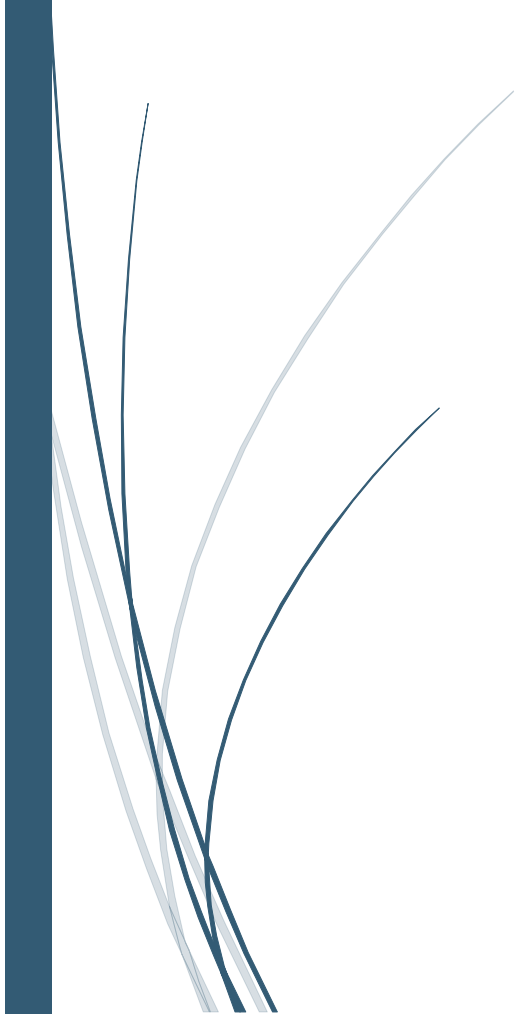
Práctica Final

Arquitectura y diseño de sistemas web y cliente/servidor



Contenido

Análisis del Problema

Análisis superficial del proyecto a realizar

Respecto al análisis, se nos presenta una potencial aplicación web con 2 partes intrínsecamente conectadas.

- La **parte del administrador**, que implica funciones cruciales para el control y supervisión efectiva de las operaciones del cine. El administrador tiene la capacidad de gestionar las entidades clave del sistema, como películas, salas y entradas. Puede realizar operaciones de inserción, eliminación, modificación y consulta en estas entidades, lo que permite mantener actualizada la base de datos y la programación de proyecciones. Además, la gestión de reservas y la generación de informes proporcionan herramientas valiosas para evaluar el rendimiento del cine. La autenticación y validación de sesión garantizan la seguridad y el acceso autorizado a estas funciones administrativas, asegurando que solo personal autorizado pueda realizar cambios y consultar información crítica. El administrador es una parte fundamental para el buen funcionamiento y la toma de decisiones estratégicas (por potenciales problemas o cambios) dentro del cine.
- La parte del cliente se centra en otorgar a los clientes una experiencia personalizada y accesible a los usuarios, así como segura (para el usuario mismo y para las partes relacionadas con el administrador). Los clientes tienen la capacidad de registrarse en la plataforma, lo que les permite acceder a funcionalidades exclusivas como la realización de reservas y la contribución de comentarios, que no deben estar permitidas en caso de no tener una sesión activa (previamente necesario haberse registrado). La función de reserva ofrece a los clientes la posibilidad de seleccionar películas, fechas, horas y salas, con una representación gráfica de la sala para elegir los asientos deseados. Además, se debe facilitar un proceso de pago seguro, en este caso, solicitando un número de tarjeta para cargar el importe correspondiente. Tras una reserva exitosa, se proporciona al cliente un número de referencia para acceder al cine. Los clientes también pueden visualizar información técnica de las películas, así como leer las opiniones de otros usuarios y por ende formular sus propios comentarios. Este enfoque centrado en el usuario no solo optimiza la interacción del cliente con el sistema, sino que también contribuye a la construcción de una comunidad en línea en torno al cine, mejorando la fidelización de los espectadores.

Para poder otorgar esta experiencia, tanto al administrador como al cliente se debe implementar mediante un patrón MVC.

¿Por qué un patrón MVC?

Descripción del problema

El patrón MVC permite separar la lógica de negocio (Modelo), la presentación (Vista) y la gestión de la interacción del usuario con la página (Controlador). En nuestro caso, aplicado a una página relacionada con un cine (a nivel de gestión interno del administrador como el cliente) el modelo contendría la lógica para la gestión de datos (películas, salas, entradas), mientras que la vista se encargaría de la representación visual de la página; y el controlador manejaría las acciones del usuario (administrador o cliente) y la lógica de la aplicación.

Consideraciones

Los aspectos a considerar para tomar la solución de un patrón MVC es la separación de las partes principales que hacen al patrón MVC lo que es, es decir, el modelo, la vista y el controlador (lo hace mucho más sencillo de abordar), su escalabilidad y el mantenimiento, en caso de errores será mucho más sencillo de localizar y neutralizar. Así como la reutilización de componentes si quisiéramos usar este mismo modelo en otro tipo de vista, que en nuestro caso no aplica, ya que solo haremos el cine, pero es algo a considerar en un caso en el "mundo real".

Ventajas y Desventajas del patrón MVC

Ventajas:

- Separación de responsabilidades que facilita el mantenimiento y escalabilidad.
- Reutilización de componentes en potenciales futuros proyectos (no aplicable en nuestro caso a priori).
- Facilitar las pruebas y control de problemas (Quality Assurance o QA).

Desventajas:

- Puede suponer algo más complejo que otras soluciones más simples.
- Puede haber una sobrecarga inicial en la creación de capas (que es solucionable).

Implementación

La página web, al estar realizada en JAVA implementando el patrón MVC, está dividida en "Web Pages" (JSPs, JavaScript, Css), "Source Packages" (clases java con los modelos, controladores y las clases de los objetos), así como otros ficheros que contienen las dependencias necesarias y archivos para controlar la versión de las herramientas que implementa la página web (Maven, tomcat o glassfish, etc.). En este caso nos centraremos en explicar las "Web Pages" así como los "Source Packages".

Vista

La importancia en esta parte del código reside en varias partes, que son las que se explicarán.

Components

- Footer.jsp: Este código JSP representa la sección de pie de página de una página web. Incluye información institucional, enlaces de navegación, y un formulario para suscribirse a noticias. También carga scripts JavaScript para mejorar la interactividad y la funcionalidad de la página.
- Head.jsp: Este código JSP define la sección de encabezado de una página web, configurando metadatos y enlaces a hojas de estilo. Incluye el título "Cine Grupo 2" y carga archivos CSS para el diseño y la apariencia de la página.
- Header.jsp: Este código JSP define la sección de encabezado del contenido de una página web, que incluye un menú de navegación con opciones como "Inicio" y "Películas". Además, presenta un formulario de búsqueda y opciones de inicio de sesión, registro y cierre de sesión condicionadas según la autenticación del usuario, permitiendo el acceso a perfiles de administrador.
- Hero.jsp: Este JSP muestra un encabezado con el título "Películas" y una ruta de navegación.
- Login.jsp: Este JSP define un formulario de inicio de sesión emergente, solicitando al usuario su correo electrónico y contraseña, con opciones para recordar la sesión y recuperar la contraseña.
- Preloading.jsp: Este JSP define la sección de preloading (carga previa) de una página web, mostrando un logo y un indicador visual durante la carga del contenido.
- Signup.jsp: Este JSP define un formulario emergente de registro, solicitando al usuario su nombre, apellido, correo electrónico y contraseña, con validaciones de entrada. El formulario se envía a través de la acción "registro" mediante el método POST.

CSS y misceláneos

En cuanto a CSS hay varios componentes que se encargan del manejo de fuentes mediante librerías, así como diferentes estilos que se emplean tanto en la parte del administrador como en la del cliente. No se ahondará en que hace cada archivo ya que son componentes que, de forma resumida, se encargan del estilo de la página de una forma relativamente estándar.

Además hay diferentes carpetas en las que se contienen imágenes para su uso en diferentes partes de la página.

JavaScript

Hay 1 archivo JavaScript que se encarga de la parte del administrador, se encarga de mostrar un mensaje de tal forma que quede acorde con el estilo de la página durante 5 segundos. Posteriormente encontramos diferentes archivos JavaScript (Jquery, y sus plugins) que se encarga del manejo de muchas de las partes que estilizan la vista del cliente, son librerías implementadas.

Archivos JSP

- ConfirmacionPago.jsp: Este JSP confirma la realización exitosa de una reserva de cine, mostrando el número de referencia. La página incluye un enlace para regresar al inicio y utiliza archivos JavaScript para la funcionalidad adicional, como la carga previa y la mejora de la interfaz de usuario. La información de referencia se obtiene mediante el parámetro "referencia" de la solicitud HTTP.
- Crear-pelicula.jsp: Este JSP proporciona una interfaz para crear una nueva película. Importa clases Java para la manipulación de datos, muestra mensajes de error si los hay, y presenta un formulario con campos para información detallada de la película, incluyendo opciones desplegables para género y actores. También permite seleccionar actores mediante checkboxes y, al enviar el formulario, los datos se envían a través de una solicitud POST a "/crearPelicula". Además, incluye un enlace para regresar a la gestión de películas y utiliza estilos CSS para la presentación.
- Crear-proyeccion.jsp: Este JSP permite crear una proyección al proporcionar un formulario con opciones desplegables para seleccionar una película, una sala y definir la fecha y hora de la proyección. Importa clases Java para la manipulación de datos, maneja mensajes de error y utiliza estilos CSS para la presentación. Al enviar el formulario, los datos se envían a través de una solicitud POST a "/crearProyeccion". Además, incluye un enlace para regresar a la gestión de proyecciones.
- crearSala: Este JSP proporciona una interfaz para crear una sala de cine, presentando un formulario con campos para ingresar el nombre de la sala, el número de filas y columnas. Al enviar el formulario, los datos se envían a través de una solicitud POST a "/crearSala". Además, incluye un enlace para regresar a la gestión de salas y utiliza estilos CSS para la presentación.
- Editar-pelicula.jsp: Este JSP permite editar la información de una película. Proporciona un formulario prellenado con los datos actuales de la película, incluyendo nombre, sinopsis, página oficial, título original, género, nacionalidad, duración, año, distribuidora, director, otros datos, clasificación de edad, portada y actores. Los campos del formulario se llenan con la información actual de la película, y se pueden modificar. Al hacer clic en "Editar película", se envían los datos actualizados a través de una solicitud POST a "/editarPelicula/id" para realizar la edición. Además, incluye un enlace para regresar a la gestión de películas y utiliza estilos CSS para la presentación.
- Editar-proyeccion.jsp: Este JSP permite editar la información de una proyección. Presenta un formulario prellenado con los datos actuales de la proyección, incluyendo la película, la sala y la fecha y hora de la proyección. Los campos del formulario se llenan con la información actual de la proyección, y se pueden modificar. Al hacer clic en "Editar proyección", se envían los datos actualizados a través de una solicitud POST a "/editarProyeccion/id" para realizar la edición. Además, incluye un enlace para regresar a la gestión de proyecciones y utiliza estilos CSS para la presentación.
- editarSala.jsp: Este JSP permite editar la información de una sala. Presenta un formulario prellenado con los datos actuales de la sala, incluyendo el nombre de la sala, el número

de filas y el número de columnas. Los campos del formulario se llenan con la información actual de la sala y se pueden modificar. Al hacer clic en "Actualizar Sala", se envían los datos actualizados a través de una solicitud POST a "/editarSala/id" para realizar la edición. Además, incluye un enlace para regresar a la gestión de salas y utiliza estilos CSS para la presentación. Si no se encuentra la sala o hay un error al cargarla, muestra un mensaje indicando el problema.

- Gest-entradas.jsp: Este JSP gestiona y muestra las entradas de una proyección específica. Recupera la lista de entradas y el identificador de la proyección desde los atributos de la solicitud. Utiliza estos datos para generar una tabla HTML que muestra información sobre cada entrada, incluyendo su ID, fila, columna y estado (sin reserva, descartado o reservado). También proporciona enlaces para cambiar el estado de cada entrada (descartar, habilitar o quitar reserva). Además, muestra mensajes de error o éxito utilizando JavaScript y estilos CSS para la presentación. Si no hay entradas disponibles, la tabla estará vacía.
- Gest-informes.jsp: Este JSP gestiona la presentación de informes relacionados con películas. Utiliza JavaScript para mostrar mensajes de error o éxito. La página incluye informes sobre películas agrupadas por género, mostrando el nombre de cada película en tablas separadas por género. Además, presenta informes generales sobre otros aspectos, como el número total de cada tipo de informe, utilizando tablas para mostrar estos datos. La presentación se realiza utilizando estilos CSS, y los mensajes de error o éxito se gestionan dinámicamente mediante JavaScript. La información se recupera de los atributos de la solicitud y se muestra de manera estructurada y organizada en la página HTML.
- Gest-peliculas.jsp: Este JSP gestiona la presentación y gestión de películas. Utiliza JavaScript para mostrar mensajes de error o éxito. La página incluye una tabla que presenta información sobre las películas, como su ID, nombre, género, año y clasificación. Permite la creación de nuevas películas mediante un enlace "Crear película". Además, proporciona enlaces "Editar" y "Eliminar" para cada película en la tabla. La información se recupera de la base de datos a través de la capa de acceso a datos ('PeliculaDAO') y se presenta de manera estructurada en la página HTML. Los mensajes de error o éxito se gestionan dinámicamente mediante JavaScript.
- Gest-proyecciones.jsp: Este JSP gestiona la presentación y gestión de proyecciones cinematográficas. Utiliza JavaScript para mostrar mensajes de error o éxito. La página incluye una tabla que presenta información sobre cada proyección, como su ID, nombre de la película, nombre de la sala, fecha y hora, total de entradas y entradas vendidas. Además, proporciona enlaces "Gestionar Entradas", "Editar" y "Eliminar" para cada proyección en la tabla. La información se recupera de la capa de acceso a datos ('ProyeccionDAO') y se presenta de manera estructurada en la página HTML. Los mensajes de error o éxito se gestionan dinámicamente mediante JavaScript.
- Gest-reservas.jsp: Este JSP gestiona la presentación y gestión de reservas. Utiliza JavaScript para mostrar mensajes de error o éxito. La página incluye una tabla que presenta información sobre cada reserva, como su ID, referencia, precio, número de tarjeta y ID del cliente. La información se recupera de la capa de acceso a datos ('ReservaDAO') y se presenta de manera estructurada en la página HTML. Los mensajes de error o éxito se gestionan dinámicamente mediante JavaScript. La interfaz proporciona una opción para volver al menú principal.
- Gest-salas.jsp: Este JSP se encarga de la gestión de salas para administradores. Utiliza JavaScript para mostrar mensajes de error o éxito de forma dinámica. La página incluye una tabla que presenta información sobre cada sala, como su ID, nombre, filas y columnas. Se proporcionan enlaces para editar o eliminar cada sala, y hay opciones para crear una nueva sala o volver al menú principal. La interfaz está diseñada con un estilo definido por hojas de estilo CSS.

