PRÁCTICA FINAL: GESTIÓN DE PELÍCULAS Y ACTORES

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN DESARROLLO ÁGIL PARA LA WEB

2025-2026

Asignatura: Frameworks backend y microservicios

Alumno: Juan Higuero López

Fecha: 22 de febrero de 2026

Contenido

1. INTRODUCCIÓN

El presente proyecto consiste en la creación y desarrollo de una aplicación web completa para la gestión de películas, actores, usuarios y opiniones. El objetivo principal es crear este proyecto sobre una arquitectura de microservicios robusta y escalable.

Para ello, se ha utilizado el ecosistema de Spring Boot junto con Spring Cloud, implementando patrones esenciales como la centralización de servicios mediante Eureka, el enrutamiento dinámico a través de un API Gateway y la persistencia de datos con JPA y MySQL. La aplicación permite interactuar con dos dominios de datos diferenciados (el catálogo de cine y la gestión de usuarios/críticas) unificados bajo una misma interfaz de usuario desarrollada con Thymeleaf.

2. ANÁLISIS Y DISEÑO DE LA ARQUITECTURA

El presente proyecto se basa en una arquitectura distribuida donde cada componente tiene una responsabilidad única. A continuación, se detallan los elementos que componen el sistema y las decisiones técnicas tomadas.

2.1. Componentes de la Arquitectura

1. **Base de datos en Workbench (MySQL):** Para la persistencia de la información se ha seleccionado el sistema gestor de bases de datos relacional **MySQL Server**, gestionado y modelado a través de la herramienta **MySQL Workbench**. Se han creado dos esquemas: `peliculasactoresdb` y `usuariosopinionesdbsec`.

2. **Estandarización del Ecosistema**: Para garantizar la compatibilidad entre los distintos componentes, se ha unificado la versión de Spring Boot a la 3.5.9 en todos los proyectos (Backend de Películas/Actores, Backend de Usuarios/Opiniones, Cliente Web, Servidor Eureka y Gateway).

3. **Service Discovery (Eureka Server):** Se ha creado un microservicio dedicado a la detección de servicios. Actúa como un registro centralizado donde todos los microservicios se inscriben al arrancar, desacoplando las IPs físicas de los nombres lógicos. Dependencia clave: `spring-cloud-starter-netflix-eureka-server`.

4. **API Gateway:** Implementación de un servidor que actúa como puerta de enlace única. Intercepta las peticiones del Cliente (Puerto 8100) y las enruta dinámicamente hacia el microservicio correspondiente consultando a Eureka y utilizando balanceo de carga (`lb://`). Dependencias clave: `spring-cloud-starter-gateway`.

5. **Microservicios (Backend):** Se han desarrollado dos microservicios independientes encargados de la persistencia de datos y la lógica de negocio expuesta mediante API REST: uno dedicado a la gestión del catálogo cinematográfico (Películas y Actores) y otro exclusivo para la administración de Usuarios y sus Opiniones. Ambos componentes implementan Spring Data JPA para el mapeo objeto-relacional con bases de datos MySQL y operan como Clientes Eureka, garantizando su registro y descubrimiento automático dentro de la arquitectura distribuida.

6. **Cliente Web (Frontend):** Actúa como orquestador y frontend. Se ha refactorizado para incluir la gestión de imágenes local (servicio `UploadFileService`) y seguridad con Spring Security. Este cliente se encarga de unificar la información del backend sobre una misma interfaz.

2.2. Seguridad (Spring Security)

Las funcionalidades dependen del rol del usuario: el administrador gestiona toda la plataforma, el usuario registrado puede valorar películas y el visitante tiene acceso de solo lectura al contenido web.

Se ha implementado una configuración robusta basada en roles para esta lógica (RBAC):

- **WebSecurityConfig:** Define qué rutas son públicas (listado, detalles) y cuáles requieren autenticación o rol `ADMIN` (gestión, borrado).

- **Vistas:** Integración de `thymeleaf-extras-springsecurity6` para mostrar u ocultar botones de administración según el rol.

2.3. Diseño de Base de Datos (Scripts de Creación)

Se utilizan dos esquemas de MySQL independientes. A continuación, se presentan los scripts de creación de las estructuras:

Esquema 1: Usuarios y Opiniones (`usuariosopinionesdbsec`)

