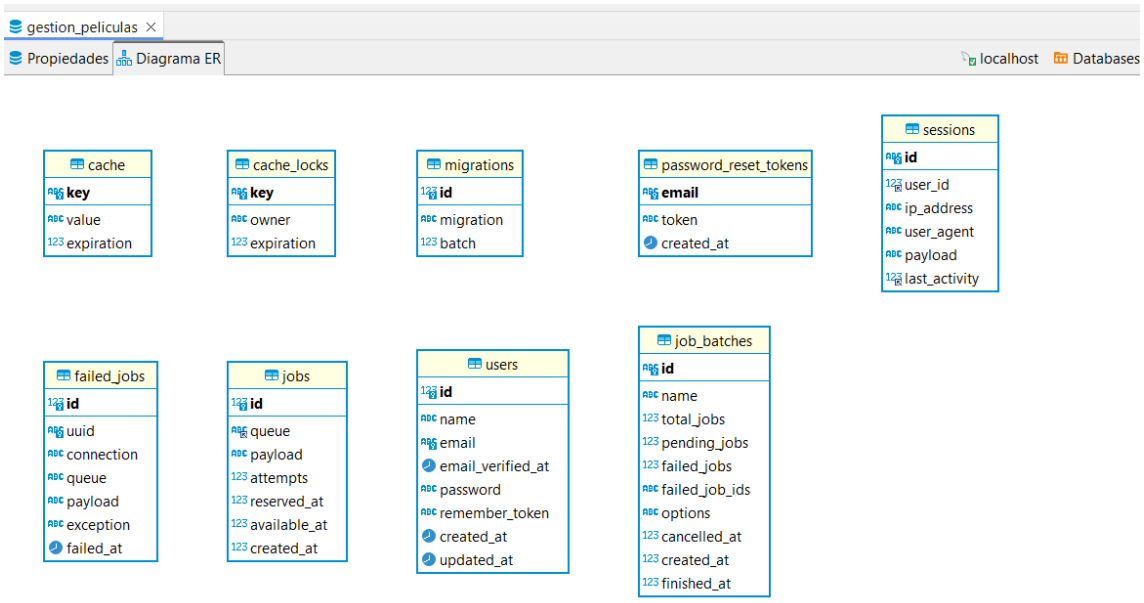


Configura la base de datos en el archivo. env para usar MySQL.

Una vez dentro del proyecto nos dirigimos a configurar el archivo **.env** de tal manera que configuramos los parámetros para el acceso a la conexión con la bd.

```
.env
LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=gestion_peliculas
DB_USERNAME=root
DB_PASSWORD=

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
```

Una vez dentro el gestor podremos apreciar las tablas por defecto que tiene laravel:



2. Completa los pasos siguientes para desarrollar la aplicación.
3. Realiza capturas de pantalla de cada paso e inclúyelas en un documento PDF que entregarás junto con el código en un archivo comprimido (ZIP).

---

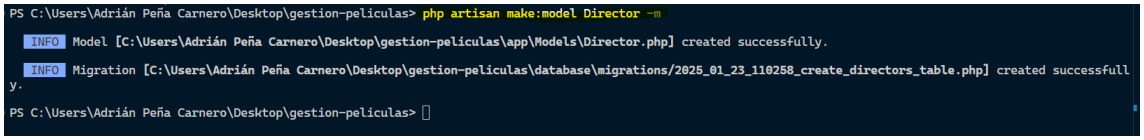
2. Requisitos:

1. Modelos y Migraciones

Crea los modelos y sus migraciones correspondientes para las siguientes entidades:

1. **Director**:
    - **nombre** (string, requerido).
    - **fecha_nacimiento** (date, requerido).
        - *Formato esperado para la fecha:* `YYYY-MM-DD` *(ejemplo: 1965-07-31).*

Con el comando php artisan make:model Director -m crearemos el modelo y la migración a la vez:



Editaremos la migración con los datos requeridos:

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'directors', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'nombre');
            $table->date(column: 'fecha_nacimiento');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'directors');
    }
};
```

Ahora pasaríamos a configurar el modelo con su propia relación:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

1 reference | 0 implementations
class Director extends Model
{

    use HasFactory;

    0 references
    protected $fillable = ['autor','contenido','post_id'];



    0 references | 0 overrides
    public function peliculas(): HasMany {
        return $this->hasMany(related: Pelicula::class);
    }


}
```

2. **Pelicula**:
    - **titulo** (string, requerido).
    - **sinopsis** (texto, opcional).
    - **anio** (entero, requerido, entre 2000 y 2025).