- Index.jsp: Este JSP representa una página principal que muestra una lista de películas con detalles y trailers. Se estructura en secciones como encabezado, carga previa, encabezado de página, lista de películas en un slider y trailers. La información de las películas se recupera del ámbito de la aplicación y se presenta en elementos visuales interactivos. Además, utiliza JavaScript para la carga dinámica de trailers y sigue un diseño moderno y atractivo.
- Login.jsp: Este JSP representa una página de inicio de sesión con un formulario HTML. Muestra mensajes de error o éxito, si los hay, y permite a los usuarios ingresar su correo electrónico y contraseña para iniciar sesión en la aplicación. Utiliza CSS para el estilo y se comunica con el servidor a través de una solicitud POST al endpoint "/login".
- operacionesAdmin.jsp: Este JSP verifica si el usuario actual en sesión es un administrador. Si no lo es, redirecciona al inicio de sesión. Si es un administrador, muestra un menú de administrador con enlaces a diversas secciones, como la gestión de películas, salas, proyecciones, reservas e informes. También proporciona un enlace para cerrar sesión. Utiliza CSS para el estilo y enlaces para la navegación entre las diferentes secciones de administración.
- Pago.jsp: Este JSP crea una página de detalles de pago para una reserva de entradas. Incluye un formulario de pago ficticio con campos para el nombre del titular, número de tarjeta, fecha de expiración y código de seguridad. Además, muestra detalles de la reserva, como el número de entradas, el precio por entrada y el precio total. Utiliza JavaScript para la interactividad y carga de plugins. Al enviar el formulario, se procesarán los detalles de la reserva en el servlet correspondiente.
- Película.jsp: Este JSP crea la página de detalles de una película, mostrando su información, actores, sinopsis y comentarios. Utiliza etiquetas de inclusión para componentes comunes como encabezado, preloading, inicio de sesión y registro. Presenta un diseño responsivo con secciones para ver el tráiler, hacer reservas y dejar comentarios. Además, incluye detalles específicos de la película como el reparto, director, género, año de lanzamiento, duración, nacionalidad, distribuidora y otros datos. Permite a los usuarios autenticados dejar comentarios y muestra la cantidad total de comentarios.
- Películas.jsp: Este JSP presenta una lista de películas obtenida desde el ámbito de la aplicación. Cada película muestra su imagen, nombre, año, sinopsis, duración y director. El diseño es responsivo, con detalles adicionales como el número total de películas y un anuncio en la barra lateral. Se incluyen componentes comunes como encabezado, preloading, inicio de sesión y registro.
- Perfil.jsp: Este JSP es un place holder al que redirigir mediante los controladores para derivar al usuario a su correspondiente vista.
- Reserva.jsp: Este JSP gestiona la visualización y reserva de proyecciones de películas. Muestra detalles de una película, como su nombre, año, clasificación, y fechas disponibles para proyecciones. Utiliza una estructura de pestañas para alternar entre la información general y las fechas disponibles. Cada fecha de proyección incluye detalles como la sala, fecha y hora. Además, presenta un diseño responsivo y un modal interactivo para seleccionar y reservar asientos. Se utilizan scripts JavaScript para cargar dinámicamente los asientos y gestionar la reserva.
- salaCine.jsp: Este JSP representa la interfaz de selección de asientos en un cine. Muestra una pantalla, filas de asientos y permite al usuario seleccionar asientos haciendo clic en ellos. Utiliza JavaScript para cambiar el estado visual de los asientos seleccionados. También incluye un botón de reserva y un enlace para volver a la página de inicio.
- signup.jsp: Este JSP crea una página web para que los usuarios creen una cuenta. Incluye un formulario con campos para nombre, apellido, correo electrónico y contraseña. Además, muestra cualquier mensaje de error almacenado en la sesión y utiliza CSS para el estilo de la página.

Controladores

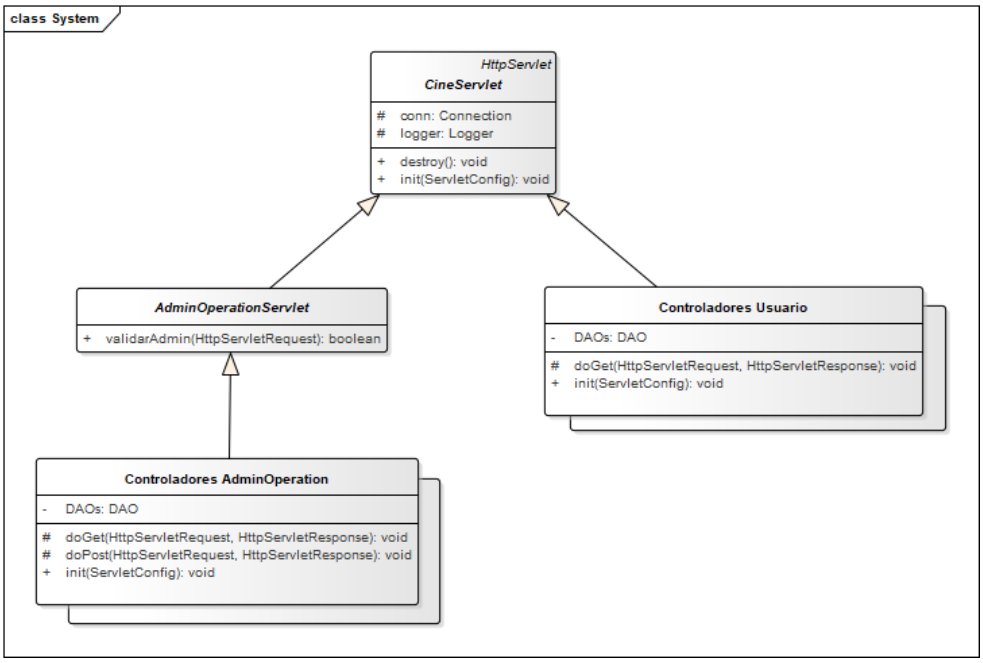
Para empezar, deberemos tener en cuenta que cada controlador tendrá que emplear una conexión a la base de datos (cada servlet, una conexión). Y que para tener registro dentro de lo que pasa dentro de cada servlet, cada uno de los controladores tendrá que emplear un objeto de tipo logger, identificado por su nombre. Para no tener que repetir todos estos atributos dentro de cada clase, se ha empleado la clase ABSTRACTA **CineServlet**. Esta clase tiene como atributos la conexión a la base de datos y el objeto de tipo logger.

Como sabemos, los métodos init() y destroy() son aquellas funciones que serán ejecutadas al iniciar o destruirse el servlet. Por lo que el método init(), es el encargado de iniciar la conexión a la base de datos, empleando la clase DatabaseConnection del DAO y de iniciar el logger. Y el método destroy() será el encargado de cerrar esta conexión y así asegurar que no haya conflictos al terminar con la vida del servidor.

Para la parte del administrador, se ha creado otra clase ABSTRACTA **AdminOperationServlet** que es una clase que hereda de la clase abstracta CineServlet (por lo que incluye todos los métodos que usa esta clase) y tiene dentro la función validarAdmin() que tiene la finalidad de validar si el usuario que tienen o no la sesión es de tipo Administrador. Esta clase se utiliza para evitar la repetición de código al validar al administrador en los controladores de sus operaciones.

Por lo que los controladores de los administradores heredaran de la clase abstracta AdminOperationServlet y los demás controladores de CineServlet. Cada controlador, tendrá uno o muchos DAOs como atributo, por lo que se tendrá que modificar el método init del servidor para poder inicializarlos y asignarles la conexión del controlador que tiene a cada uno de ellos. Primero se llamará al método init() del padre (CineServlet.init()) ya que inicializa la conexión y después se realizará la inicialización de los DAOs que utilice.

Por lo que nos quedaría un diagrama de clases de la siguiente manera:



A continuación, explicaremos cada uno de los controladores separados por los generales, orientados al cliente y las operaciones del administrador, donde solo podrán acceder administradores.

Operaciones del cliente

- **CerrarSesion.java:** es un servlet que gestiona el cierre de sesión. Al recibir una solicitud GET, verifica si hay una sesión activa y un atributo de usuario, y si es así, elimina el atributo

de usuario de la sesión y redirige al usuario a la página de inicio. También tiene un método doPost que llama a 'processRequest', el cual genera una página HTML de bienvenida.

- **ComentarioServlet.java:** servlet que maneja solicitudes POST para agregar comentarios a películas. El servlet verifica la existencia de un usuario en la sesión y obtiene los parámetros del formulario, como el ID de la película y el texto del comentario. Luego, crea un objeto 'Comentario', establece sus atributos y utiliza un objeto 'ComentarioDAO' para almacenar el comentario en la base de datos. Si el comentario se registra con éxito, redirige a la página de la película correspondiente. En caso de error, redirige a una página de error (404.jsp). Además, registra mensajes informativos o de advertencia en el registro de la aplicación.
- **EntradasServlet.java:** 'EntradasServlet' es un servlet que maneja solicitudes GET para obtener información sobre las entradas de una proyección. El servlet recupera el ID de la proyección de los parámetros de la solicitud, accede a la base de datos para obtener la información de las entradas asociadas a esa proyección y devuelve la información en formato JSON al cliente. El servlet maneja posibles errores, como la falta de un ID de proyección en la solicitud, un ID de proyección no válido o problemas al acceder a la base de datos. Además, registra mensajes de error en el registro de la aplicación en caso de problemas con la conexión a la base de datos.
- **LoginController.java:** es un servlet que maneja las solicitudes GET y POST para el inicio de sesión. Al recibir una solicitud GET, verifica si hay una sesión de usuario activa y redirige al perfil del usuario si es así, o a la página de inicio de sesión ('login.jsp') si no hay sesión. Al recibir una solicitud POST, valida las credenciales del usuario, establece una sesión en caso de éxito y redirige al perfil del usuario, o vuelve a la página de inicio de sesión con un mensaje de error si las credenciales son incorrectas. Utiliza la clase 'ClienteDAO' para interactuar con la base de datos y gestiona las sesiones con 'HttpSession'.
- **PeliculaServlet.java:** es un servlet que maneja solicitudes GET para mostrar detalles de una película. Recupera el ID de la película de la solicitud, accede a la base de datos a través de 'PeliculaDAO' para obtener la información de la película y la coloca en el ámbito de solicitud para que la página JSP pueda acceder a ella. Dependiendo de la página de origen, redirige a la página de reserva ('reserva.jsp') si se solicita desde allí y hay una sesión de usuario válida, o simplemente muestra la información de la película en la página 'pelicula.jsp'. Si la película no se encuentra o hay errores, envía respuestas de error correspondientes.
- **PeliculasServlet.java:** servlet que maneja solicitudes GET para mostrar todas las películas. A accede a la base de datos a través de 'PeliculaDAO' para obtener la lista de todas las películas y coloca esta lista en el ámbito de solicitud para que la página JSP pueda accederla. En caso de errores al obtener las películas o al interactuar con la base de datos, registra el error y envía una respuesta de error correspondiente. Finalmente, cierra la conexión a la base de datos en un bloque 'finally' para garantizar su cierre. La página JSP a la que redirige es 'peliculas.jsp'.
- **PerfilController.java:** es un servlet que gestiona las solicitudes GET para mostrar el perfil del usuario. En el método 'doGet', verifica si hay un usuario autenticado almacenado en la sesión. Si hay un usuario, redirige a diferentes páginas según si el usuario es administrador o no. Si no hay un usuario autenticado, redirige a la página de inicio de sesión. El método 'doPost' llama al método 'processRequest', que genera una respuesta HTML.
- **ReservaServlet.java:** El servlet maneja las solicitudes relacionadas con la creación de reservas para proyecciones. En el método 'doPost', obtiene información del formulario de pago, como nombre, número de tarjeta, fecha de expiración, código de seguridad, asientos seleccionados, y precio total. Luego, crea una nueva reserva y la almacena en la base de datos. El número de referencia de la reserva se genera aleatoriamente, y se

actualiza el atributo 'reserva_id' de las entradas seleccionadas. Después, redirige a una página de confirmación de pago o a una página de error, según el resultado de la operación.

- **SalaServlet.java:** El servlet maneja solicitudes relacionadas con la obtención de información sobre salas. En el método 'doGet', recupera el ID de la sala de los parámetros de la solicitud, accede a la base de datos para obtener la información detallada de esa sala y devuelve los datos en formato JSON como respuesta.
- **SignupController.java:** gestiona el registro de nuevos usuarios. En el método 'doGet', verifica si ya hay un usuario autenticado y redirige al perfil si es así; de lo contrario, redirige al usuario a la página de registro. En el método 'doPost', recoge los datos del formulario de registro, verifica la disponibilidad del correo electrónico, y procede a insertar el nuevo cliente en la base de datos. Después, redirige al usuario a la página de inicio de sesión con un mensaje de éxito o error.

Operaciones del Administrador

- **AdminOperationServlet.java:** extiende otra clase llamada 'CineServlet'. Esta clase proporciona un método 'validarAdmin' que toma una solicitud HTTP como parámetro, obtiene la sesión asociada con la solicitud y verifica si el usuario almacenado en la sesión (atributo "usuario") es un objeto de la clase 'Cliente' y si este cliente tiene el rol de administrador. Si la sesión no existe o el usuario no es un administrador, el método devuelve 'false'; de lo contrario, devuelve 'true'. Esta clase sirve como base para servlets que realizan operaciones administrativas, asegurando que solo los administradores tengan acceso a estas operaciones.
- **CambiarEntradaController.java:** extiende la clase 'AdminOperationServlet' y maneja las solicitudes GET para cambiar el estado de una entrada en la base de datos. El servlet valida si el usuario es un administrador. Si es así, extrae el identificador de la entrada de la ruta de la solicitud, recupera la entrada de la base de datos, y luego actualiza su estado en función de si tiene una reserva o no. Finalmente, redirige a la página de gestión de proyecciones con un mensaje de éxito o error, según corresponda.
- **CrearPeliculaController.java:** Procesa solicitudes GET y POST para la creación de nuevas películas en la base de datos. En el método 'doGet', se valida si el usuario es un administrador y se establece una lista de géneros antes de redirigir a la página de creación de películas. En el método 'doPost', se obtienen los parámetros del formulario, se crea un objeto Película, se inserta en la base de datos y se actualiza la lista de películas en el ámbito de la aplicación. Luego, se redirige a la página de gestión de películas con mensajes de éxito o error.
- **CrearProyeccionController.java:** Procesa solicitudes GET y POST para la creación de nuevas proyecciones de películas en la base de datos. En el método 'doGet', se valida si el usuario es un administrador antes de redirigir a la página de creación de proyecciones. En el método 'doPost', se obtienen los parámetros del formulario, se crea un objeto 'Proyeccion', se inserta en la base de datos junto con las entradas correspondientes, y se redirige a la página de gestión de proyecciones con mensajes de éxito o error.
- **CrearSalaController.java:** maneja solicitudes GET y POST relacionadas con la creación de salas de cine. En el método 'doGet', se valida si el usuario es un administrador antes de redirigirlo a la página de creación de salas. En el método 'doPost', se obtienen los parámetros del formulario, como el nombre de la sala, el número de filas y columnas, y se utiliza un objeto 'SalaDAO' para insertar estos datos en la base de datos. Luego, se establecen atributos de sesión según si la operación fue exitosa o no, y se redirige a la página de gestión de salas.