Gestiona usuarios, roles y críticas.

```
-- -----------------------------------------------------
-- Schema usuariosopinionesdbsec
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `usuariosopinionesdbsec` DEFAULT
CHARACTER SET utf8 COLLATE utf8_spanish_ci ;
USE `usuariosopinionesdbsec` ;

-- Table Authorities
CREATE TABLE IF NOT EXISTS `Authorities` (
  `idRol` INT NOT NULL AUTO_INCREMENT,
  `authority` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`idRol`))
ENGINE = InnoDB;

-- Table Users
CREATE TABLE IF NOT EXISTS `Users` (
  `idUsuario` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(60) NOT NULL,
  `correo` VARCHAR(45) NOT NULL,
  `enable` TINYINT NOT NULL DEFAULT 1,
  `idRol` INT NOT NULL,
  PRIMARY KEY (`idUsuario`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE,
  UNIQUE INDEX `correo_UNIQUE` (`correo` ASC) VISIBLE,
  INDEX `fk_Users_Authorities_idx` (`idRol` ASC) VISIBLE,
  CONSTRAINT `fk_Users_Authorities`
    FOREIGN KEY (`idRol`)
    REFERENCES `Authorities` (`idRol`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;

INSERT INTO `usuariosopinionesdbsec`.`authorities` (idRol,
authority) VALUES (1, 'ROLE_ADMIN');
INSERT INTO `usuariosopinionesdbsec`.`authorities` (idRol,
authority) VALUES (2, 'ROLE_USER');

-- Table Opiniones
CREATE TABLE IF NOT EXISTS `Opiniones` (
  `idOpinion` INT NOT NULL AUTO_INCREMENT,
  `idUsuario` INT NOT NULL,
  `idPelicula` INT NOT NULL,
  `opinion` TEXT,
  `puntuacion` INT NOT NULL,
  PRIMARY KEY (`idOpinion`),
  UNIQUE INDEX `unique_Usuario_Pelicula` (`idUsuario` ASC,
`idPelicula` ASC) VISIBLE,
  INDEX `fk_Opiniones_Usuarios_idx` (`idUsuario` ASC) VISIBLE,
  CONSTRAINT `fk_Opiniones_Users`
    FOREIGN KEY (`idUsuario`)
    REFERENCES `Users` (`idUsuario`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Esquema 2: Películas y Actores (`peliculasactoresdb`)

Gestiona el catálogo audiovisual y las relaciones N:M.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `peliculasactoresdb` DEFAULT
CHARACTER SET utf8 COLLATE utf8_spanish_ci ;
USE `peliculasactoresdb`;

DROP TABLE IF EXISTS `peliculasactoresdb`.`Peliculas`,
`peliculasactoresdb`.`Actores`,