- **director_id** (entero, clave foránea que referencia al id de la tabla directors).

(Procedemos a realizar los pasos dichos anteriormente, pero con los parámetros requeridos en este caso)

```
PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas> php artisan make:model Pelicula -m

  INFO  Model [C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas\app\Models\Pelicula.php] created successfully.

  INFO  Migration [C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas\database\migrations/2025_01_23_110344_create_peliculas_table.php] created successfully.

PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas>
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

1 reference | 0 implementations
class Pelicula extends Model
{


    0 references
    protected $fillable = ['titulo','sinopsis','anio','director_id'];
    0 references | 0 overrides
    public function director(): BelongsTo {
        return $this->belongsTo(related: Director::class);
    }

}
```

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'peliculas', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'titulo');
            $table->text(column: 'sinopsis')->nullable();
            $table->year(column: 'anio');
            $table->foreignId(column: 'director_id')->constrained(table: 'directors')->onDelete(action: 'cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'peliculas');
    }
};
```

2. Relaciones

Define las relaciones entre los modelos:

- **Director**: Un director puede haber dirigido muchas películas (relación uno a muchos).

Como podemos ver un director puede tener muchas películas y lo dejaríamos tal que así:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

1 reference | 0 implementations
class Director extends Model
{

    use HasFactory;

    0 references
    protected $fillable = ['autor','contenido','post_id'];



    0 references | 0 overrides
    public function peliculas(): HasMany {
        return $this->hasMany(related: Pelicula::class);
    }


}
```

- **Pelicula**: Una película pertenece a un único director.

Una película pertenece a un único autor:

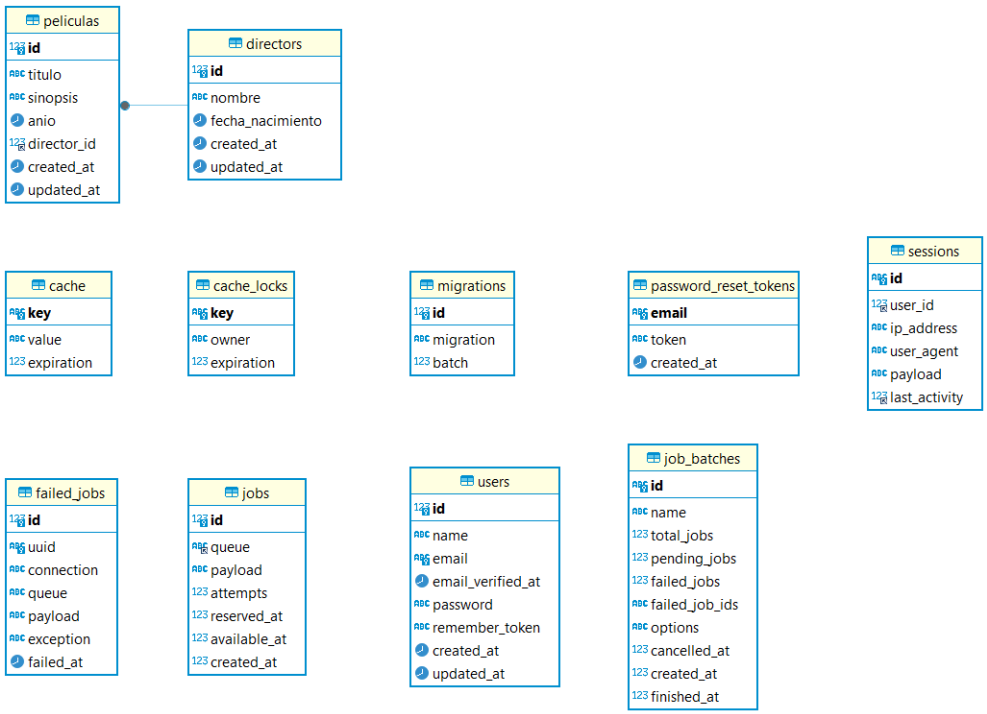
```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

1 reference | 0 implementations
class Pelicula extends Model
{


    0 references
    protected $fillable = ['titulo','sinopsis','anio','director_id'];
    0 references | 0 overrides
    public function director(): BelongsTo {
        return $this->belongsTo(related: Director::class);
    }

}
```