- **EditarPeliculaController.java:** Gestiona solicitudes GET y POST relacionadas con la edición de información de películas. En el método 'doGet', se valida si el usuario es un administrador antes de permitir la edición. Luego, obtiene el ID de la película desde la URL, valida el ID y obtiene los datos de la película desde la base de datos para mostrarlos en un formulario de edición. En el método 'doPost', se obtienen los datos actualizados del formulario y se utiliza un objeto 'PeliculaDAO' para actualizar la información de la película en la base de datos. Se manejan las excepciones de SQL y se actualiza la lista de películas en la aplicación.
- **EditarProyeccionController.java:** maneja solicitudes GET y POST relacionadas con la edición de información de proyecciones. En el método 'doGet', se valida si el usuario es un administrador antes de permitir la edición. Luego, obtiene el ID de la proyección desde la URL, valida el ID y obtiene los datos de la proyección desde la base de datos para mostrarlos en un formulario de edición. En el método 'doPost', se obtienen los datos actualizados del formulario y se utiliza un objeto 'ProyeccionDAO' para actualizar la información de la proyección en la base de datos. Se manejan las excepciones de SQL y se redirige a la página de gestión de proyecciones con un mensaje de éxito.
- **EditarSalaController.java:** gestiona las solicitudes GET y POST relacionadas con la edición de salas. En el método 'doGet', se valida si el usuario es un administrador antes de permitir la edición. Luego, obtiene el ID de la sala desde la URL y recupera los detalles de la sala desde la base de datos para mostrarlos en un formulario de edición. En el método 'doPost', se valida nuevamente la autorización del administrador, se obtienen los datos actualizados del formulario y se utiliza un objeto 'SalaDAO' para actualizar la información de la sala en la base de datos. Se manejan las excepciones de SQL y se redirige a la página de gestión de salas con un mensaje de éxito o error.
- **EliminarPeliculaController.java:** emplea solicitudes GET para eliminar películas. Valida la autorización del administrador, elimina la película según el ID proporcionado y actualiza la lista de películas. En caso de éxito, redirige a la página de gestión de películas con un mensaje positivo; en caso de error, redirige con un mensaje de error.
- **EliminarProyeccionController.java:** es un servlet Java que maneja solicitudes GET para eliminar proyecciones. Verifica la autorización del administrador, elimina la proyección según el ID proporcionado y redirige a la página de gestión de proyecciones. En caso de éxito, muestra un mensaje de éxito; en caso de error, muestra un mensaje de error y redirige nuevamente a la página de gestión de proyecciones.
- **EliminarSalaController.java:** utiliza solicitudes GET para eliminar salas. Verifica la autorización del administrador, extrae el ID de la sala de la URL, elimina la sala correspondiente y redirige a la página de gestión de salas. Si la eliminación tiene éxito, muestra un mensaje de éxito; en caso de error, muestra un mensaje de error y redirige nuevamente a la página de gestión de salas.
- **GestEntradasProyController.java:** es un servlet que maneja solicitudes GET para mostrar las entradas asociadas a una proyección. Verifica la autenticación del administrador, obtiene las entradas de la base de datos y las muestra en una página JSP ('gest-entradas.jsp').
- **GestInformesController.java:** es un servlet que maneja solicitudes GET para generar informes relacionados con películas, proyecciones y salas. Verifica la autenticación del administrador, recupera datos de películas, proyecciones y salas desde la base de datos, y genera informes que incluyen el número de entradas totales y vendidas, el número de películas y el número de salas. Luego, presenta estos informes en una página JSP ('gest-informes.jsp').
- **GestPeliculasController.java:** servlet que maneja solicitudes GET para la gestión de películas. Verifica la autenticación del administrador y redirige a la página JSP ('gest-peliculas.jsp') para la gestión de películas si el usuario es un administrador. Si el usuario no es un administrador, lo redirige a la página de inicio ('index.jsp').

- **GestProyeccionesController.java:** servlet que maneja solicitudes GET para la gestión de proyecciones en. Verifica la autenticación del administrador, obtiene la lista de proyecciones y redirige a la página JSP ('gest-proyecciones.jsp') si el usuario es un administrador; de lo contrario, lo redirige a la página de inicio ('index.jsp').
- **GestReservasController.java:** servlet que maneja solicitudes GET para la gestión de reservas. Verifica la autenticación del administrador, obtiene la lista de todas las reservas y redirige a la página JSP ('gest-reservas.jsp') si el usuario es un administrador; de lo contrario, lo redirige a la página de inicio ('index.jsp'). La lista de reservas se almacena en el atributo "reservas" para ser utilizada en la página JSP.
- **GestSalasController.java:** servlet que maneja solicitudes GET para la gestión de salas. Verifica la autenticación del administrador, obtiene la lista de todas las salas y redirige a la página JSP ('gest-salas.jsp') si el usuario es un administrador. La lista de salas se almacena en el atributo "salas" en la sesión para ser utilizada en la página JSP. Si el usuario no es un administrador, lo redirige a la página de inicio ('index.jsp').

Modelo

DAO

- ActorDAO.java: El código Java define una clase 'ActorDAO' que interactúa con la base de datos. Proporciona métodos para obtener todos los actores, añadir actores a una película, obtener actores asociados a una película y eliminar actores de una película. Utiliza operaciones SQL con 'PreparedStatement' y se conecta a la base de datos mediante la clase 'DatabaseConnection'.
- ClienteDAO.java: La clase 'ClienteDAO' en Java gestiona la interacción con la base de datos para la entidad Cliente. Proporciona métodos para obtener una conexión a la base de datos, validar el inicio de sesión de un usuario, insertar nuevos clientes y verificar la existencia de un correo electrónico en la base de datos. Utiliza operaciones SQL con 'PreparedStatement' y manejo de excepciones para gestionar errores de conexión y consulta.
- ComentarioDAO.java: El código pertenece a la clase 'ComentarioDAO' en Java, que se encarga de interactuar con la base de datos para gestionar comentarios. El método 'agregarComentario' recibe un objeto de tipo 'Comentario' y lo inserta en la base de datos utilizando una sentencia SQL preparada. Registra información sobre el éxito o fracaso de la operación en los registros de registro (logs) y devuelve un booleano indicando si la inserción fue exitosa o no. La clase utiliza manejo de excepciones para gestionar posibles errores durante la ejecución de la consulta SQL.
- DatabaseConnection.java: Se encarga de establecer la conexión con la base de datos Derby. Al crearse una instancia de la clase, se registra el controlador JDBC de Derby y se intenta establecer una conexión con la base de datos utilizando la URL proporcionada. La conexión se mantiene como un miembro estático de la clase y puede ser obtenida mediante el método 'getConnection()'. En caso de errores durante el proceso de conexión, se registran mensajes de error en los registros de registro (logs) utilizando el mecanismo de registro de la biblioteca Java 'java.util.logging'.
- EntradaDAO: Proporciona métodos para realizar operaciones relacionadas con las entradas en una base de datos.
- PeliculaDAO.java: Es responsable de gestionar la interacción con la base de datos para las operaciones relacionadas con películas en un sistema de gestión de cine. Proporciona funciones para insertar, actualizar, eliminar y recuperar información sobre películas. Además, incluye métodos para obtener detalles como actores, comentarios y proyecciones asociadas a una película específica. La clase utiliza JDBC para conectarse a una base de

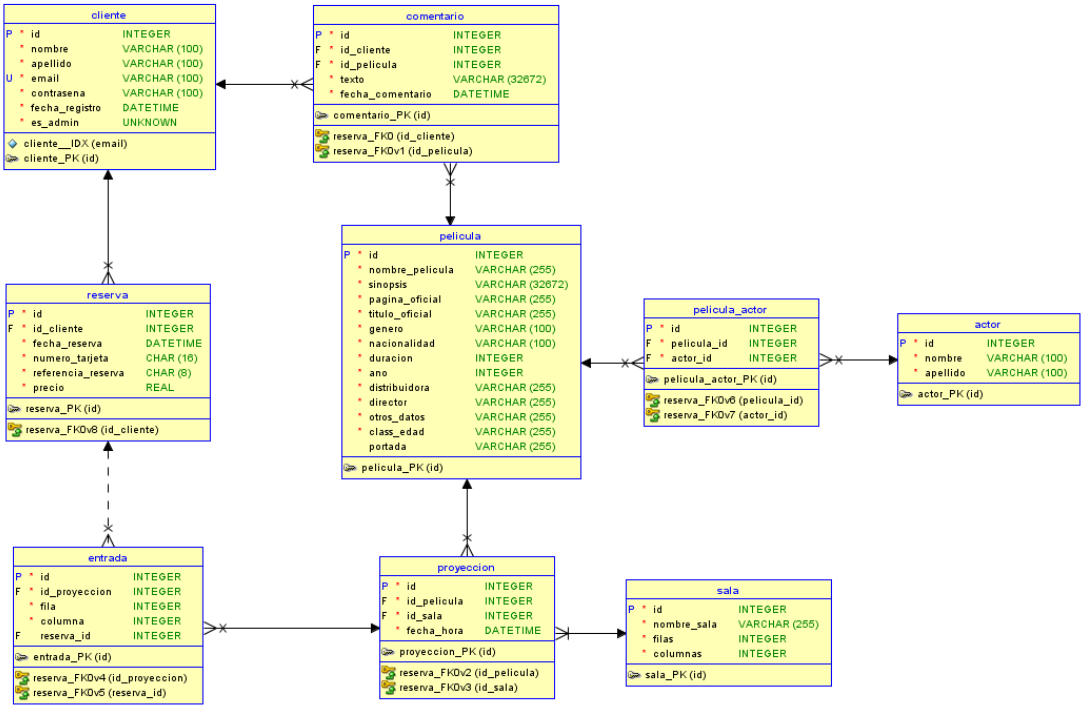
datos Derby, maneja excepciones y registra eventos en logs. También colabora con la clase 'ActorDAO' para gestionar la información de los actores asociados a las películas.

- ProyeccionDAO.java: Gestiona la interacción con la base de datos para realizar operaciones relacionadas con proyecciones en un sistema de gestión de cine. Proporciona métodos para obtener todas las proyecciones, crear una nueva proyección, obtener el ID de una proyección, eliminar una proyección, validar la existencia de una proyección por su ID, obtener detalles de una proyección por su ID, editar una proyección y obtener información sobre las proyecciones y las entradas vendidas.
- ReservaDAO.java: Administra operaciones de la base de datos relacionadas con reservas. Contiene los métodos para obtener todas las reservas existentes y crear nuevas reservas. El método 'all' consulta y retorna todas las reservas existentes. El método 'crearReserva' inserta una nueva reserva y devuelve su ID generado, utilizando PreparedStatement y manejo de excepciones SQL con registros en un log.
- SalaDAO.java: Gestiona operaciones de base de datos relacionadas con las salas de cine. Ofrece métodos para insertar una nueva sala, mostrar todas las salas existentes, eliminar una sala y actualizar información de una sala. La conexión a la base de datos se establece en el constructor. Los métodos utilizan PreparedStatements para ejecutar consultas y actualizaciones. También, hay métodos para obtener y construir información de una sala específica a partir de su ID. Se utiliza un registro de log para manejar excepciones.

Entidades

En las entidades se encuentran las clases de Actor, Cliente, Comentario, Entrada, Película, Proyección, Reserva y Sala. Son clases típicas de un lenguaje orientado a objetos. Se emplean para crear objetos que se van a introducir en la base de datos y viceversa. Algunas poseen varios constructores que crean un objeto concreto de esa clase dependiendo de la consulta que se le haga a la base de datos.

Base de Datos



La base de datos de la aplicación web (basada en apache derby) tiene la siguiente distribución interna en cuanto a tablas y relaciones:

Tabla actor

- Campos: 'id' (clave primaria), 'nombre', 'apellido'.
- Relaciones: Esta tabla representa a los actores de las películas. La relación con otras tablas se establece a través de la tabla de unión 'pelicula_actor', indicando una relación de muchos a muchos con la tabla 'pelicula'.

Tabla cliente

- Campos: 'id' (clave primaria), 'nombre', 'apellido', 'email', 'contrasena', 'fecha_registro', 'es_admin'.
- Relaciones: Representa a los clientes del sistema. La tabla 'reserva' tiene una relación de uno a muchos con esta tabla, ya que un cliente puede realizar varias reservas, pero una reserva está asociada a un único cliente. La tabla 'comentario' también tiene una relación de uno a muchos con esta tabla, ya que un cliente puede hacer varios comentarios, pero un comentario está asociado a un único cliente.

Tabla sala

- Campos: 'id' (clave primaria), 'nombre_sala', 'filas', 'columnas'.
- Relaciones: No se evidencian relaciones directas con otras tablas, pero se espera que la tabla 'proyeccion' tenga una relación de muchos a uno con esta tabla, ya que varias proyecciones pueden ocurrir en la misma sala.

Tabla pelicula

- Campos: 'id' (clave primaria), 'nombre_pelicula', 'sinopsis', 'pagina_oficial', 'titulo_oficial', 'genero', 'nacionalidad', 'duracion', 'ano', 'distribuidora', 'director', 'otros_datos', 'class_edad', 'portada', 'trailer'.
- Relaciones: La tabla 'pelicula_actor' establece una relación de muchos a muchos con la tabla 'actor'. La tabla 'comentario' tiene una relación de uno a muchos con esta tabla, ya que varios comentarios pueden estar asociados a una película, pero un comentario está relacionado con una única película. La tabla 'proyeccion' tiene una relación de uno a muchos con esta tabla, ya que una película puede proyectarse en varias ocasiones, pero una proyección está asociada a una única película.

Tabla comentario

- Campos: 'id' (clave primaria), 'id_cliente' (clave foránea), 'id_pelicula' (clave foránea), 'texto', 'fecha_comentario'.
- Relaciones: Las claves foráneas 'id_cliente' y 'id_pelicula' están relacionadas con las tablas 'cliente' y 'pelicula', respectivamente.