`peliculasactoresdb`.`Peliculas_y_actores` ;

-- Tabla de las películas
CREATE TABLE IF NOT EXISTS `peliculasactoresdb`.`Peliculas` (
  `idPelicula` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(150) NOT NULL,
  `año` YEAR NOT NULL,
  `duracion` INT(5) NOT NULL,
  `pais` VARCHAR(45) NOT NULL,
  `direccion` VARCHAR(100) NOT NULL,
  `genero` VARCHAR(45) NULL,
  `sinopsis`TEXT NULL,
  `imagen` TEXT NOT NULL,
  PRIMARY KEY (`idPelicula`))
ENGINE = InnoDB;

-- Tabla de los actores
CREATE TABLE IF NOT EXISTS `peliculasactoresdb`.`Actores` (
  `idActor` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NOT NULL,
  `fechaNacimiento` DATE NOT NULL,
  `pais` VARCHAR(45) NOT NULL,
  `imagen` TEXT NOT NULL,
  PRIMARY KEY (`idActor`))
ENGINE = InnoDB;

-- Tabla intermedia de peliculas y actores
CREATE TABLE IF NOT EXISTS
`peliculasactoresdb`.`Peliculas_y_actores` (
  `Peliculas_idPelicula` INT NOT NULL,
  `Actores_idActor` INT NOT NULL,
  PRIMARY KEY (`Peliculas_idPelicula`, `Actores_idActor`),
  INDEX `Peliculas_y_actores_Peliculas_idx`
(`Peliculas_idPelicula` ASC) VISIBLE,
```

```
  INDEX `Peliculas_y_actores_Actores_idx` (`Actores_idActor` ASC)
VISIBLE,
  CONSTRAINT `fk_pya_pelicula`
    FOREIGN KEY (`Peliculas_idPelicula`)
    REFERENCES `peliculasactoresdb`.`Peliculas` (`idPelicula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pya_actor`
    FOREIGN KEY (`Actores_idActor`)
    REFERENCES `peliculasactoresdb`.`Actores` (`idActor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

3. MANUAL DE USUARIO Y DESPLIEGUE

3.1. Guía de Ejecución del Proyecto

Para la correcta puesta en marcha del sistema, es necesario seguir un orden estricto de ejecución debido a las dependencias entre microservicios:

1. **Base de Datos:** Ejecutar los scripts SQL de creación de tablas en MySQL Workbench.
2. **Datos Iniciales:** Ejecutar los scripts de inserción de datos (proporcionados en el punto 3.4) para poblar las bases de datos.
3. **Eureka Server:** Iniciar el microservicio del servidor Eureka.
4. **Backends:** Iniciar los microservicios de "Películas/Actores" y "Usuarios/Opiniones".
