UNIVERSIDAD TECNOLÓGICA
DE HONDURAS

PROYECTO DE PROGRAMACION AVANZADA II

GRUPO 22

Alumno:

Uziel Alejandro García Velásquez

Catedrático:

Louis Carlos Zelaya Vallejo

Asignatura:

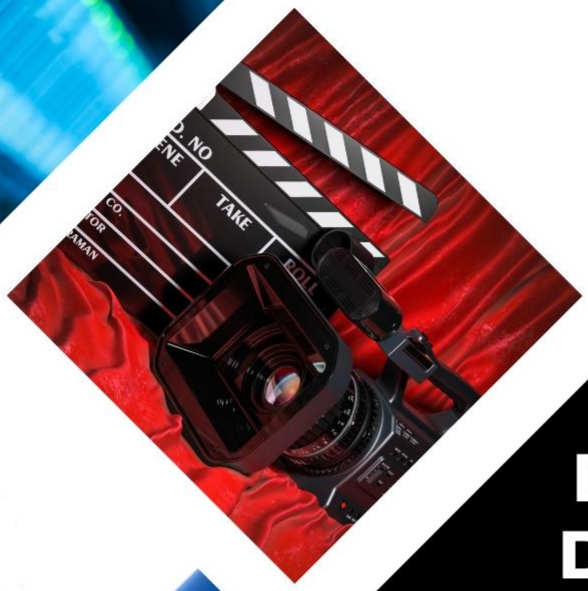
PROGRAMACION AVANZADA II

Tema:

Modificación de BD y creación de proyecto en JAVA.

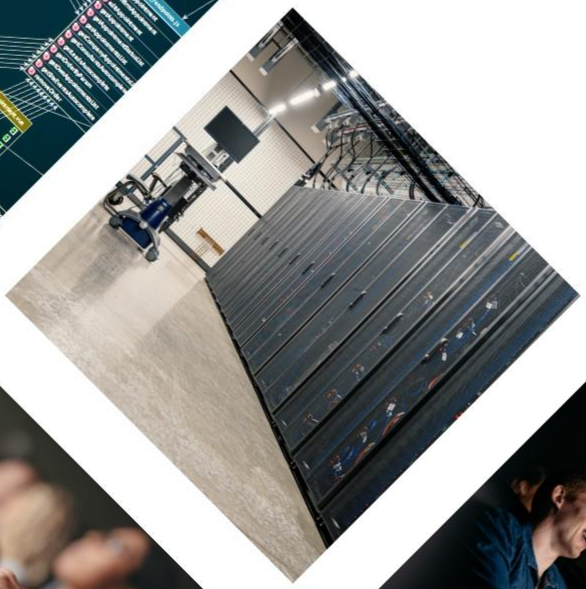
Fecha:

16 de julio de 2025



DOCUMENTACION DE BASE DE DATOS Y PROYECTO EN JAVA DEL CINE

2025



Visión General del Proyecto

Este proyecto consiste en el diseño y creación de una base de datos relacional para la gestión completa de un cine.

El objetivo principal es permitir el control eficiente de las operaciones más importantes del cine, como la administración de películas, funciones, salas, boletos, clientes, empleados, promociones y ventas de snacks.

La base de datos fue diseñada pensando en un entorno realista y funcional para una taquilla presencial. Por eso, se evitó incluir procesos como fidelización o recolección de datos sensibles innecesarios, como correos electrónicos, enfocándose más bien en una atención directa al cliente.

Entre las características clave se incluyen:

- Registro de peliculas con su idioma, duracion y clasificacion segun el sistema usado en Honduras.
- Administracion de salas 2D y 3D, con una capacidad predeterminada.
- Registro de funciones, boletos vendidos y clientes que asisten.
- Aplicacion de promociones como 2x1 y descuentos para estudiantes.
- Control de ventas de snacks asociados a boletos.
- Uso de triggers para aplicar automaticamente descuentos a estudiantes al momento de comprar un boleto.
- Indices para optimizar consultas frecuentes, como ver las funciones de una pelicula o calcular el total recaudado por una funcion.