Tabla proyeccion

- Campos: 'id' (clave primaria), 'id_pelicula' (clave foránea), 'id_sala' (clave foránea), 'fecha_hora'.
- Relaciones: Las claves foráneas 'id_pelicula' e 'id_sala' están relacionadas con las tablas 'pelicula' y 'sala', respectivamente.

Tabla película_actor

- Campos: 'id' (clave primaria), 'pelicula_id' (clave foránea), 'actor_id' (clave foránea).
- Relaciones: Las claves foráneas 'pelicula_id' y 'actor_id' están relacionadas con las tablas 'pelicula' y 'actor', respectivamente.

Tabla reserva

- Campos: 'id' (clave primaria), 'id_cliente' (clave foránea), 'fecha_reserva', 'numero_tarjeta', 'referencia_reserva', 'precio'.
- Relaciones: La clave foránea 'id_cliente' está relacionada con la tabla 'cliente'.

Tabla entrada

- Campos: 'id' (clave primaria), 'id_proyeccion' (clave foránea), 'fila', 'columna', 'reserva_id' (clave foránea).
- Relaciones: Las claves foráneas 'id_proyeccion' y 'reserva_id' están relacionadas con las tablas 'proyeccion' y 'reserva', respectivamente.

Como conclusión, se hay relaciones de uno a muchos entre las tablas:

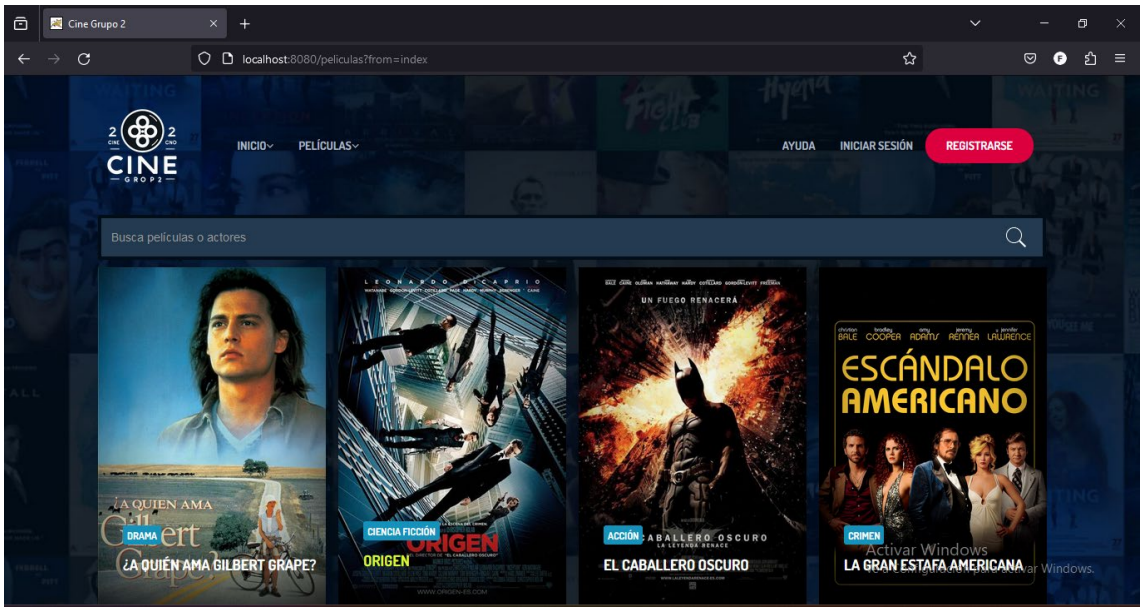
- 'cliente' y 'reserva'
- 'cliente' y 'comentario'
- 'pelicula' y 'comentario'

- ‘pelicula’ y ‘proyeccion’
- ‘sala’ y ‘proyeccion’

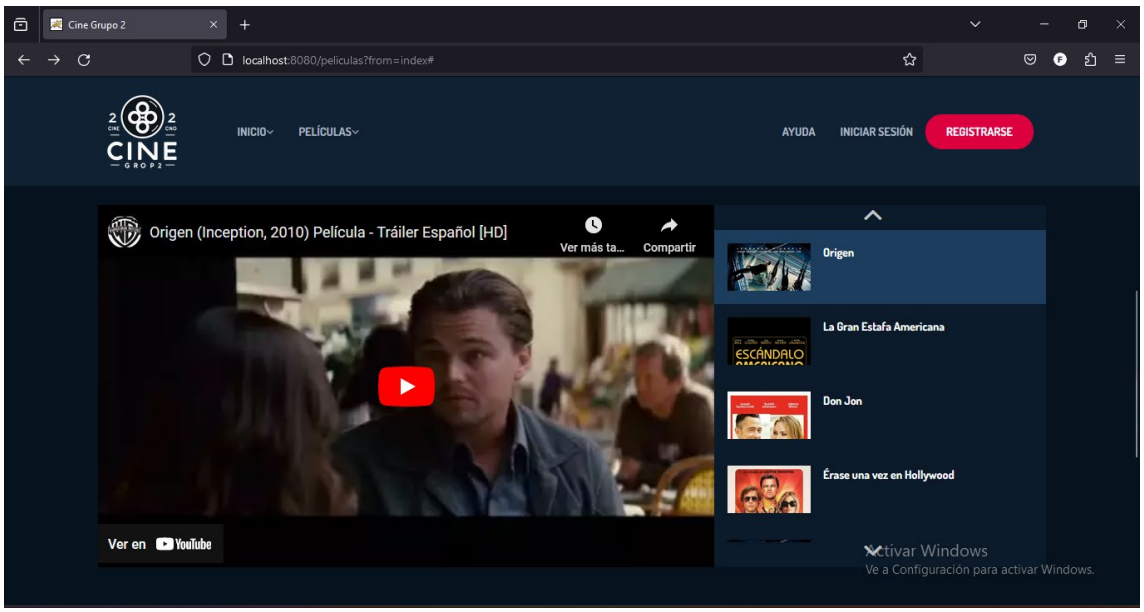
Y hay relación de muchos a muchos entre las tablas:

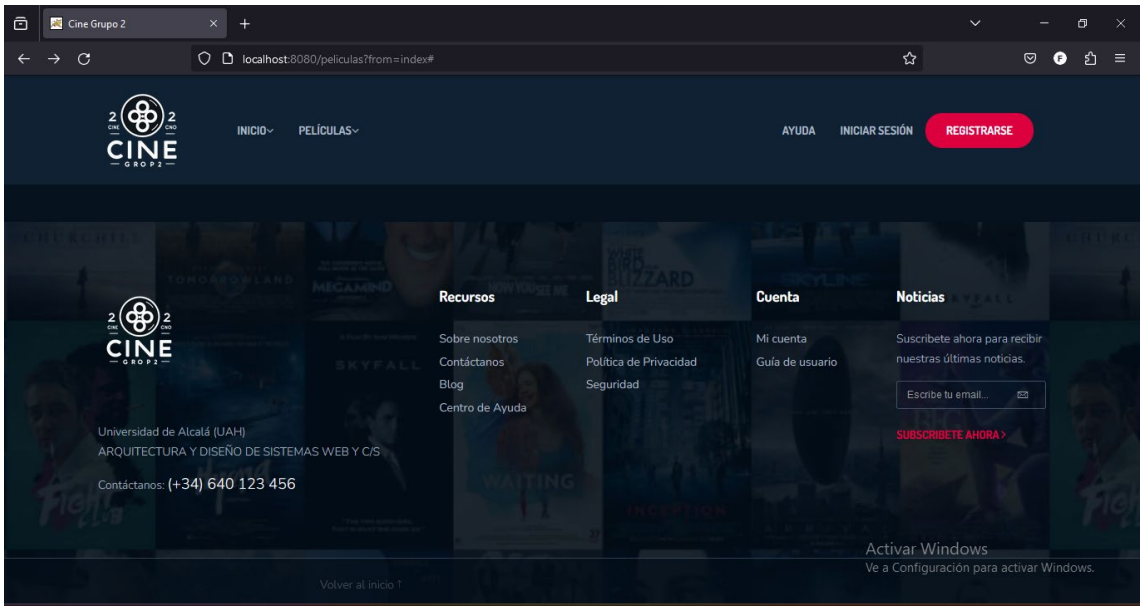
- ‘actor’ y ‘pelicula’ mediante la tabla de unión ‘pelicula_actor’.

Manual de usuario

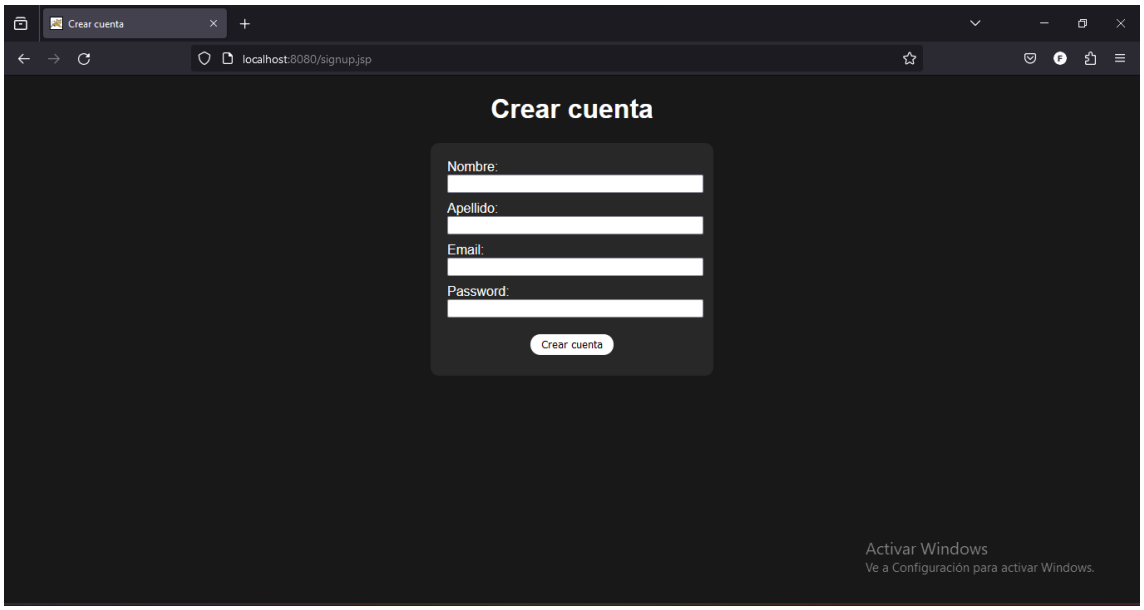


Primero tenemos la ventana principal, desde la cual podemos ir a inicio o acceder a la lista de películas, mediante las listas que aparecen a la izquierda del header. En la derecha tenemos el botón de ayuda (que es un placeholder simulando un botón de ayuda real) así como un botón para iniciar sesión o registrarse. Si hacemos click en los títulos de cualquiera de las películas accederemos a la película en si directamente.

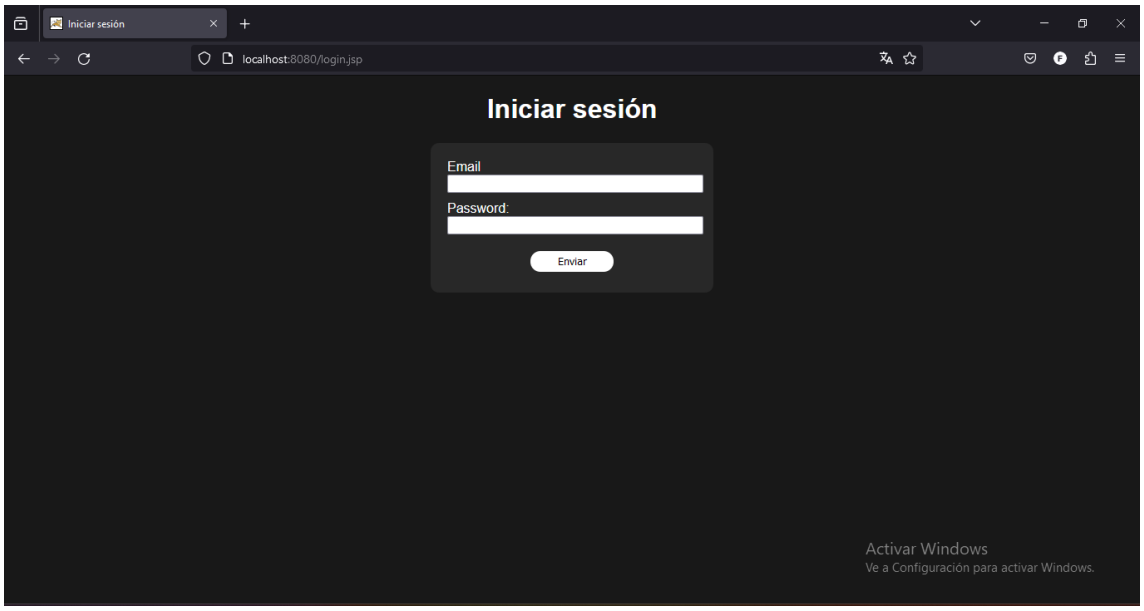




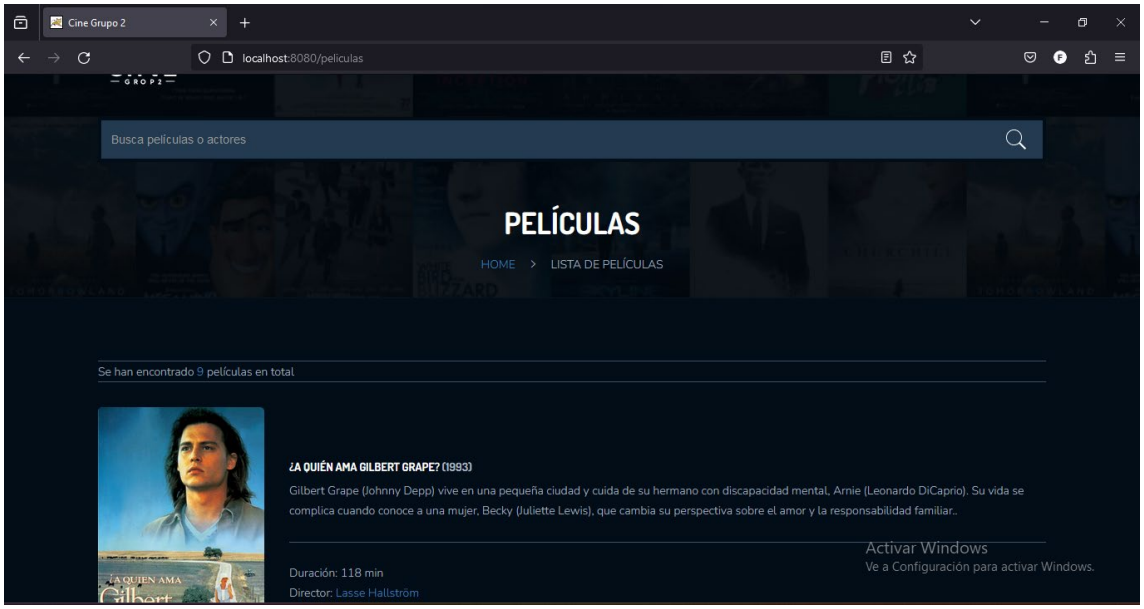
Más abajo tenemos los trailers de las películas, así como el footer de la página, que contiene links placeholder para simular enlaces típicos de una página de estas características.