5. **Gateway:** Iniciar el servidor Gateway.
6. **Cliente Web:** Iniciar el microservicio Cliente.
7. **Acceso:** Abrir un navegador y acceder a `http://localhost:8100`.

3.2. Navegación Pública (Sin Sesión)

La interfaz ha sido diseñada para ser intuitiva y visualmente atractiva:

- **Sección de Películas:** Muestra un *grid* visual con las portadas de las películas.

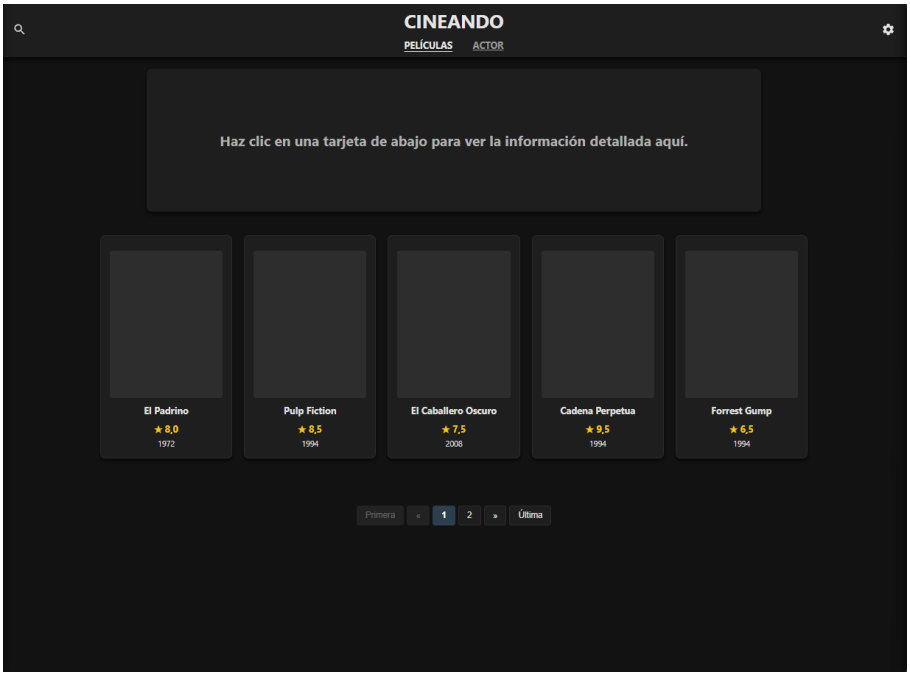


Ilustración 1. Página principal de películas

Al hacer clic en una tarjeta, la información detallada se despliega en la parte superior (sinopsis, director, etc.). Incluye paginación para navegar por el catálogo.

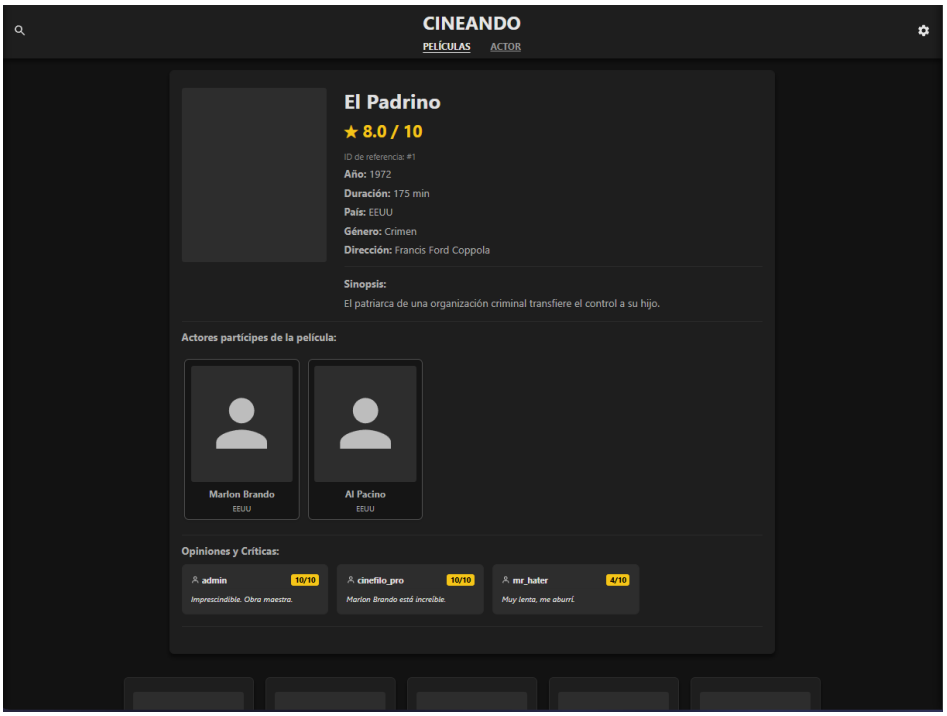


Ilustración 2. Página principal de películas (película seleccionada)

- **Búsqueda y Filtros:** En el lateral izquierdo existe un panel para filtrar películas por ID, título, género o ID del actor que participa en la película.

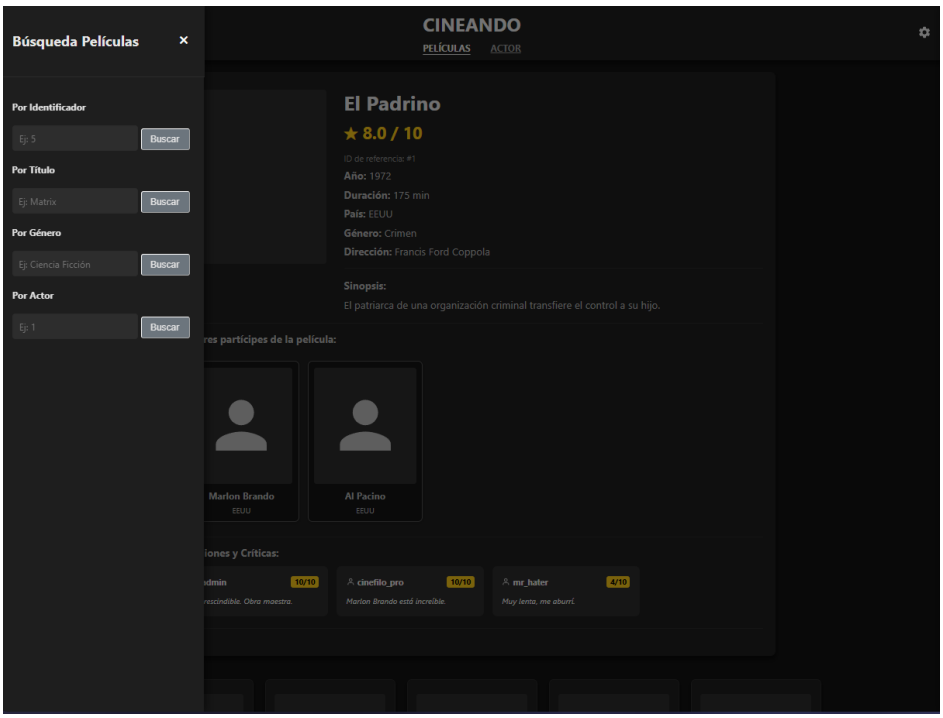


Ilustración 3. Filtros de búsqueda para películas

- **Sección de Actores:** Accesible desde el menú, presenta una interfaz similar orientada a los actores, con sus propios filtros de búsqueda.

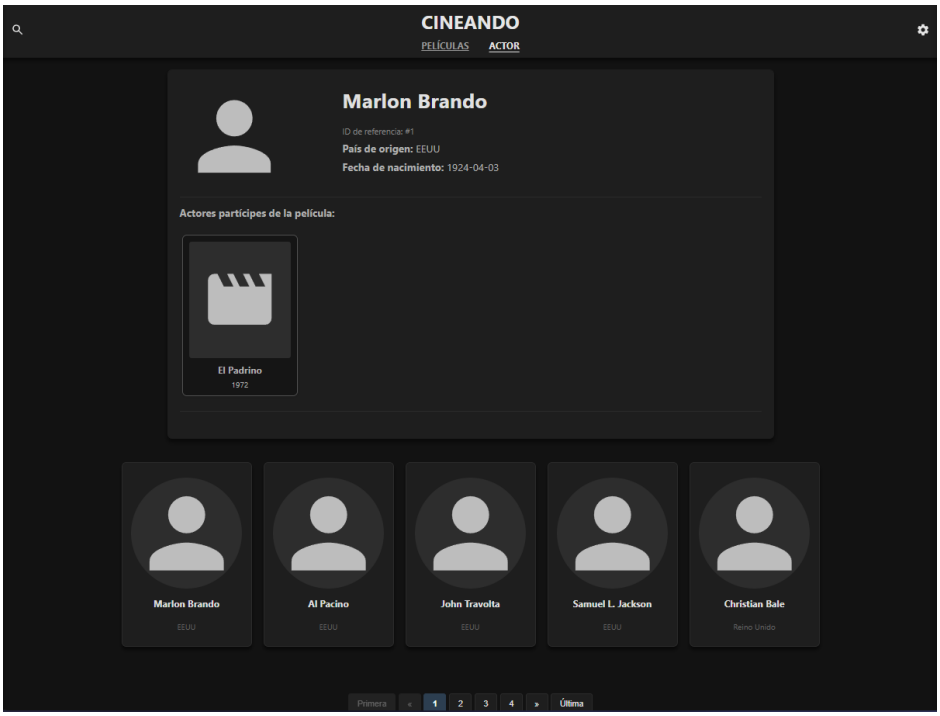


Ilustración 4. Página principal de actores

- **Ajustes Visuales:** Se dispone de una barra de ajustes que permite alternar entre **Modo Claro** y **Modo Oscuro** en tiempo real y acceder al login.

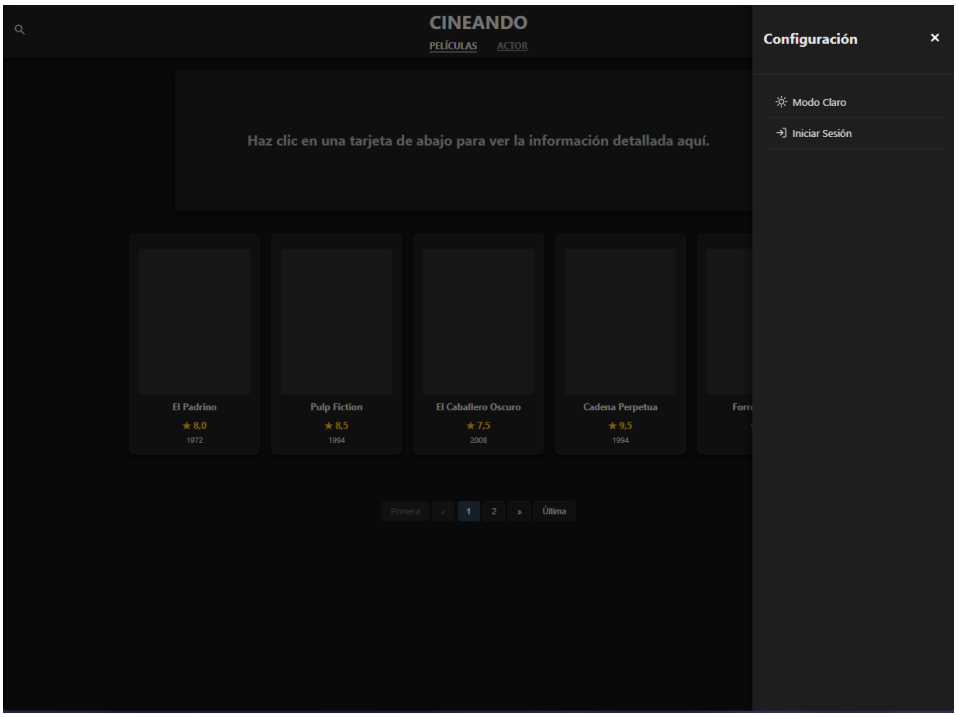


Ilustración 5. Barra lateral de ajustes (Modo oscuro)

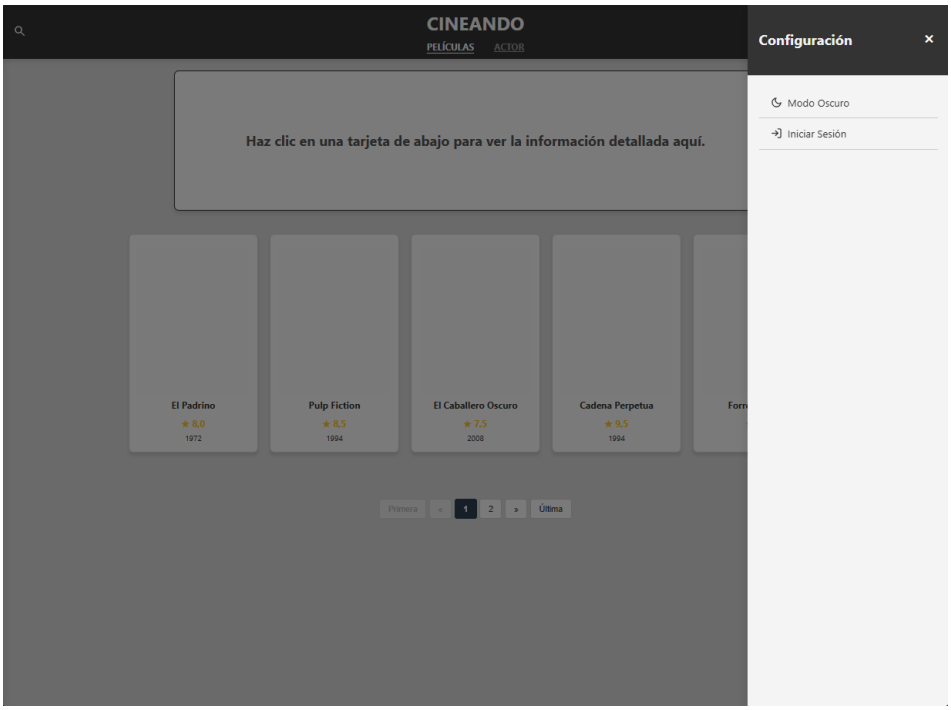


Ilustración 6. Cambio a modo claro

3.3. Gestión de Usuarios y Administración

Para acceder a las funciones avanzadas, el usuario debe autenticarse mediante la sección de inicio de sesión en la barra de ajustes:

- **Login y Registro:** Estas páginas permiten acceder a funciones de la página web exclusivas para usuarios y administradores (requiere de autenticación).

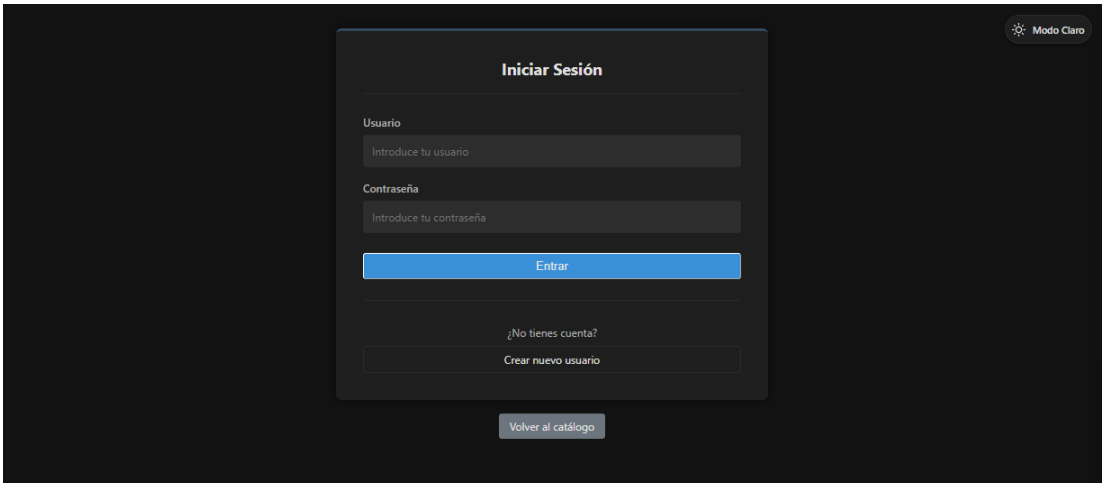


Ilustración 7. Página de login

Al registrarse, se puede asignar un rol (ADMIN o USER).

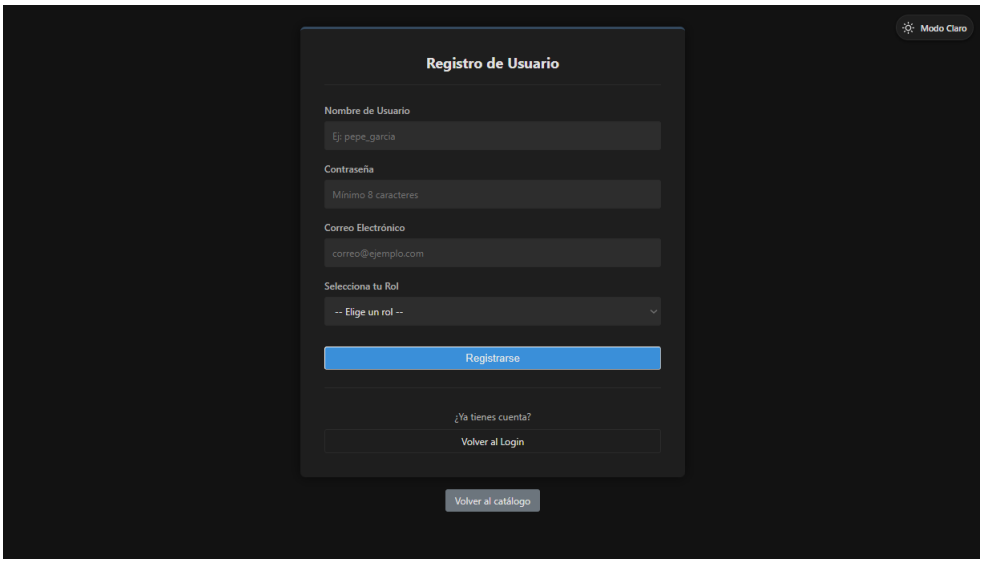


Ilustración 8. Página de registro

- **Rol de Usuario:**