Configura estas relaciones en los modelos `Director` y `Película`

2.1 Ejecutar migraciones

Ahora ejecutaríamos las migraciones para tener todas las tablas de la bd:

```
PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas> php artisan migrate

  WARN  The database 'gestion_peliculas' does not exist on the 'mysql' connection.

  Would you like to create it? (yes/no) [yes]
  > yes

  INFO  Preparing database.

  Creating migration table ..... 79.97ms DONE

  INFO  Running migrations.

  0001_01_01_000000_create_users_table ..... 78.42ms DONE
  0001_01_01_000001_create_cache_table ..... 16.48ms DONE
  0001_01_01_000002_create_jobs_table ..... 83.48ms DONE
  2025_01_23_110258_create_directors_table ..... 8.89ms DONE
  2025_01_23_110344_create_peliculas_table ..... 47.05ms DONE

PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas>
```



---

3. Controlador y Rutas

Crea un controlador llamado `PeliculaController` con las siguientes acciones:

1. **index()**: Devuelve una vista con el listado de películas y sus directores.
2. **create()**: Devuelve una vista con un formulario para crear una nueva película.
    - El formulario debe incluir:
        - Título.
        - Sinopsis.
        - Año (campo de tipo `number`, con valores entre 2000 y 2025).
        - Desplegable para seleccionar un director existente.
3. **store(Request $request)**: Almacena una nueva película en la base de datos.
4. **edit($id)**: Devuelve una vista con un formulario para editar una película existente.
    - El formulario debe incluir los datos actuales de la película y permitir cambiar el director.
5. **update(Request $request, $id)**: Actualiza los datos de una película en la base de datos.
6. **destroy($id)**: Elimina una película de la base de datos.

Configura las rutas necesarias en el archivo `web.php`

```
PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas> php artisan make:controller PeliculaController --resource

  INFO  Controller [C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas\app\Http\Controllers\PeliculaController.php] created successfully.

PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas>
```

```php
<?php

namespace App\Http\Controllers;

use App\Models\Pelicula;
use App\Models\Director;
use Illuminate\Http\Request;

0 references | 0 implementations
class PeliculaController extends Controller
{
    /**
     * Devuelve una vista con el listado de películas y sus directores.
     */
    0 references | 0 overrides
    public function index(): View
    {
        $peliculas = Pelicula::with(relations: 'director')->get();
        return view(view: 'peliculas.index', data: compact(var_name: 'peliculas'));
    }

    /**
     * Devuelve una vista con un formulario para crear una nueva película.
     */
    0 references | 0 overrides
    public function create(): View
    {
        $directores = Director::all();
        return view(view: 'peliculas.create', data: compact(var_name: 'directores'));
    }
```

```php
/**
 * Almacena una nueva película en la base de datos.
 */
0 references | 0 overrides
public function store(Request $request): RedirectResponse
{
    $request->validate(rules: [
        'titulo' => 'required|max:255',
        'anio' => 'required|integer|between:2000,2025',
        'director_id' => 'required|exists:directors,id',
    ]);

    Pelicula::create(attributes: [
        'titulo' => $request->titulo,
        'sinopsis' => $request->sinopsis,
        'anio' => $request->anio,
        'director_id' => $request->director_id,
    ]);

    return redirect()->route(route: 'peliculas.index')->with(key: 'success', value: 'Película creada con éxito.');
}

/**
 *  Devuelve una vista con un formulario para editar una película existente.
 */
0 references | 0 overrides
public function edit($id): View
{
    $pelicula = Pelicula::findOrFail(id: $id);
    $directores = Director::all();
    return view(view: 'peliculas.edit', data: compact(var_name: 'pelicula', var_names: 'directores'));
}
```

```php
/**
 * Actualiza los datos de una película en la base de datos.
 */
0 references | 0 overrides
public function update(Request $request, $id): RedirectResponse
{
    $request->validate(rules: [
        'titulo' => 'required|max:255',
        'anio' => 'required|integer|between:2000,2025',
        'director_id' => 'required|exists:directors,id',
    ]);

    $pelicula = Pelicula::findOrFail(id: $id);
    $pelicula->update([
        'titulo' => $request->titulo,
        'sinopsis' => $request->sinopsis,
        'anio' => $request->anio,
        'director_id' => $request->director_id,
    ]);

    return redirect()->route(route: 'peliculas.index')->with(key: 'success', value: 'Película actualizada con éxito.');
}

/**
 * Elimina una película de la base de datos.
 */
0 references | 0 overrides
public function destroy($id): RedirectResponse
{
    $pelicula = Pelicula::findOrFail(id: $id);
    $pelicula->delete();

    return redirect()->route(route: 'peliculas.index')->with(key: 'success', value: 'Película eliminada con éxito.');
}
```