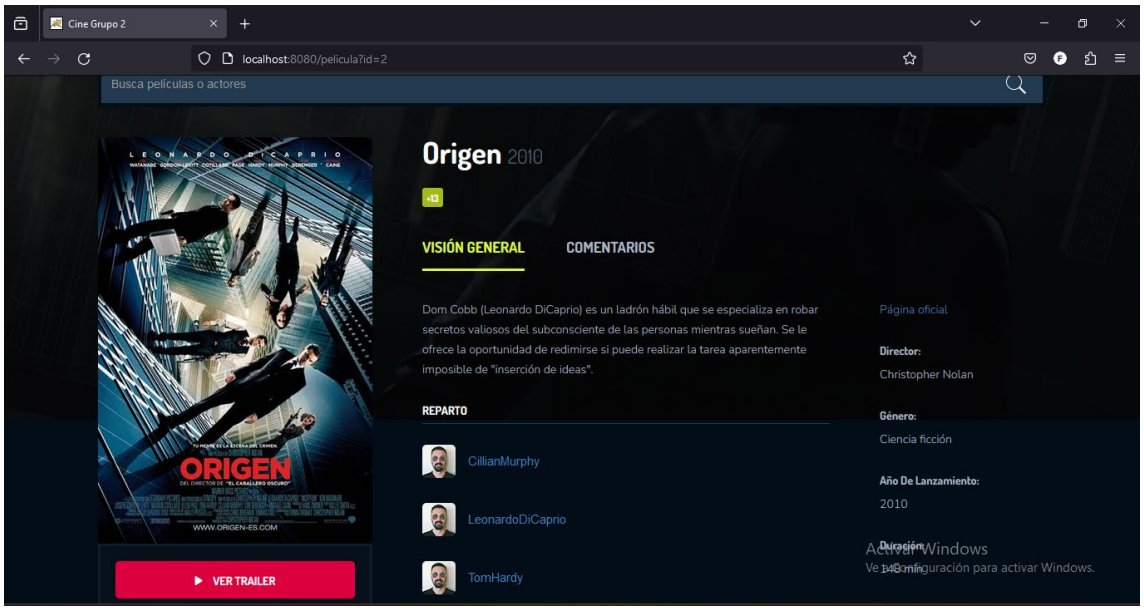
Haciendo click en el botón de registrar tenemos la siguiente ventana, que nos permite crear un usuario para acceder a ciertas funcionalidades exclusivas de ser uno.



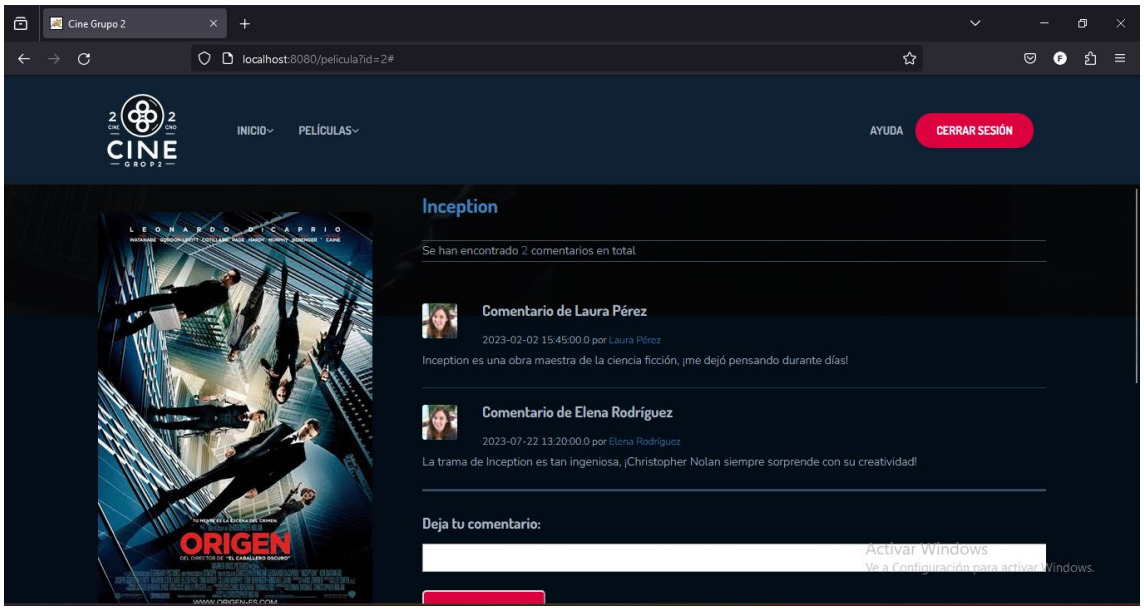
Haciendo click en el botón de iniciar sesión tenemos la siguiente ventana, que nos permitiría iniciar sesión, tanto como admin como cliente. En este caso iniciamos sesión con un cliente para explicar las funcionalidades del mismo.



Haciendo click en la lista de películas nos encontramos con la siguiente ventana. Que nos permite seleccionar cualquier película que tenga proyección en el cine.

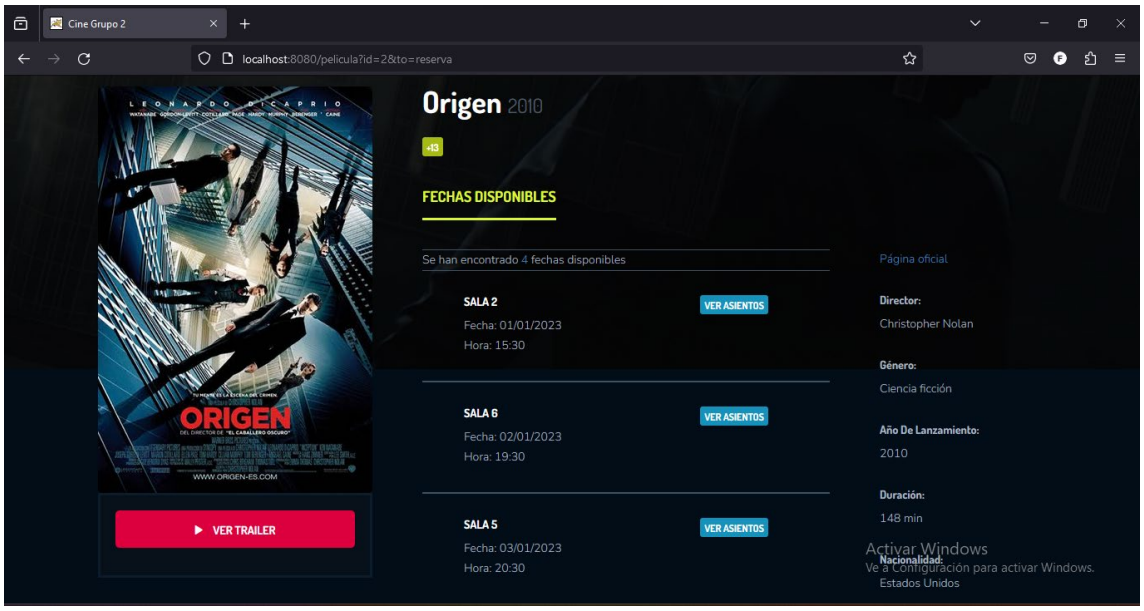


Al hacer click en una película nos encontramos con información básica de la misma.

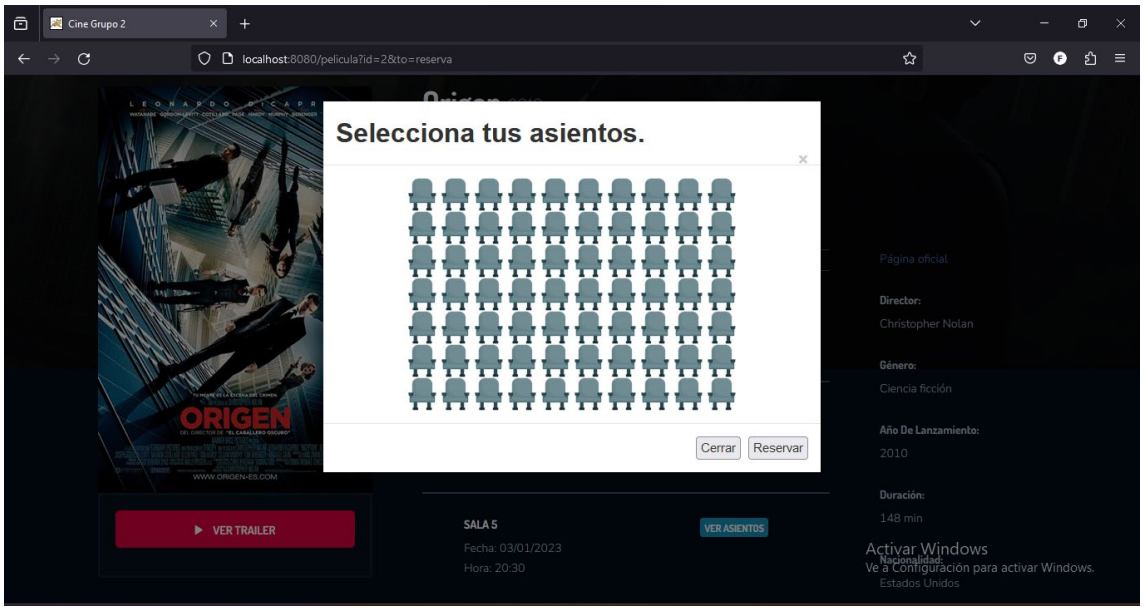


Además podemos ver comentarios de otras personas así como dejar los nuestros propios (todo ello en caso de ser un usuario con la sesión iniciada).

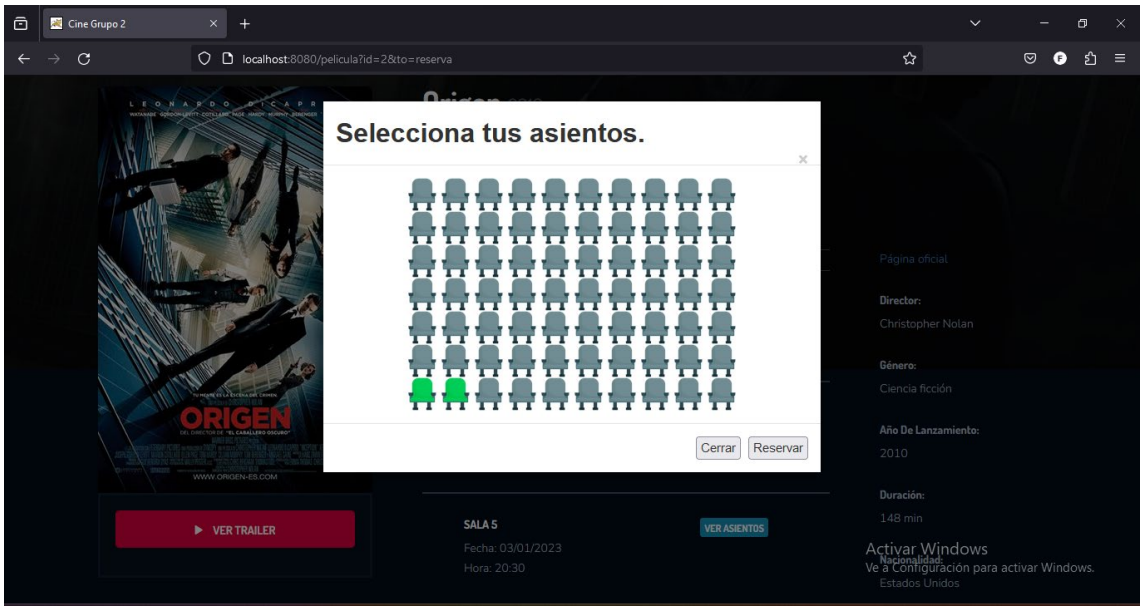
También podemos hacer reserva para ver la película, procederemos a verlo en la siguiente captura.



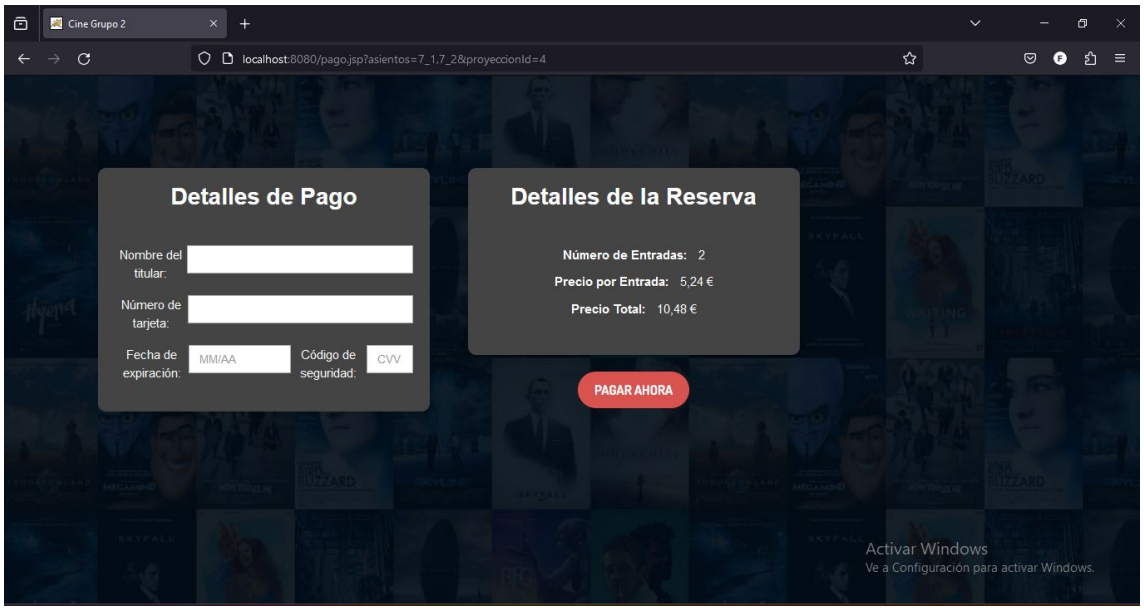
Al hacer click en el botón de hacer reserva se nos muestra las salas en las que se proyecta la película así como la información de las proyecciones.