El proyecto incluye tambien datos de ejemplo realistas para poder probar todas las funciones sin necesidad de ingresar datos manualmente desde cero.

Este sistema puede ser utilizado como base para desarrollar una aplicacion web o de escritorio para gestionar las operaciones de un cine, y se presta muy bien para seguir creciendo con futuras integraciones como reportes financieros, control de stock de snacks, o incluso un pequeño modulo de reservas en linea.

Documentación de la Base de Datos del Cine

Este script crea y organiza una base de datos llamada **cine_db**, que se encarga de gestionar toda la información necesaria para el funcionamiento de un cine. Aquí te explico de manera sencilla qué hace cada parte y cómo se relacionan las tablas.

1. Reinicio de la base de datos

Para empezar, si la base de datos ya existe, la eliminamos para poder crearla desde cero y asegurarnos que todo esté limpio y actualizado:

```
DROP DATABASE IF EXISTS cine_db;
CREATE DATABASE cine_db;
USE cine_db;
```

2. Tablas principales y su función

Películas

La tabla **peliculas** almacena la información de las películas que se van a proyectar, incluyendo:

- Título
- Idioma (español, inglés o subtítulos)
- Duración en minutos
- Clasificación por edades
- Descripción breve

Clientes

Aquí guardamos los datos de las personas que compran boletos:

- Nombre
- Edad
- Si son estudiantes (importante para descuentos)

Roles y Empleados

- **roles**: Define los diferentes puestos de trabajo dentro del cine, como cajero o seguridad.

- **empleados**: Guarda la información de las personas que trabajan en el cine y su rol.

Usuarios

Esta tabla es para el sistema de login, donde cada usuario tiene un nombre de usuario, contraseña (guardada de forma segura) y está asociado a un empleado.

Salas

Aquí están los diferentes espacios físicos donde se proyectan las películas, indicando:

- Nombre de la sala
- Tipo de proyección (2D o 3D)
- Capacidad (cantidad de personas que pueden entrar)

Funciones

Las **funciones** representan las sesiones o horarios en los que se proyectan las películas, vinculando:

- La película que se va a proyectar
- La sala donde será la función
- La fecha y hora

Asientos

Cada función tiene sus asientos asignados para controlar si están disponibles o ya ocupados, evitando sobreventas.

Boletos

Los boletos que compran los clientes están vinculados a:

- La función a la que asistirán
- El cliente que los compra

- El asiento reservado (si aplica)
- El precio pagado
- El método de pago (efectivo o tarjeta)

Además, si el cliente es estudiante, automáticamente se le aplica un descuento gracias a un trigger especial.

Promociones

Esta tabla guarda las ofertas que puede tener el cine, como:

- 2x1 los viernes
- Descuentos para estudiantes
- Descuentos especiales por día

Cada promoción incluye su nombre, descripción, porcentaje de descuento, día de la semana al que aplica y a quién está dirigida.

Snacks y ventas

- **snacks**: Lista de productos para consumir en el cine con sus precios.
- **ventas_snacks**: Registro de las compras de snacks asociadas a cada boleto, con cantidades.

3. Datos iniciales insertados

Para comenzar a trabajar, se agregan ejemplos de:

- Roles del personal (Cajero, Gerente, etc.)
- Películas con sus detalles
- Clientes
- Empleados
- Usuarios para acceder al sistema
- Salas con su tipo y capacidad
- Funciones programadas
- Promociones activas
- Snacks disponibles para venta

4. Automatización: Descuento para estudiantes

Se creó un **trigger** que, antes de guardar un boleto, revisa si el cliente es estudiante. Si lo es, automáticamente aplica un 60% de descuento en el precio del boleto.