    - Puede acceder a la ficha de una película y utilizar el formulario de opiniones para valorar (estrellas) y comentar la cinta. En la siguiente ilustración se puede ver el botón que permite acceder al formulario de votación.

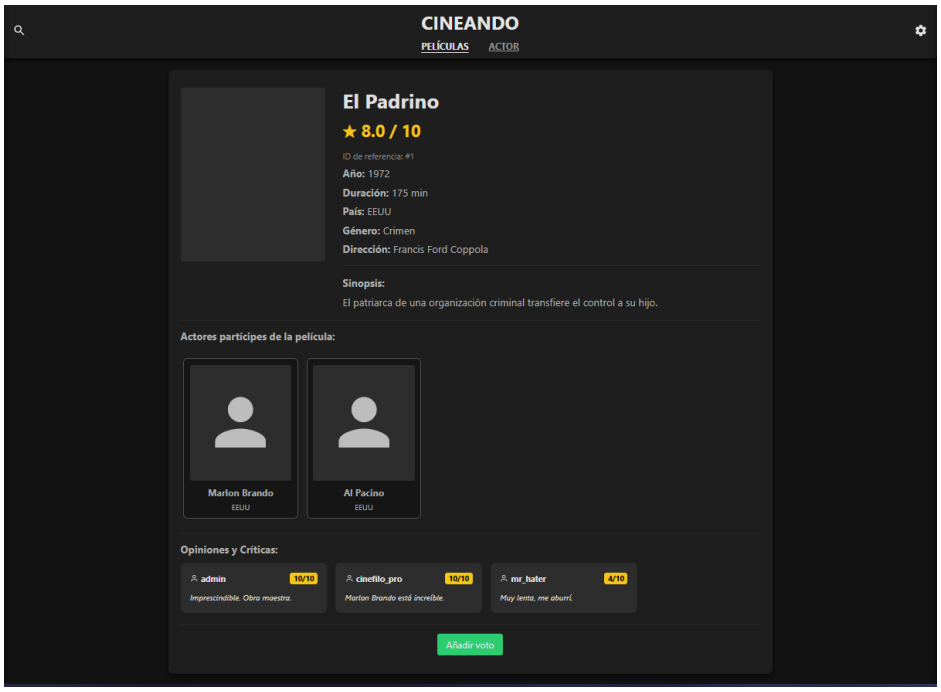


Ilustración 9. Botón para votar películas (Rol de usuario)

En esta ilustración se puede ver la página que permite realizar el voto de la película seleccionada.

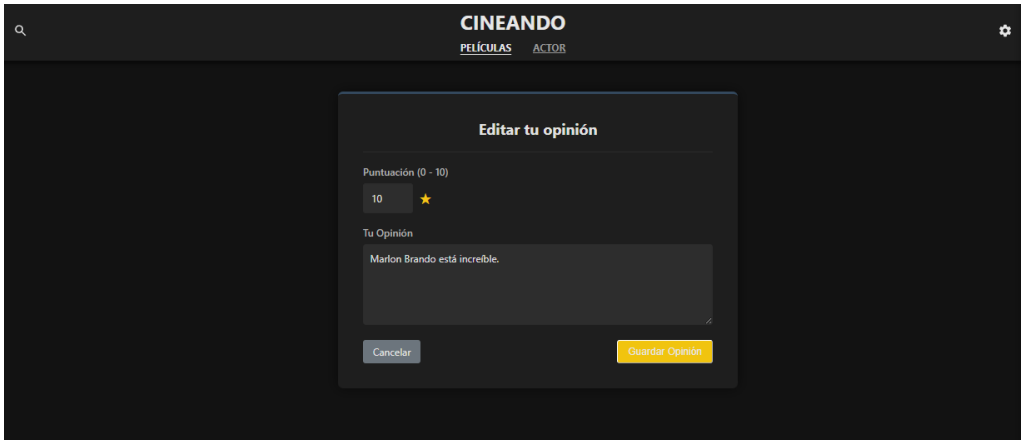


Ilustración 10. Página de votación de película

- **Rol de Administrador:**

  - Dispone de acceso integral a las herramientas de administración a través de la barra lateral de ajustes. Desde allí, gestiona usuarios y modera críticas, además de administrar el catálogo de películas y actores con permisos para crear, editar, eliminar y vincular fichas. En la siguiente imagen se pueden ver los botones de editar, borrar, cambiar relación con actores y añadir voto para la película deseada. Además de acceso a la gestión de usuarios y opiniones en la barra lateral.

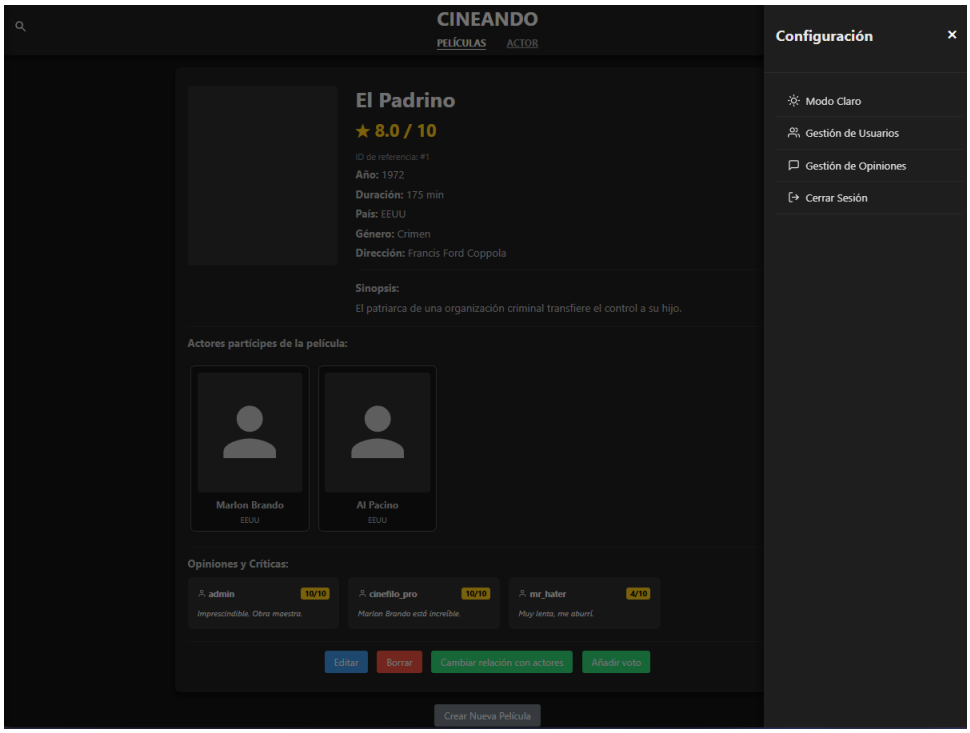


Ilustración 11. Página principal para administradores

En la siguiente ilustración se puede ver la página que da acceso a la edición y creación de películas.

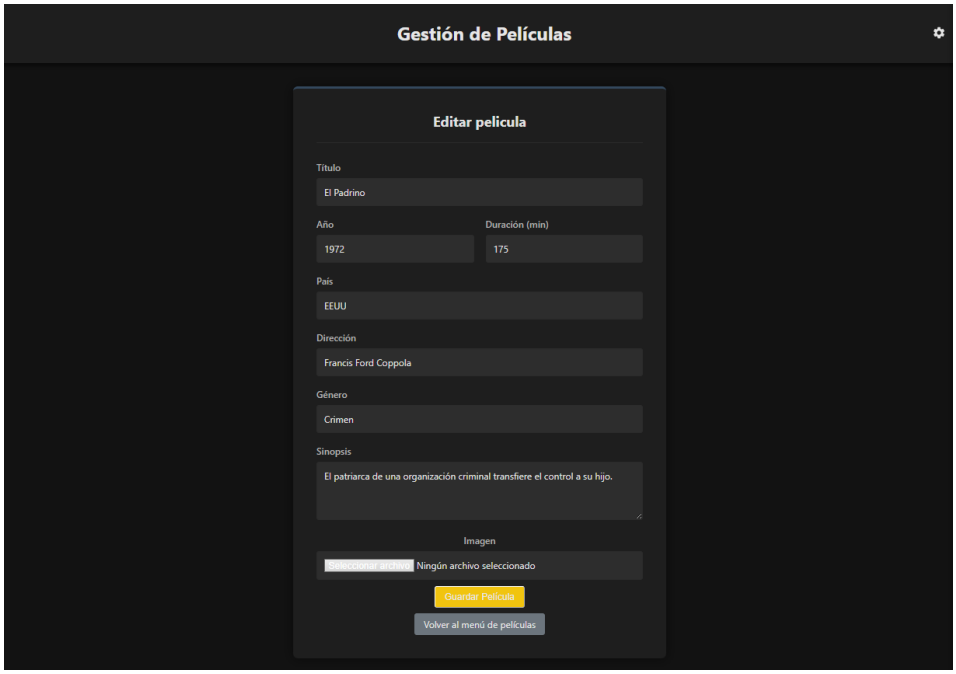


Ilustración 12. Página de creación/edición de películas

En la siguiente ilustración se puede ver la página que permite añadir o quitar los actores que participan en la película.

Ilustración 13. Página de cambio de relaciones entre películas y actores

En las siguientes ilustraciones se pueden ver las páginas que permiten gestionar los usuarios y opiniones existentes.

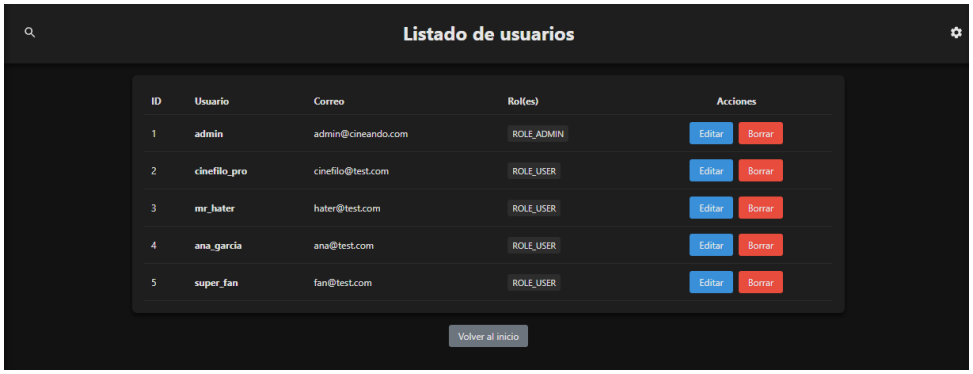


Ilustración 14. Página de gestión de usuarios

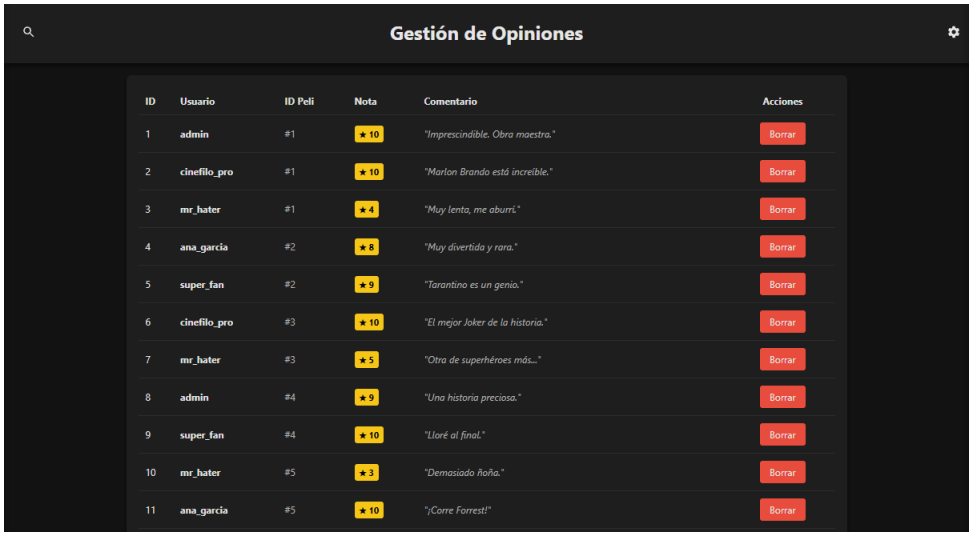
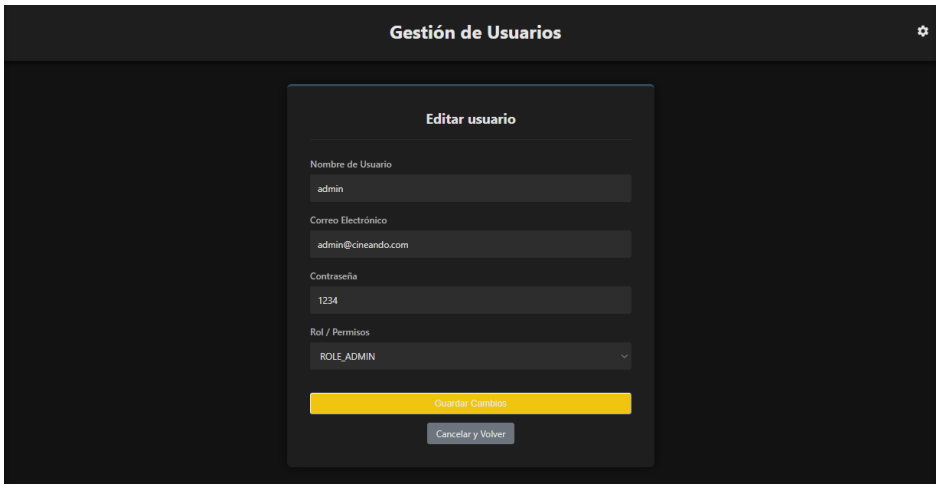


Ilustración 15. Página de gestión de opiniones

En la siguiente ilustración se puede ver la página que permite editar un usuario.



3.4. Datos de Prueba (Scripts de Inserción)

Para facilitar las pruebas, se adjuntan los scripts SQL necesarios para poblar el sistema con datos coherentes.

Datos para `peliculasactoresdb`:

```
USE `peliculasactoresdb`;
SET FOREIGN_KEY_CHECKS = 0;
TRUNCATE TABLE `Peliculas_y_actores`;
TRUNCATE TABLE `Actores`;
TRUNCATE TABLE `Peliculas`;

-- Insertar Películas
INSERT INTO `Peliculas` (`titulo`, `año`, `duracion`, `pais`,
`direccion`, `genero`, `sinopsis`, `imagen`) VALUES
('El Padrino', 1972, 175, 'EEUU', 'Francis Ford Coppola',
'Crimen', 'El patriarca de una organización criminal transfiere
el control a su hijo.', ''),
('Pulp Fiction', 1994, 154, 'EEUU', 'Quentin Tarantino',
'Crimen', 'Historias cruzadas de criminales en Los Ángeles.',
''),
('El Caballero Oscuro', 2008, 152, 'EEUU', 'Christopher Nolan',
'Acción', 'Batman se enfrenta al Joker, quien busca el caos en
Gotham.', ''),
('Cadena Perpetua', 1994, 142, 'EEUU', 'Frank Darabont', 'Drama',
'La historia de esperanza de dos hombres encarcelados.', ''),
('Forrest Gump', 1994, 142, 'EEUU', 'Robert Zemeckis', 'Drama',
'La vida de un hombre con bajo CI que presencia momentos
históricos.', ''),
('Inception', 2010, 148, 'EEUU', 'Christopher Nolan', 'Ciencia
Ficción', 'Un ladrón roba secretos corporativos a través del uso
de la tecnología de compartir sueños.', ''),
('Matrix', 1999, 136, 'EEUU', 'Hermanas Wachowski', 'Ciencia
Ficción', 'Un hacker descubre la verdadera naturaleza de su
realidad.', ''),
('El Señor de los Anillos: La Comunidad del Anillo', 2001, 178,
'Nueva Zelanda', 'Peter Jackson', 'Fantasía', 'Un hobbit debe
destruir un anillo poderoso para salvar el mundo.', ''),
('Interstellar', 2014, 169, 'EEUU', 'Christopher Nolan', 'Ciencia
Ficción', 'Un equipo de exploradores viaja a través de un agujero
de gusano en el espacio.', ''),
('Gladiator', 2000, 155, 'EEUU', 'Ridley Scott', 'Acción', 'Un
general romano traicionado busca venganza como gladiador.', '');