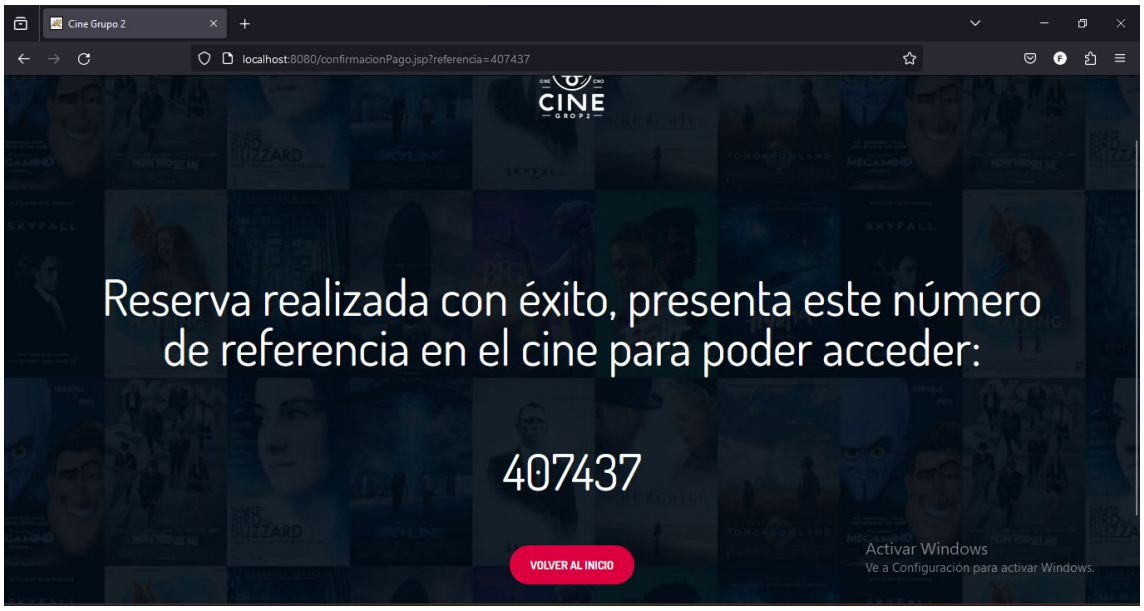
Al hacer click en ver asientos nos salen los asientos libres, que en este caso son todos, en el caso de que hubiera asientos reservados aparecerían en rojo y no podrían ser seleccionados. Procederemos a reservar los 2 primeros asientos de la izquierda.



Como podemos ver se ponen verdes al marcarlos.

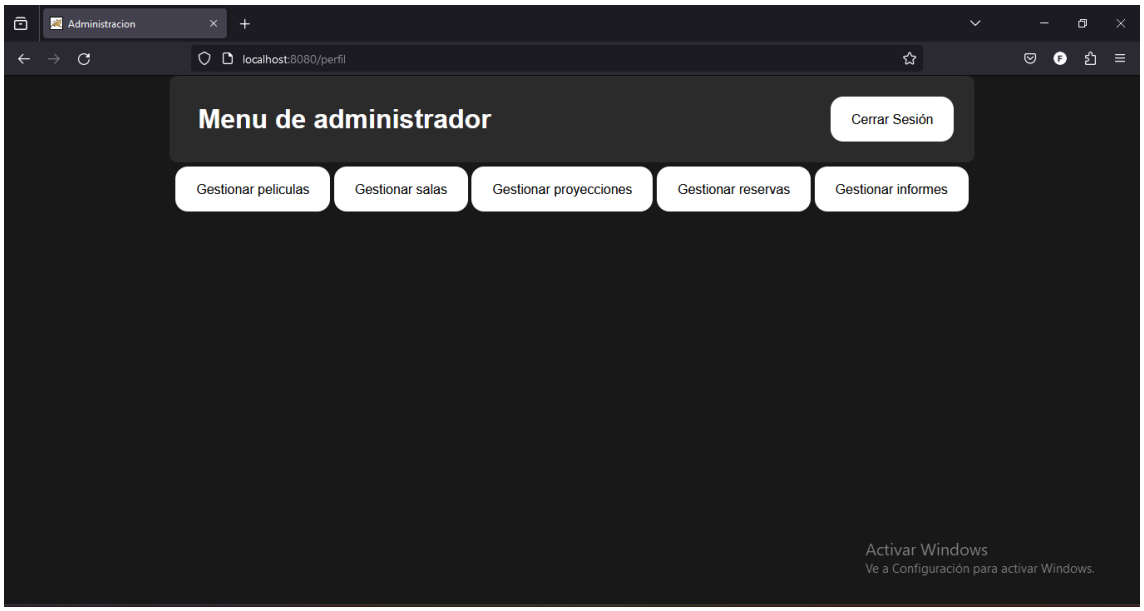


Al hacer click en reservar podemos ver la ventana de reserva, introduciendo la información que nos piden y pulsando pagar, habremos reservado las sillas deseadas.

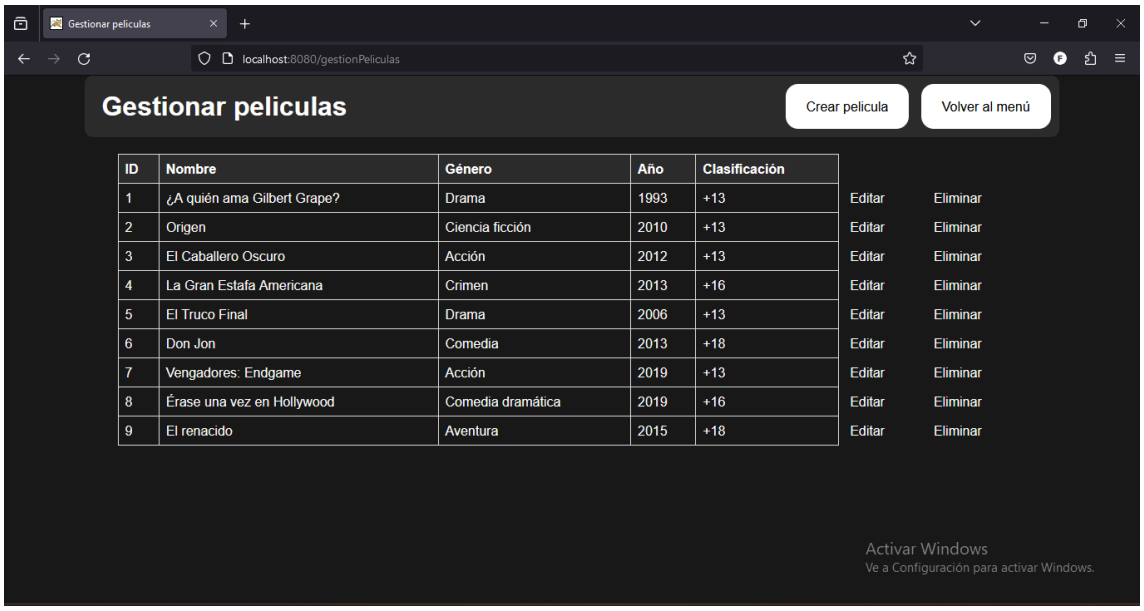


Al reservar nos saldrá el número de reserva así como un botón para volver al inicio.

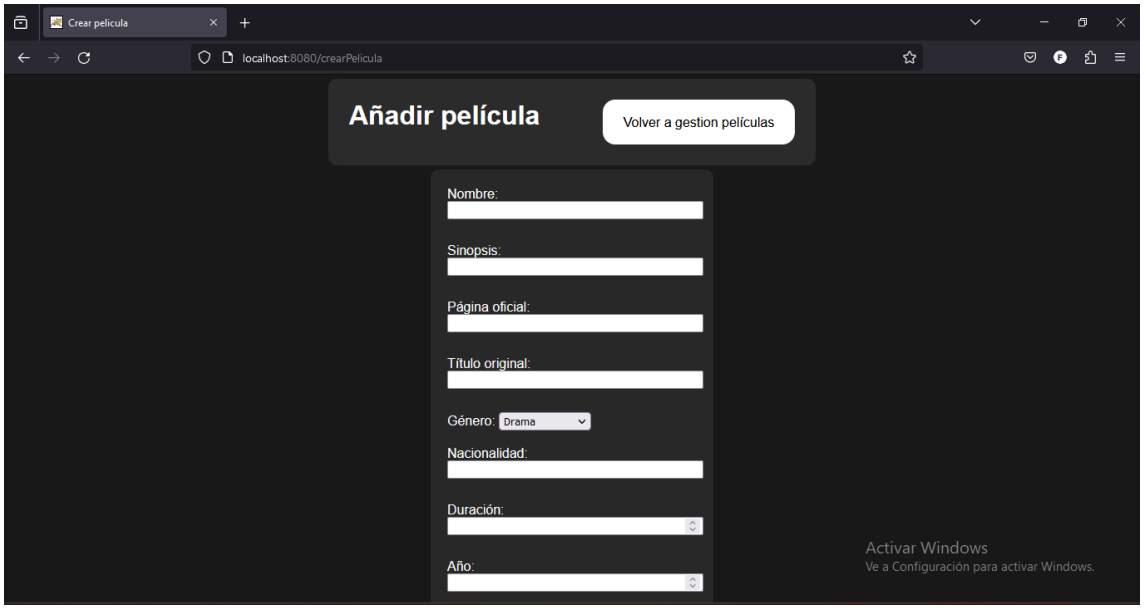
Habiendo explicado la parte del usuario, procederemos a explicar la parte del administrador, junto a todas sus funcionalidades. Cuyo nombre de usuario es admin@admin.com y contraseña es 12345.



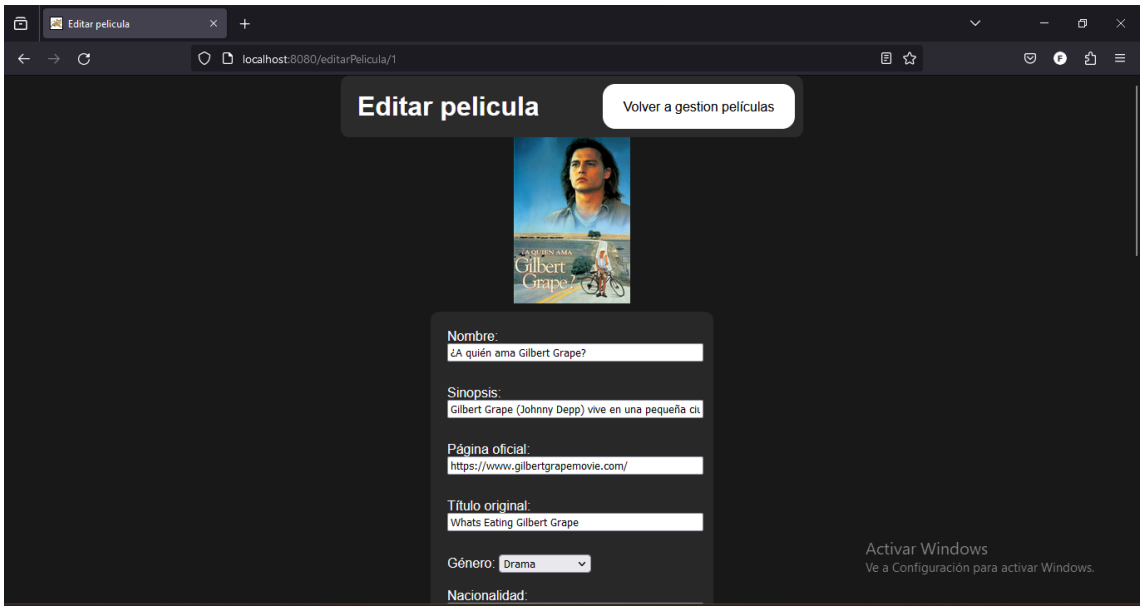
Lo primero que nos aparece es lo siguiente, una ventana que contiene varios enlaces (con aspecto de botones) que nos permiten cerrar sesión del admin (igual que el cliente), gestionar las películas, salas, proyecciones, reservas e informes. Procedemos a explicar cada uno de izquierda a derecha junto a sus funcionalidades.



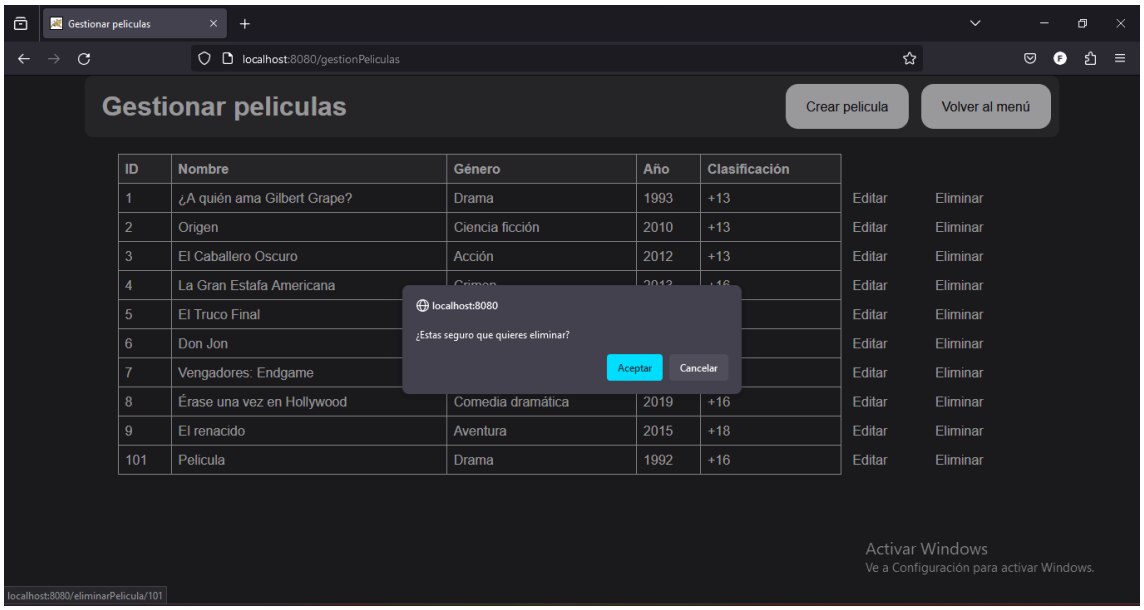
En la ventana de gestionar películas tenemos lo siguiente, una vista con las películas actuales en la bbdd junto a información esencial de la misma, podemos crear otra película, volver al menú o editar o eliminar una de las peliuclas. Procederemos a crear película, después a editar y posteriormente a eliminar una que hayamos creado.