5. Índices para mejorar rendimiento

Para que las consultas a la base de datos sean rápidas y eficientes, se crearon índices en las columnas más usadas para búsquedas y relaciones, por ejemplo:

- Índices para buscar funciones por película o sala.
- Índices para buscar boletos por función, cliente o asiento.
- Índices para búsquedas rápidas en asientos, ventas de snacks y usuarios.

Justificación Técnica de los Cambios en la Base de Datos

Para el correcto desarrollo del proyecto en Java, fue necesario realizar una serie de modificaciones y mejoras en el diseño de la base de datos que permitan una integración más eficiente y segura con la lógica de la aplicación. Estas modificaciones tienen como objetivo principal facilitar la validación de datos y la creación o manejo de nuevos campos desde la capa de Java, garantizando la integridad y consistencia de la información.

Entre los principales cambios destacan:

1. **Definición clara de tipos y restricciones en columnas**:
   Se usaron tipos específicos (como ENUM para campos de idioma, clasificación, tipo de sala, estado del asiento, etc.) que permiten validar automáticamente valores permitidos directamente desde la base de datos, evitando datos inconsistentes y facilitando el control desde la aplicación Java.
2. **Incorporación de claves foráneas y relaciones referenciales**:
   Estas relaciones aseguran la integridad referencial entre tablas (por ejemplo, entre funciones, películas y salas; boletos y clientes; asientos y funciones). Esto permite que desde Java se puedan realizar validaciones lógicas basadas en estas relaciones, evitando errores como referencias a registros inexistentes.
3. **Campos adicionales para funcionalidad específica**:
   Se añadieron campos como estudiante en clientes o asiento_id en boletos, lo que permite implementar reglas de negocio en la capa Java, como aplicar descuentos o asignar asientos específicos, con la base de datos preparada para almacenar esa información.

4. **Triggers para automatización y validación en base de datos**:
   La creación del trigger para aplicar descuento a estudiantes es un ejemplo de validación y lógica que se puede controlar a nivel de base de datos, asegurando que siempre se apliquen las reglas independientemente de la capa de aplicación. Esto complementa las validaciones que se harán en Java.
5. **Índices para mejorar el rendimiento en consultas frecuentes**:
   La creación de índices en las columnas más consultadas o relacionadas permitirá que las operaciones desde Java (consultas, búsquedas, actualizaciones) sean más rápidas y eficientes, mejorando la experiencia del usuario final.

Documentación de Proyecto Java

Estructura de Paquetes

1. cine.Cine

Clase principal. Es el **punto de inicio** del programa. Al ejecutar la aplicación, esta clase lanza el formulario de inicio de sesión.

```
public static void main(String[] args) {
    LoginForm ventana = new LoginForm();
    ventana.setVisible(true);
}
```

Función clave: Iniciar la interfaz gráfica de login.

2. view.LoginForm

Interfaz visual (no incluida en este código) donde el usuario debe ingresar su usuario y contraseña para acceder al sistema. Se conecta con la clase UsuarioDAO para validar las credenciales.

3. dao.UsuarioDAO

Este módulo permite verificar el acceso de los usuarios al sistema. Realiza la validación mediante la consulta SQL, donde compara el nombre de usuario y la contraseña cifrada con SHA-256.

Función clave:
public Usuario validarUsuario(String username, String password)

Si encuentra coincidencia, devuelve un objeto Usuario con la información del empleado asociado.
Si no hay coincidencia, devuelve null.

4. dao.FuncionDAO

Es una **interfaz DAO (Data Access Object)** que define las operaciones necesarias para trabajar con las funciones de películas:

- registrar(Funcion funcion): agregar una nueva función.
- actualizar(Funcion funcion): modificar una función existente.
- eliminar(int idFuncion): borrar una función por su ID.
- listar(): obtener una lista de todas las funciones registradas.

Esta interfaz permite que cualquier implementación respete estas reglas básicas.

5. dao.FuncionDAOImpl

Implementación concreta de la interfaz anterior. Contiene el código SQL que permite manipular las funciones en la base de datos:

- Usa **consultas parametrizadas** para prevenir inyecciones SQL.
- Controla la conexión a la base de datos a través de la clase Conexion.