Rutas

```php
<?php

use App\Http\Controllers\PeliculaController;
use App\Models\Director;
use Illuminate\Support\Facades\Route;

Route::get(uri: '/', action: function (): View {
    return view(view: 'peliculas.index');
});


Route::resource(name: 'peliculas', controller: PeliculaController::class);

Route::get(uri: '/', action: [PeliculaController::class, 'index'])->name(name: 'peliculas.index');
```

---

4. Vistas

Crea las vistas necesarias en la carpeta `resources/views/peliculas`:

1. **index.blade.php**:
    - Muestra un listado de películas con las siguientes columnas:
        - Título de la película.
        - Sinopsis.
        - Año.
        - Director.
        - Acciones (Editar y Eliminar).
    - Incluye un botón para "Crear nueva película" que enlace a `peliculas.create`.

Vista principal

```blade
@extends(view: 'layout')

@section(section: 'title', content: 'Listado de Películas')

@section(section: 'content')
<div class="container">
    <h1>Listado de Películas</h1>
    <a href="{{ route(name: 'peliculas.create') }}" class="btn btn-primary">Crear nueva película</a>
    <table class="table">
        <thead>
            <tr>
                <th>Título</th>
                <th>Sinopsis</th>
                <th>Año</th>
                <th>Director</th>
                <th>Acciones</th>
            </tr>
        </thead>
        <tbody>
            @foreach($peliculas as $pelicula)
            <tr>
                <td>{{ $pelicula->titulo }}</td>
                <td>{{ $pelicula->sinopsis }}</td>
                <td>{{ $pelicula->anio }}</td>
                <td>{{ $pelicula->director->nombre }}</td>
                <td>
                    <a href="{{ route(name: 'peliculas.edit', parameters: $pelicula->id) }}" class="btn btn-warning">Editar</a>
                    <form action="{{ route(name: 'peliculas.destroy', parameters: $pelicula->id) }}" method="POST" style="display:inline;">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-danger">Eliminar</button>
                    </form>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>
</div>
@endsection
```

2. **create.blade.php**:
    - Contiene un formulario para crear una nueva película con los campos:
        - **Título** (requerido).
        - **Sinopsis** (opcional).
        - **Año** (requerido, valores entre 2000 y 2025).
        - **Desplegable para seleccionar un director existente**.
        - **Botón "Guardar"**.

Vista creación:

```blade
@extends('layout')

@section('title', 'Crear Película')

@section('content')
<div class="container">
    <h1>Crear Película</h1>
    <form action="{{ route('peliculas.store') }}" method="POST">
        @csrf
        <div class="form-group">
            <label for="titulo">Título</label>
            <input type="text" name="titulo" id="titulo" class="form-control" required>
        </div>
        <div class="form-group">
            <label for="sinopsis">Sinopsis</label>
            <textarea name="sinopsis" id="sinopsis" class="form-control"></textarea>
        </div>
        <div class="form-group">
            <label for="anio">Año</label>
            <input type="number" name="anio" id="anio" class="form-control" min="2000" max="2025" required>
        </div>
        <div class="form-group">
            <label for="director_id">Director</label>
            <select name="director_id" id="director_id" class="form-control" required>
                @foreach($directores as $director)
                <option value="{{ $director->id }}">{{ $director->nombre }}</option>
                @endforeach
            </select>
        </div>
        <button type="submit" class="btn btn-primary">Guardar</button>
    </form>
</div>
@endsection
```

Vista edición:

3. **edit.blade.php**:
    - Contiene un formulario similar al de creación, pero precargado con los datos de la película a editar.

```blade
@extends('layout')

@section('title', 'Editar Película')

@section('content')
<div class="container">
    <h1>Editar Película</h1>
    <form action="{{ route('peliculas.update', parameters: $pelicula->id) }}" method="POST">
        @csrf
        @method('PUT')
        <div class="form-group">
            <label for="titulo">Título</label>
            <input type="text" name="titulo" id="titulo" class="form-control" value="{{ $pelicula->titulo }}" required>
        </div>
        <div class="form-group">
            <label for="sinopsis">Sinopsis</label>
            <textarea name="sinopsis" id="sinopsis" class="form-control">{{ $pelicula->sinopsis }}</textarea>
        </div>
        <div class="form-group">
            <label for="anio">Año</label>
            <input type="number" name="anio" id="anio" class="form-control" min="2000" max="2025" value="{{ $pelicula->anio }}" required>
        </div>
        <div class="form-group">
            <label for="director_id">Director</label>
            <select name="director_id" id="director_id" class="form-control" required>
                @foreach($directores as $director)
                <option value="{{ $director->id }}" {{ $pelicula->director_id == $director->id ? 'selected' : '' }}>{{ $director->nombre }}</option>
                @endforeach
            </select>
        </div>
        <button type="submit" class="btn btn-primary">Actualizar</button>
    </form>
</div>
@endsection
```