-- Insertar Actores
INSERT INTO `Actores` (`nombre`, `fechaNacimiento`, `pais`,
`imagen`) VALUES
('Marlon Brando', '1924-04-03', 'EEUU', ''),      -- ID 1
('Al Pacino', '1940-04-25', 'EEUU', ''),          -- ID 2
('John Travolta', '1954-02-18', 'EEUU', ''),    -- ID 3
```

```
('Samuel L. Jackson', '1948-12-21', 'EEUU', ''),  -- ID 4
('Christian Bale', '1974-01-30', 'Reino Unido', ''),-- ID 5
('Heath Ledger', '1979-04-04', 'Australia', ''),  -- ID 6
('Tim Robbins', '1958-10-16', 'EEUU', ''),       -- ID 7
('Morgan Freeman', '1937-06-01', 'EEUU', ''),    -- ID 8
('Tom Hanks', '1956-07-09', 'EEUU', ''),           -- ID 9
('Robin Wright', '1966-04-08', 'EEUU', ''),        -- ID 10
('Leonardo DiCaprio', '1974-11-11', 'EEUU', ''),     -- ID 11
('Joseph Gordon-Levitt', '1981-02-17', 'EEUU', ''),-- ID 12
('Keanu Reeves', '1964-09-02', 'Líbano', ''),      -- ID 13
('Laurence Fishburne', '1961-07-30', 'EEUU', ''),-- ID 14
('Elijah Wood', '1981-01-28', 'EEUU', ''),         -- ID 15
('Ian McKellen', '1939-05-25', 'Reino Unido', ''),-- ID 16
('Matthew McConaughey', '1969-11-04', 'EEUU', ''),-- ID 17
('Anne Hathaway', '1982-11-12', 'EEUU', ''),        -- ID 18
('Russell Crowe', '1964-04-07', 'Nueva Zelanda', ''),-- ID 19
('Joaquin Phoenix', '1974-10-28', 'Puerto Rico', ''); -- ID 20