Permite agregar, editar, eliminar y consultar las funciones registradas.
Los resultados se convierten en objetos de tipo Funcion, que luego pueden ser usados directamente en la interfaz gráfica.

6. model.Funcion

Es el **modelo de datos** que representa una función de cine. Cada objeto almacena:

- ID de la función.
- ID de la película.
- ID de la sala.
- Fecha y hora de la función.

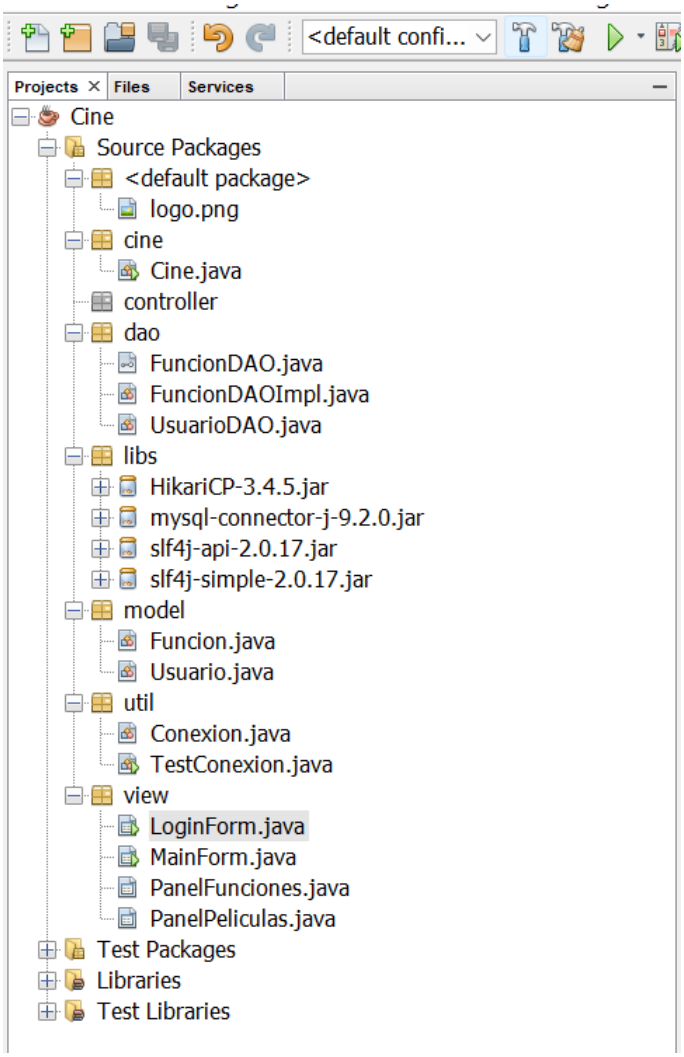
Este modelo es usado tanto para mostrar datos en la interfaz como para enviar información desde/hacia la base de datos.