5. Validación

Implementa validaciones en los métodos `store()` y `update()`:

- El campo `titulo` debe ser requerido y tener un máximo de 255 caracteres.
- El campo `anio` debe ser un número entero entre 2000 y 2025.
- El campo `director_id` debe ser requerido y corresponder a un director existente.

Validaciones correspondientes:

```php
public function store(Request $request): RedirectResponse
{
    $request->validate(rules: [
        'titulo' => 'required|max:255',
        'anio' => 'required|integer|between:2000,2025',
        'director_id' => 'required|exists:directors,id',
    ]);

    Pelicula::create(attributes: [
        'titulo' => $request->titulo,
        'sinopsis' => $request->sinopsis,
        'anio' => $request->anio,
        'director_id' => $request->director_id,
    ]);

    return redirect()->route(route: 'peliculas.index')->with(key: 'success', value: 'Película creada con éxito.');
}

/**
```

```php
/**
 * Actualiza los datos de una película en la base de datos.
 */
0 references | 0 overrides
public function update(Request $request, $id): RedirectResponse
{
    $request->validate(rules: [
        'titulo' => 'required|max:255',
        'anio' => 'required|integer|between:2000,2025',
        'director_id' => 'required|exists:directors,id',
    ]);

    $pelicula = Pelicula::findOrFail(id: $id);
    $pelicula->update([
        'titulo' => $request->titulo,
        'sinopsis' => $request->sinopsis,
        'anio' => $request->anio,
        'director_id' => $request->director_id,
    ]);

    return redirect()->route(route: 'peliculas.index')->with(key: 'success', value: 'Película actualizada con éxito.');
}
```

6. Información para Población de las Tablas

Utiliza las siguientes entradas para rellenar las tablas:

Directores:

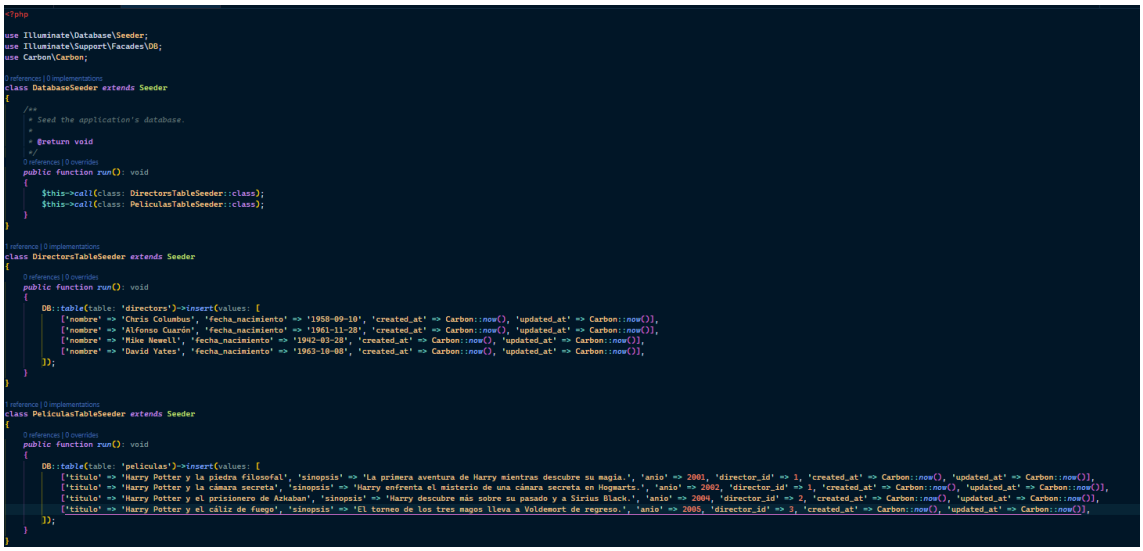
Nombre	Fecha de Nacimiento
Chris Columbus	1958-09-10
Alfonso Cuarón	1961-11-28
Mike Newell	1942-03-28
David Yates	1963-10-08

Películas:

Título	Sinopsis	Añ
Harry Potter y la piedra filosofal	La primera aventura de Harry mientras descubre su magia.	200
Harry Potter y la cámara secreta	Harry enfrenta el misterio de una cámara secreta en Hogwarts.	2
Harry Potter y el prisionero de Azkaban	Harry descubre más sobre su pasado y a Sirius Black.	2
Harry Potter y el cáliz de fuego	El torneo de los tres magos lleva a Voldemort de regreso.	200

Título **Sinopsis** **Añ**

Los datos los he insertado con el DatabaseSeeder.php y así no tenia que ir 1 a 1 insertando con el gestor de bd:



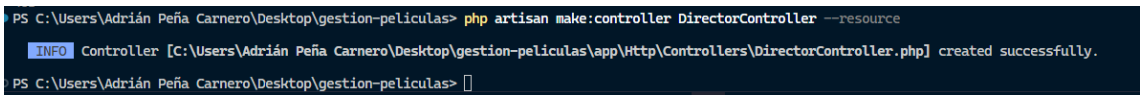
```php
<?php

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run(): void
    {
        $this->call(class: DirectorsTableSeeder::class);
        $this->call(class: PeliculasTableSeeder::class);
    }
}

class DirectorsTableSeeder extends Seeder
{
    public function run(): void
    {
        DB::table(table: 'directors')->insert(values: [
            ['nombre' => 'Chris Columbus', 'fecha_nacimiento' => '1958-09-10', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['nombre' => 'Alfonso Cuarón', 'fecha_nacimiento' => '1961-11-28', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['nombre' => 'Mike Newell', 'fecha_nacimiento' => '1942-03-28', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['nombre' => 'David Yates', 'fecha_nacimiento' => '1963-10-08', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
        ]);
    }
}

class PeliculasTableSeeder extends Seeder
{
    public function run(): void
    {
        DB::table(table: 'peliculas')->insert(values: [
            ['titulo' => 'Harry Potter y la piedra filosofal', 'sinopsis' => 'La primera aventura de Harry mientras descubre su magia.', 'anio' => 2001, 'director_id' => 1, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['titulo' => 'Harry Potter y la cámara secreta', 'sinopsis' => 'Harry enfrenta el misterio de una cámara secreta en Hogwarts.', 'anio' => 2002, 'director_id' => 1, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['titulo' => 'Harry Potter y el prisionero de Azkaban', 'sinopsis' => 'Harry descubre más sobre su pasado y a Sirius Black.', 'anio' => 2004, 'director_id' => 2, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['titulo' => 'Harry Potter y el cáliz de fuego', 'sinopsis' => 'El torneo de los tres magos lleva a Voldemort de regreso.', 'anio' => 2005, 'director_id' => 3, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
        ]);
    }
}
```

---

7. Bonus (Opcional)

Si terminas antes de tiempo, añade alguna de las siguientes funcionalidades:

- Crea un controlador y vistas para gestionar los directores (listar, crear, editar y eliminar).



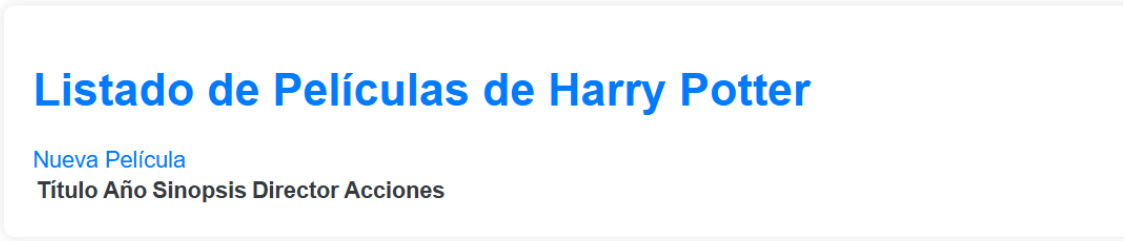
```
PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas> php artisan make:controller DirectorController --resource

  INFO  Controller [C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas\app\Http\Controllers\DirectorController.php] created successfully.

PS C:\Users\Adrián Peña Carnero\Desktop\gestion-peliculas>
```