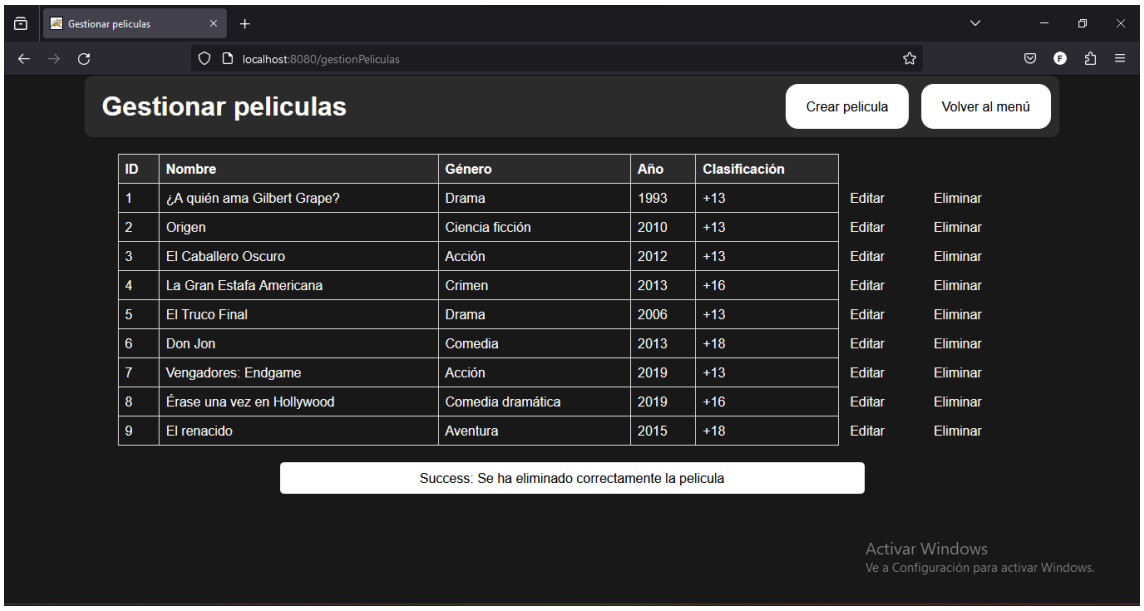
En crear película veremos la siguiente ventana, que nos permite rellenar información de la misma.



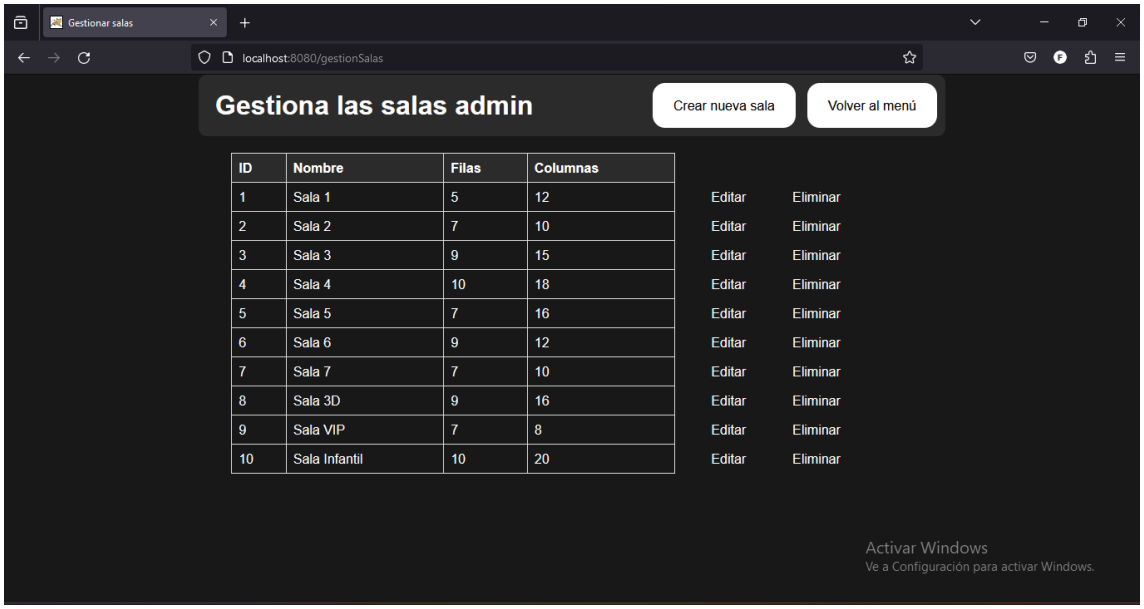
Al hacer click a editar una película nos permite editar los campos de la misma. Al editarla se nos hará saber si ha sido exitoso o ha fallado la edición por algún error interno.



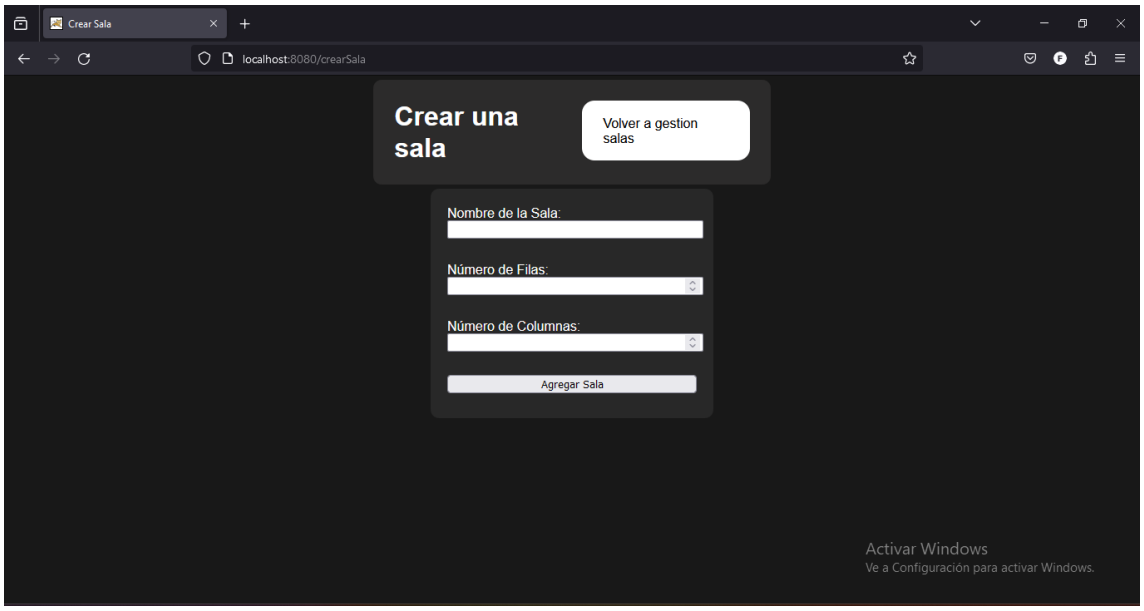
Al hacer click en eliminar nos sacará un “prompt” que nos permite decidir con una segunda oportunidad si queremos eliminarla en caso de que hayamos hecho click por error. Eliminaremos la película llamada película con ID 101 ya que se ha rellenado para mostrar la funcionalidad de este botón.



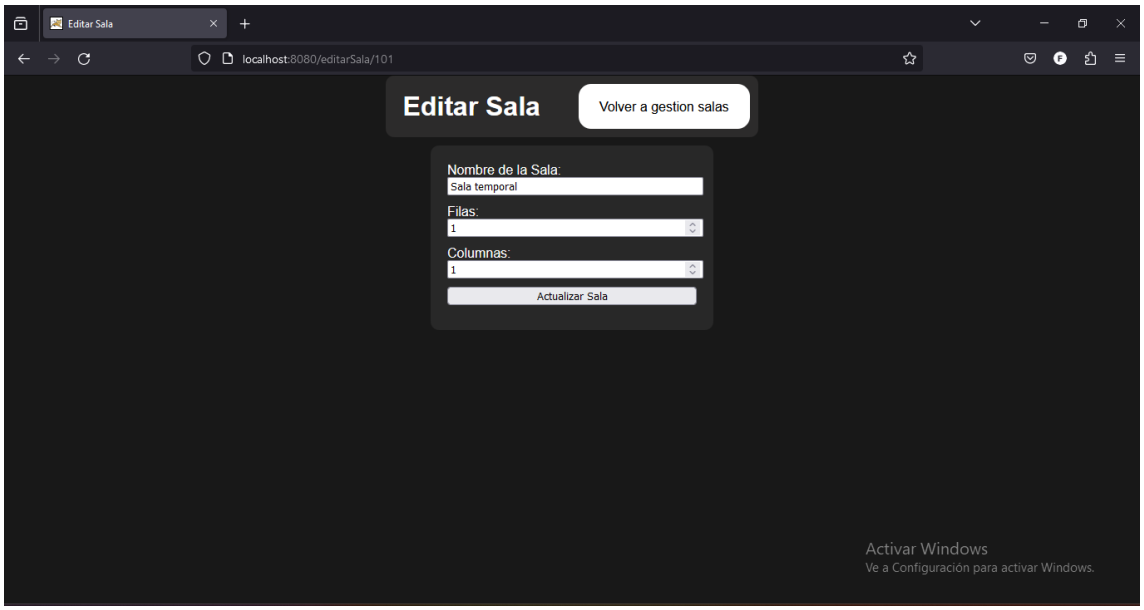
Como podemos ver se nos confirma que se ha eliminado la película, además de que ya no se muestra en la tabla. Ahora haremos click en el botón de volver al menú para acceder al menú de gestión de salas.



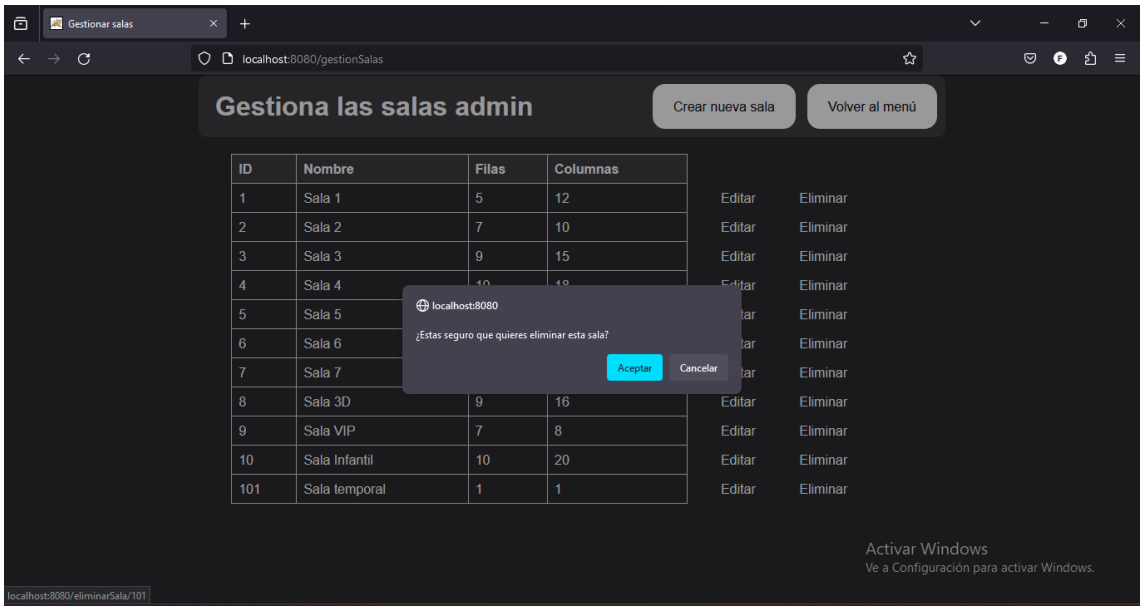
Podemos ver que la distribución es la misma que la de las películas, tenemos los mismos botones de crear, editar y eliminar.



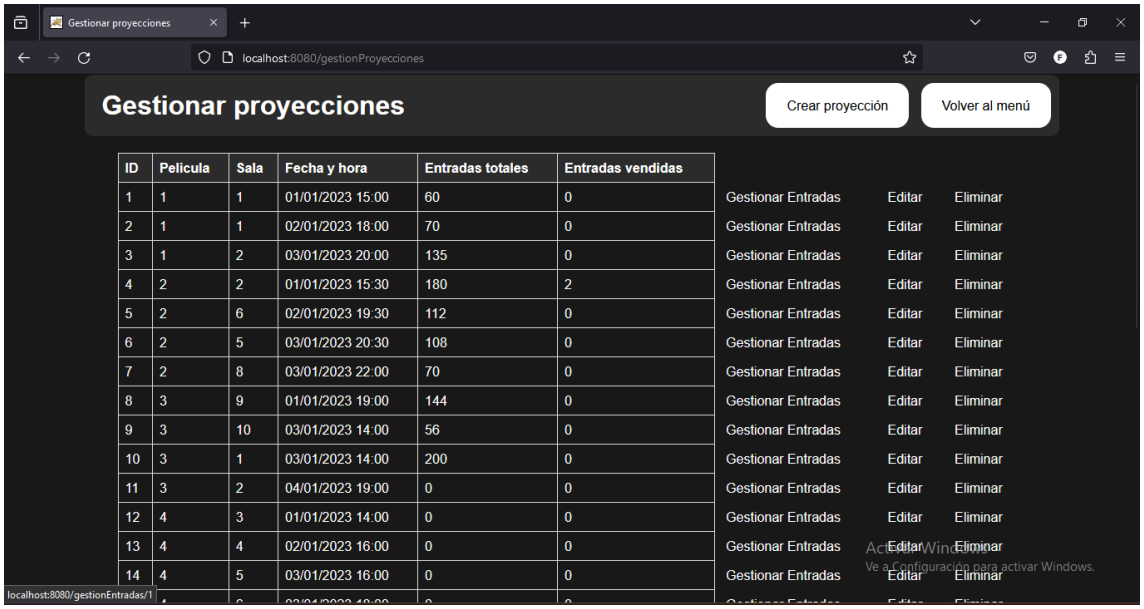
Al hacer click en crear una sala veremos la siguiente ventana que nos permite generar una sala especificando el nombre así como sus filas y columnas.