Flujo de funcionamiento

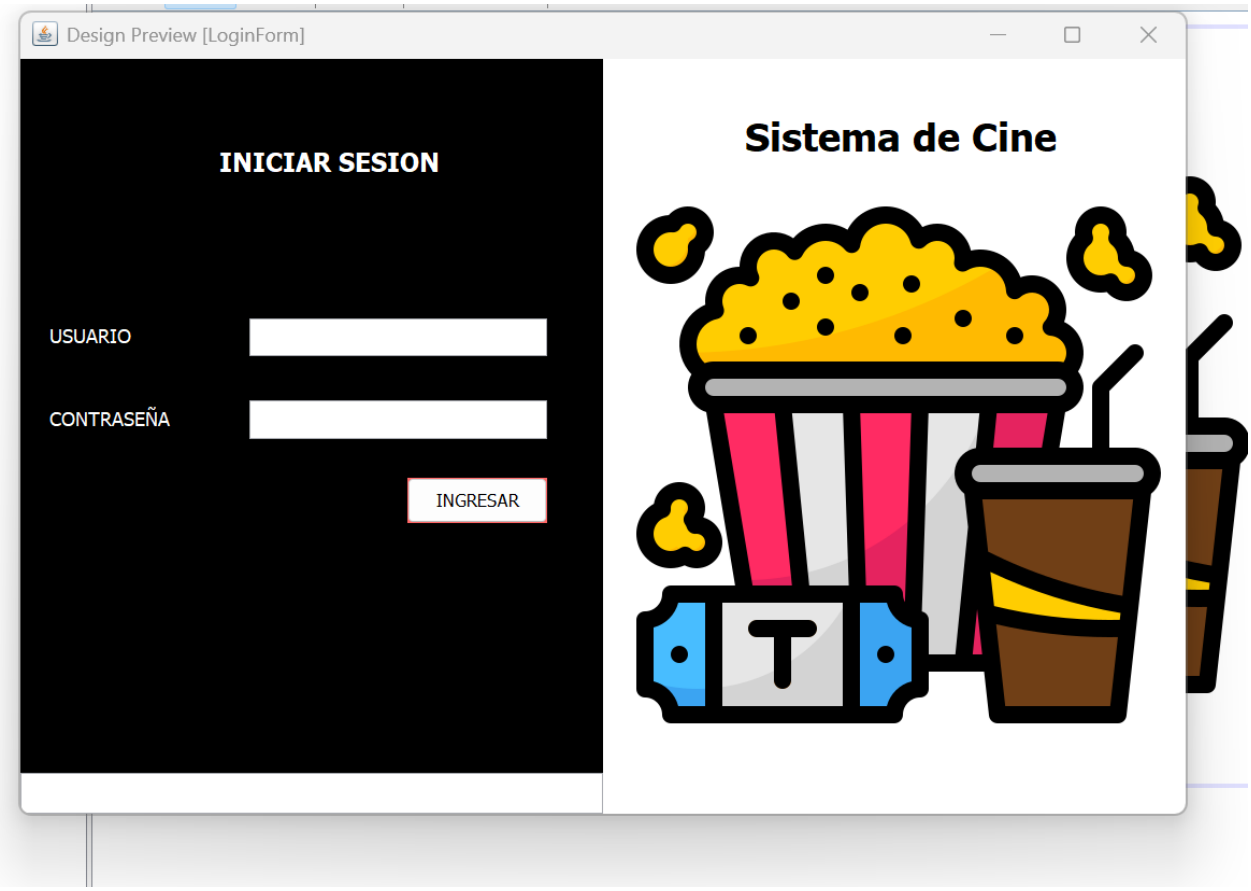
1. El usuario abre el sistema, ve el **formulario de login**.
2. Ingresa credenciales → UsuarioDAO valida el acceso.
3. Una vez dentro, la aplicación permite gestionar las funciones:
    - Agregar funciones nuevas.
    - Editar funciones existentes.
    - Eliminar funciones.
    - Consultar funciones programadas.
4. Los datos de cada función se cargan y manipulan como objetos de la clase Funcion.