-- Unir Actores con Películas
INSERT INTO `Peliculas_y_actores` (`Peliculas_idPelicula`,
`Actores_idActor`) VALUES
(1, 1), (1, 2),   -- El Padrino
(2, 3), (2, 4),   -- Pulp Fiction
(3, 5), (3, 6),   -- Batman
(4, 7), (4, 8),   -- Cadena Perpetua
(5, 9), (5, 10),  -- Forrest Gump
(6, 11), (6, 12), -- Inception
(7, 13), (7, 14), -- Matrix
(8, 15), (8, 16), -- Anillos
(9, 17), (9, 18), -- Interstellar
(10, 19), (10, 20); -- Gladiator

SET FOREIGN_KEY_CHECKS = 1;
```

Datos para `usuariosopinionesdbsec`:

```
USE `usuariosopinionesdbsec`;
SET FOREIGN_KEY_CHECKS = 0;
TRUNCATE TABLE `Opiniones`;
TRUNCATE TABLE `Users`;
DELETE FROM `Authorities` WHERE idRol > 0;
ALTER TABLE `Authorities` AUTO_INCREMENT = 1;

-- Insertar Roles
INSERT INTO `Authorities` (`authority`) VALUES ('ROLE_ADMIN'); --
1
INSERT INTO `Authorities` (`authority`) VALUES ('ROLE_USER');  --
2

-- Insertar Usuarios (Password '1234')
```

```
INSERT INTO `Users` (`username`, `password`, `correo`, `enable`,
`idRol`) VALUES
('admin', '1234', 'admin@cineando.com', 1, 1),
('cinefilo_pro', '1234', 'cinefilo@test.com', 1, 2),
('mr_hater', '1234', 'hater@test.com', 1, 2),
('ana_garcia', '1234', 'ana@test.com', 1, 2),
('super_fan', '1234', 'fan@test.com', 1, 2);

-- Insertar Opiniones
INSERT INTO `Opiniones` (`idUsuario`, `idPelicula`, `opinion`,
`puntuacion`) VALUES
(1, 1, 'Imprescindible. Obra maestra.', 10),
(2, 1, 'Marlon Brando está increíble.', 10),
(3, 1, 'Muy lenta, me aburrí.', 4),
(4, 2, 'Muy divertida y rara.', 8),
(5, 2, 'Tarantino es un genio.', 9),
(2, 3, 'El mejor Joker de la historia.', 10),
(3, 3, 'Otra de superhéroes más...', 5),
(1, 4, 'Una historia preciosa.', 9),
(5, 4, 'Lloré al final.', 10),
(3, 5, 'Demasiado ñoña.', 3),
(4, 5, '¡Corre Forrest!', 10),
(1, 6, 'Visualmente perfecta.', 9),
(2, 6, 'Un lío mental maravilloso.', 8),
(5, 7, 'Revolucionaria.', 10),
(4, 7, 'Keanu Reeves guapísimo.', 8),
(1, 10, 'Épica.', 9),
(3, 10, 'Solo peleas y sangre.', 4),
(5, 10, 'Fuerza y Honor.', 10);

SET FOREIGN_KEY_CHECKS = 1;
```

4. CONCLUSIONES

La realización de este proyecto ha permitido transformar una gestión básica de datos en una plataforma completa y moderna basada en microservicios. Se han logrado cambios visuales muy significativos respecto a la práctica inicial, abandonando la gestión clásica basada en tablas simples para ofrecer una experiencia de usuario rica, con visualización de portadas tipo *grid* y detalles interactivos.

La implementación de Spring Security y la separación de bases de datos han conseguido una mayor robustez de la aplicación, demostrando cómo una arquitectura desacoplada facilita la escalabilidad y el mantenimiento del código en entornos de desarrollo ágil.