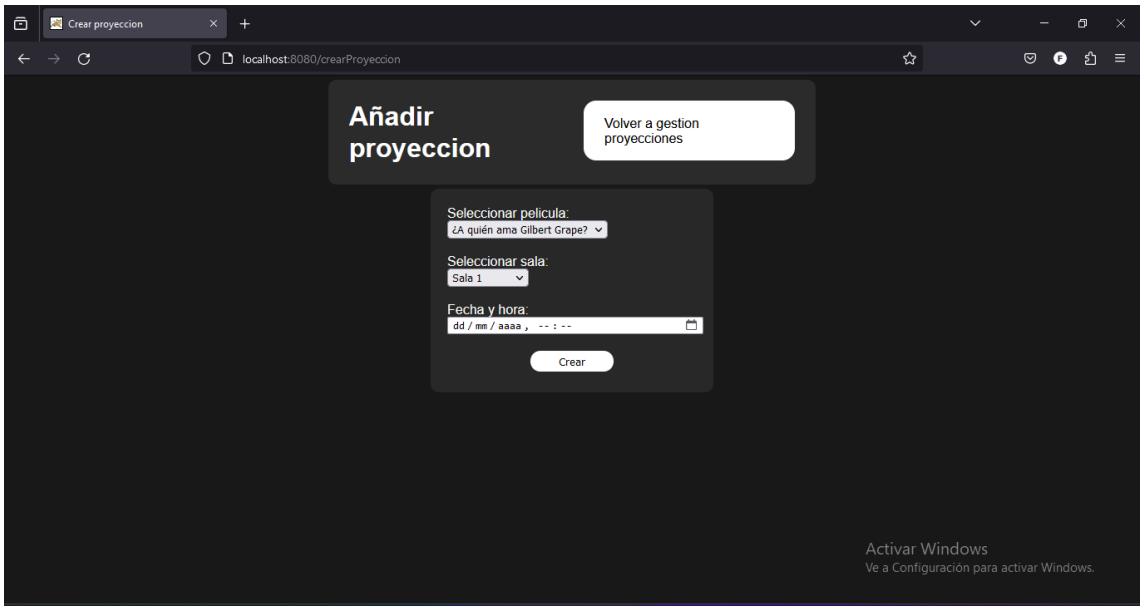
Al hacer click en editar veremos la siguiente página, que es igual a la de crear; simplemente tiene los valores ya rellenados ya que pertenecen a una sala existente, al hacer click en actualizar sala veremos si ha sido exitoso o no (a la hora de haberla editado).



Por último al hacer click en eliminar podremos ver el siguiente "prompt" igual que anteriormente. Al hacer click en eliminar nos hará saber si ha sido exitoso o no. En este caso eliminaremos la sala llamada sala temporal con ID 101. Ahora pasaremos al menú de gestionar proyecciones.

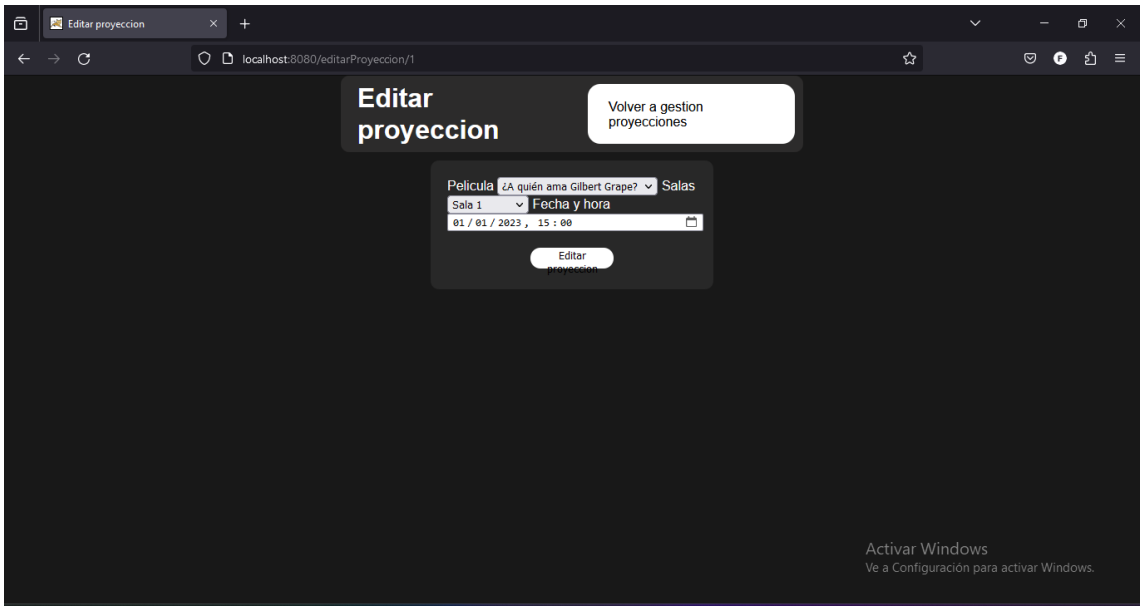


En esta ventana podemos ver las proyecciones junto a su información (se puede ver las 2 reservas que hemos hecho anteriormente). Tenemos 5 botones: crear proyección, volver al menú, gestionar entradas, editar y eliminar (los veremos en este orden).

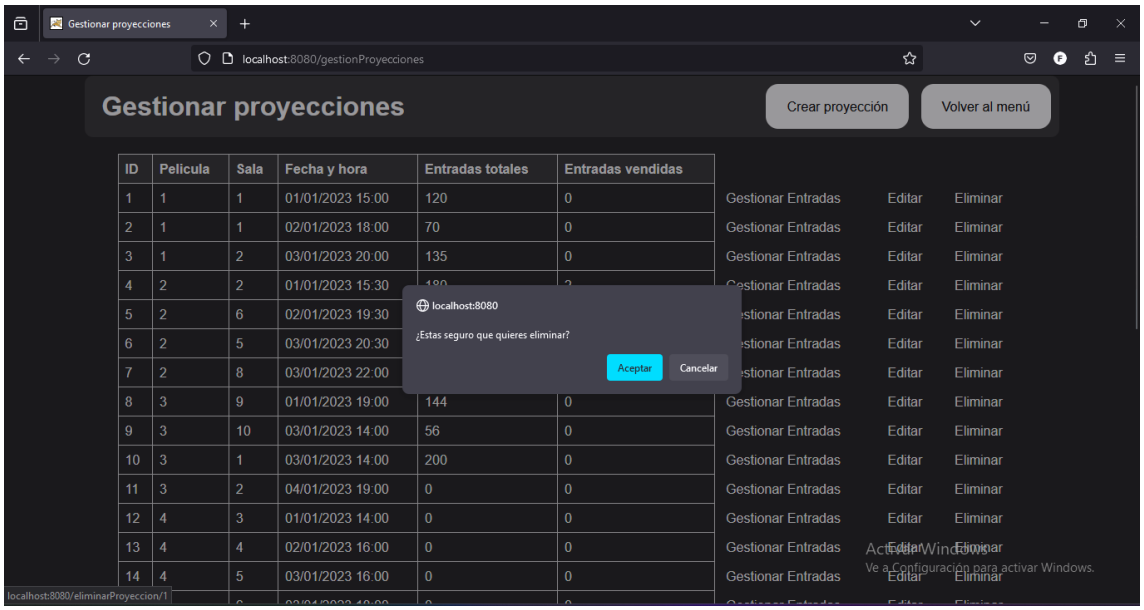


Al hacer click en crear proyección veremos la siguiente ventana, que nos permite seleccionar la película, sala así como la fecha y hora de la proyección a crear.

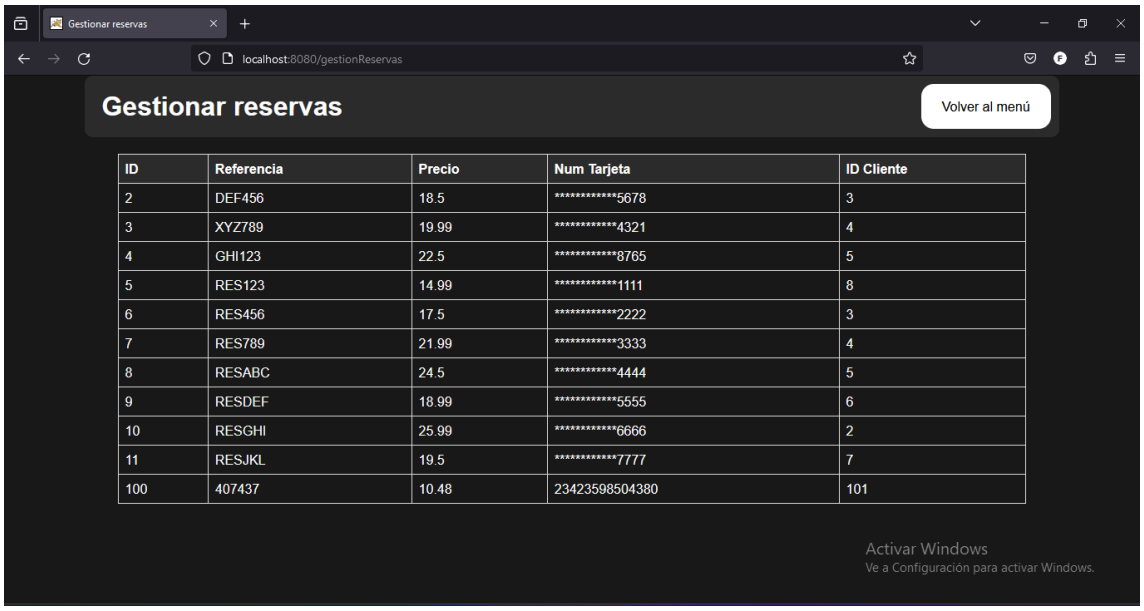
Al hacer click en gestionar entradas de una proyección veremos el siguiente menú en el cual podremos descartar cualquier entrada de una proyección.



Al hacer click en editar proyección podremos editar la proyección que hayamos seleccionado, su hora, sala y película.

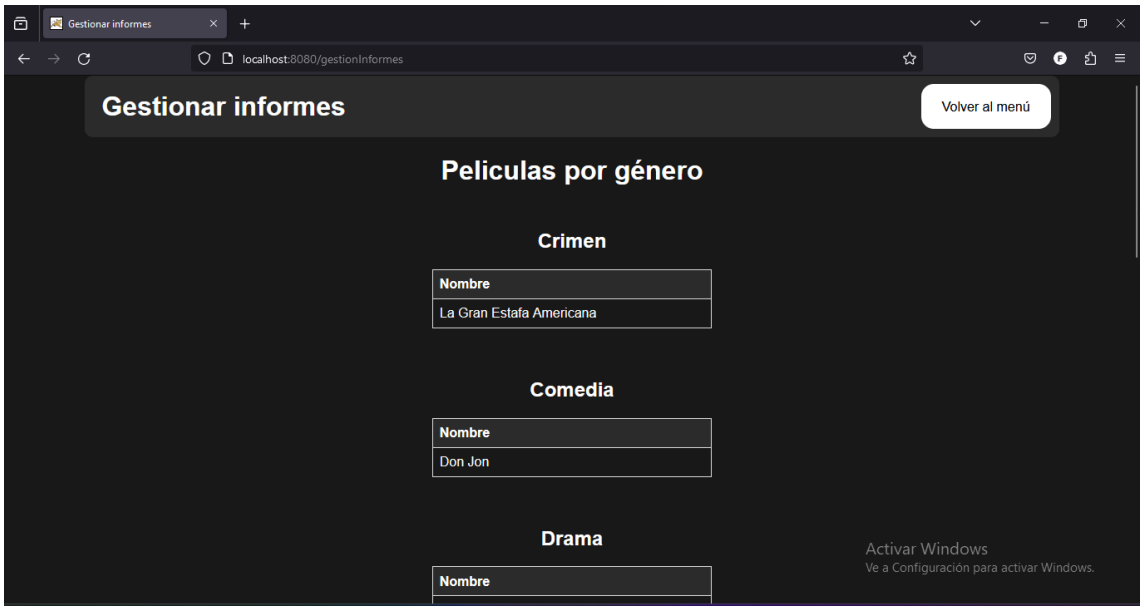


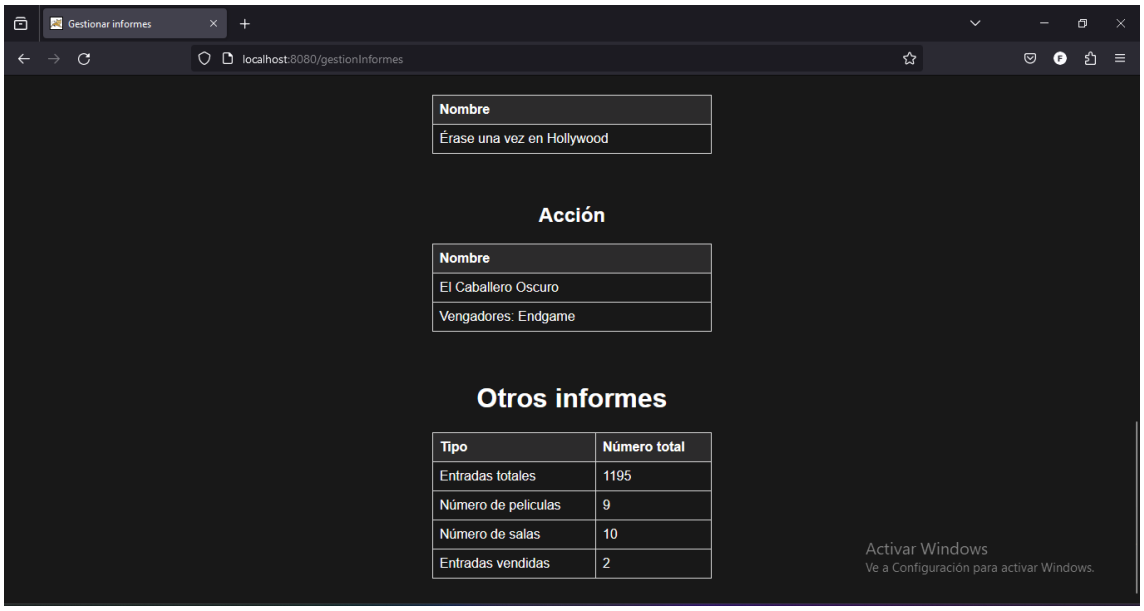
Al hacer click en eliminar veremos el siguiente “prompt” con un funcionamiento igual al resto de veces que hemos realizado esta operación, pero con las proyecciones. Ahora procederemos a visualizar las reservas.



ID	Referencia	Precio	Num Tarjeta	ID Cliente
2	DEF456	18.5	**********5678	3
3	XYZ789	19.99	**********4321	4
4	GHI123	22.5	**********8765	5
5	RES123	14.99	**********1111	8
6	RES456	17.5	**********2222	3
7	RES789	21.99	**********3333	4
8	RESABC	24.5	**********4444	5
9	RESDEF	18.99	**********5555	6
10	RESGHI	25.99	**********6666	2
11	RESJKL	19.5	**********7777	7
100	407437	10.48	23423598504380	101

Esta ventana nos permitirá ver todas las reservas con el número de tarjeta así como el id del cliente que ha hecho la reserva, su numero de referencia, precio y el id de la reserva.





Por último en gestionar informes tendremos información de las películas ordenadas por género así como otras cosas importantes a tener en cuenta dentro de la página. Desde aquí podremos volver al menú y cerrar sesión como admin.