ANEXOS

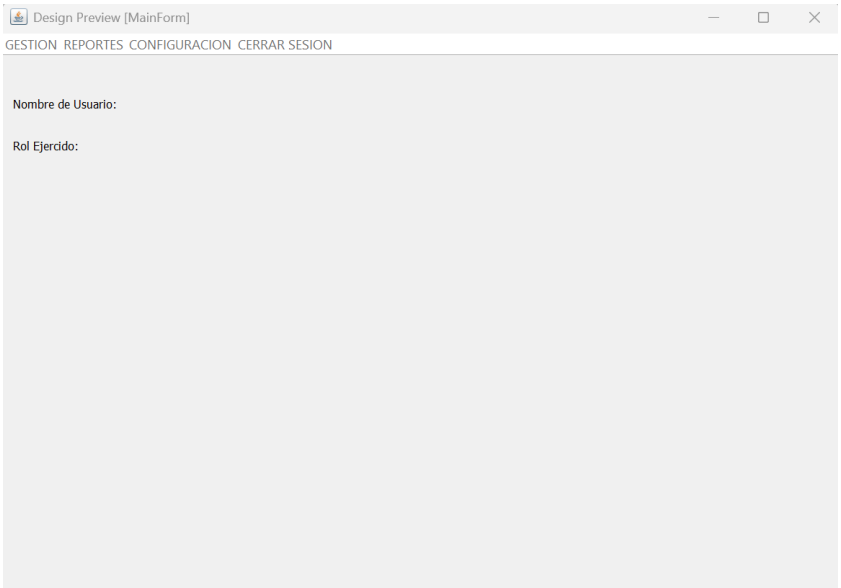
Estructura del proyecto:



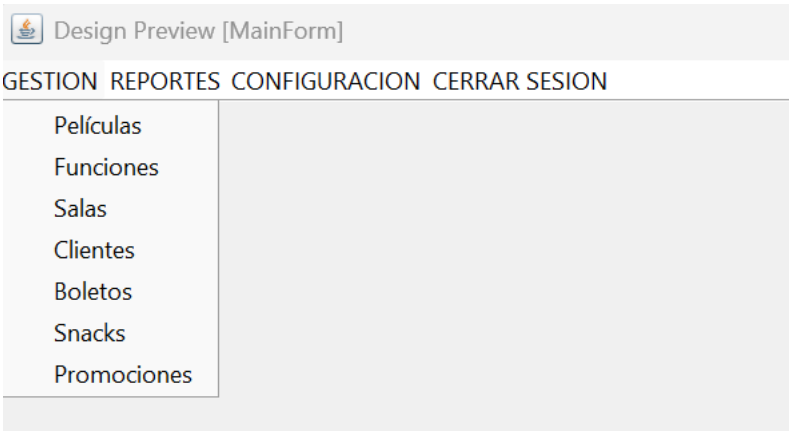
Pantalla de login del proyecto:

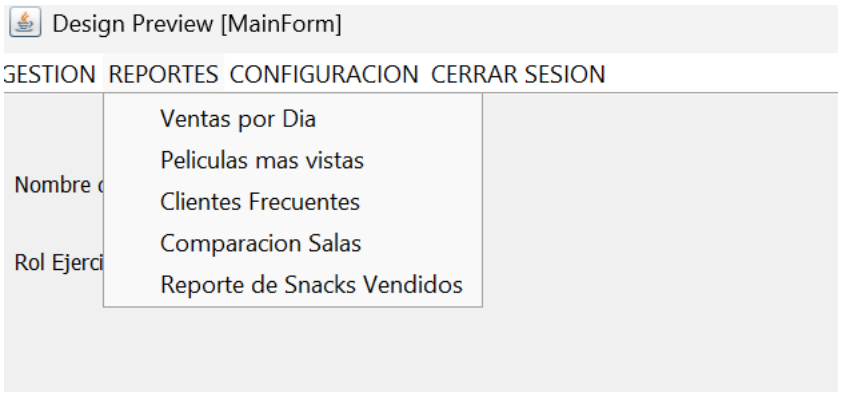


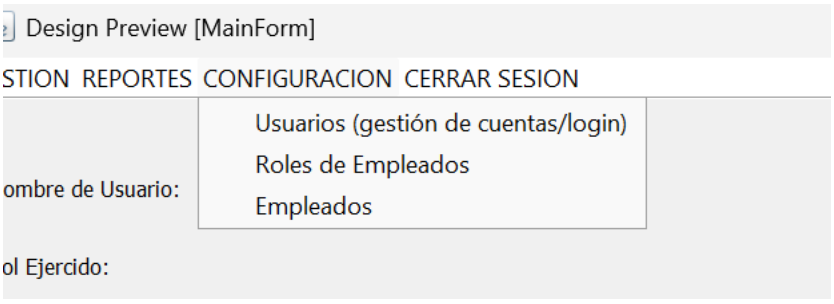
MainForm es mi segunda ventana y es la principal que hará la función de ejecturar otras ventanas. Por ejemplo: gestión películas, gestión boletos, gestión asiento etc…