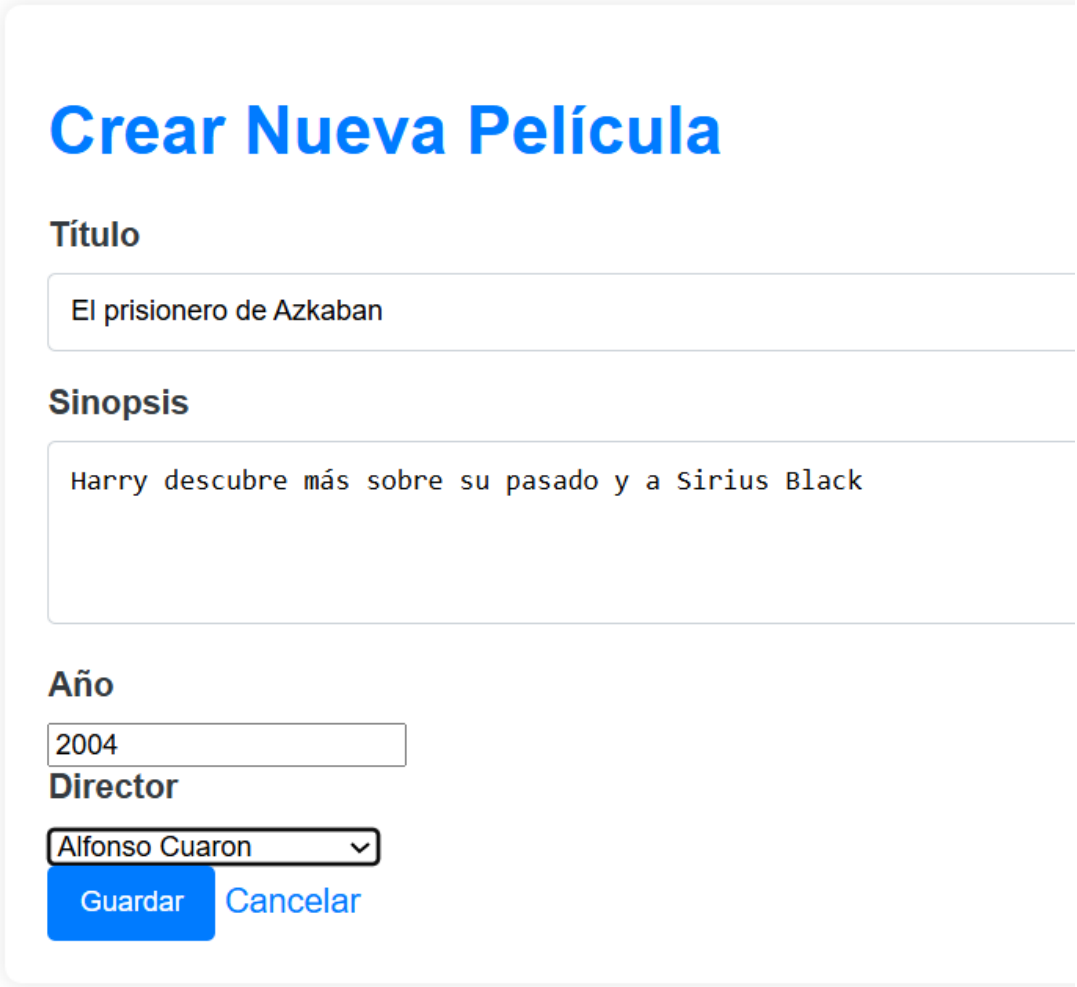
- Añade una funcionalidad para filtrar películas por director en la vista de listado.

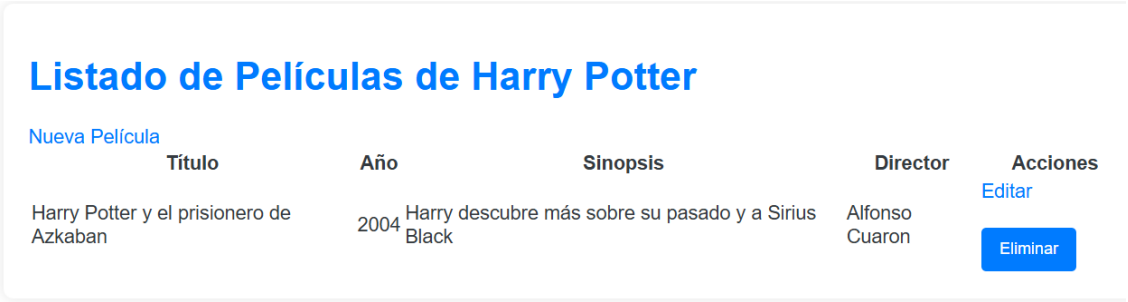
Criterios de Evaluación:

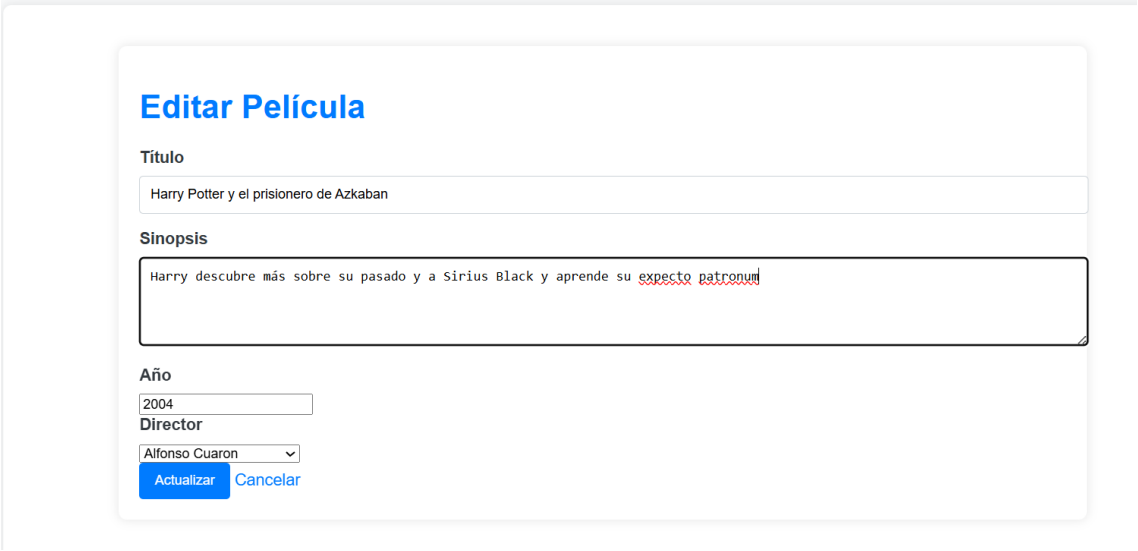
Aspecto	Porcentaje
Modelos y Migraciones	20%
Relaciones y Eloquent	15%
Controlador y Rutas	25%
Vistas y Diseño Básico	25%
Validación	15%
Funcionalidad Extra (Bonus)	10% (opcional)

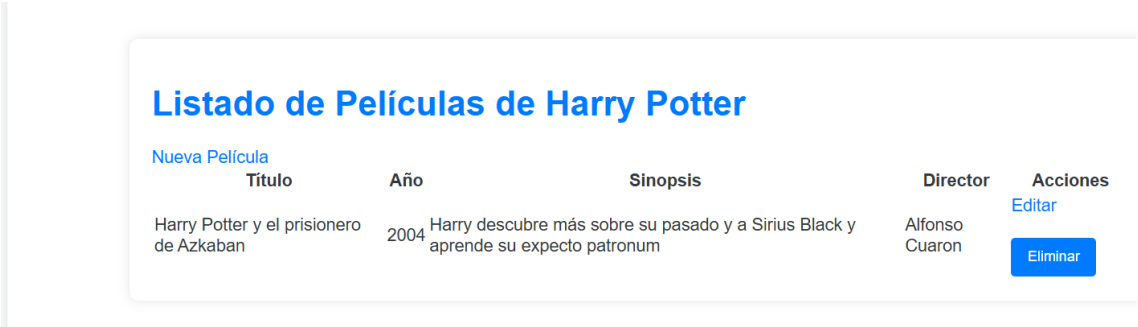
Ejemplo de ejecución:




Crear Nueva Película
Título
El prisionero de Azkaban
Sinopsis
Harry descubre más sobre su pasado y a Sirius Black
Año
2004
Director
Alfonso Cuaron
Guardar
Cancelar


Listado de Películas de Harry Potter
Nueva Película
Título
Año
Sinopsis
Director
Acciones
Harry Potter y el prisionero de Azkaban
2004
Harry descubre más sobre su pasado y a Sirius Black
Alfonso Cuaron
Editar
Eliminar


Editar Película
Título
Harry Potter y el prisionero de Azkaban
Sinopsis
Harry descubre más sobre su pasado y a Sirius Black y aprende su expecto patronum
Año
2004
Director
Alfonso Cuaron
Actualizar
Cancelar


Listado de Películas de Harry Potter
Nueva Película
Título
Año
Sinopsis
Director
Acciones
Harry Potter y el prisionero de Azkaban
2004
Harry descubre más sobre su pasado y a Sirius Black y aprende su expecto patronum
Alfonso Cuaron
Editar
Eliminar