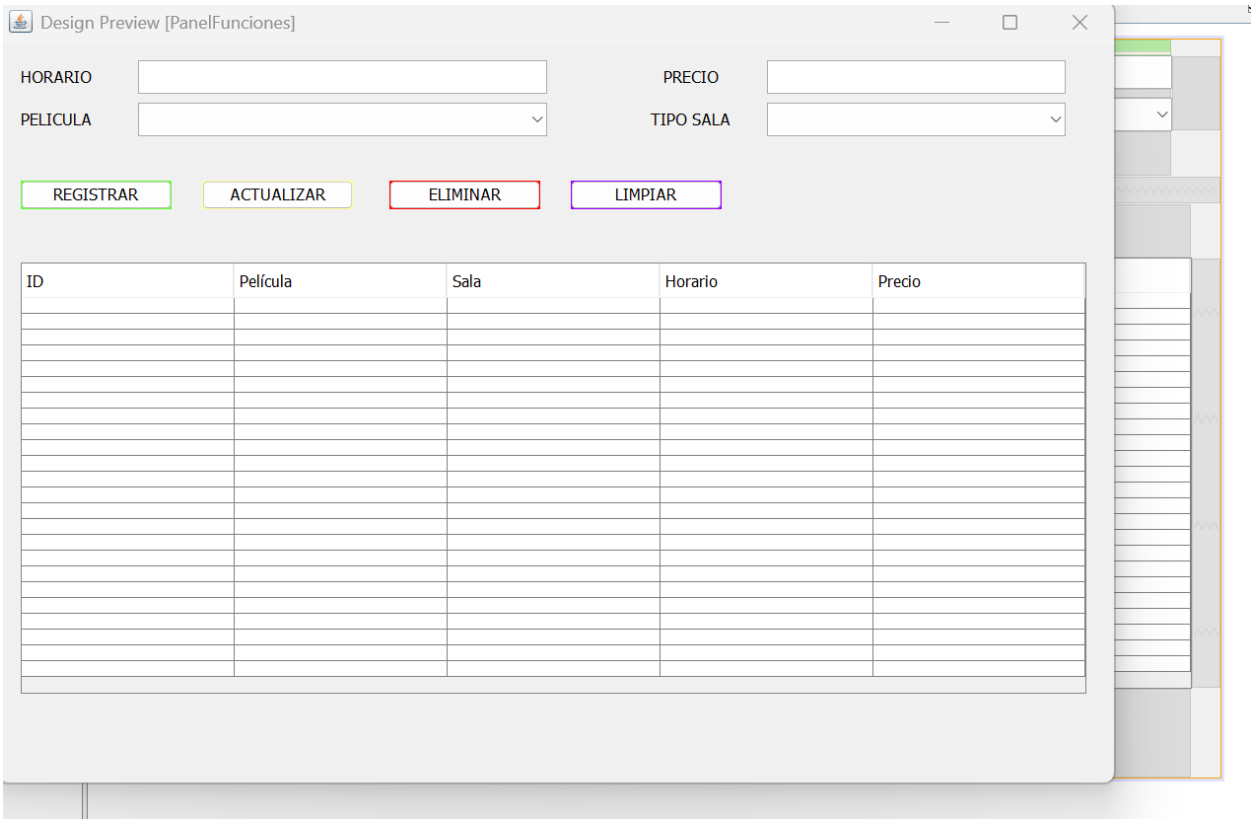
Menu en el MainForm con las opciones disponibles.







Ventana de gestión de Funciones:



Ventana de gestión de películas: