GUIÓN DEFENSA PROYECTO

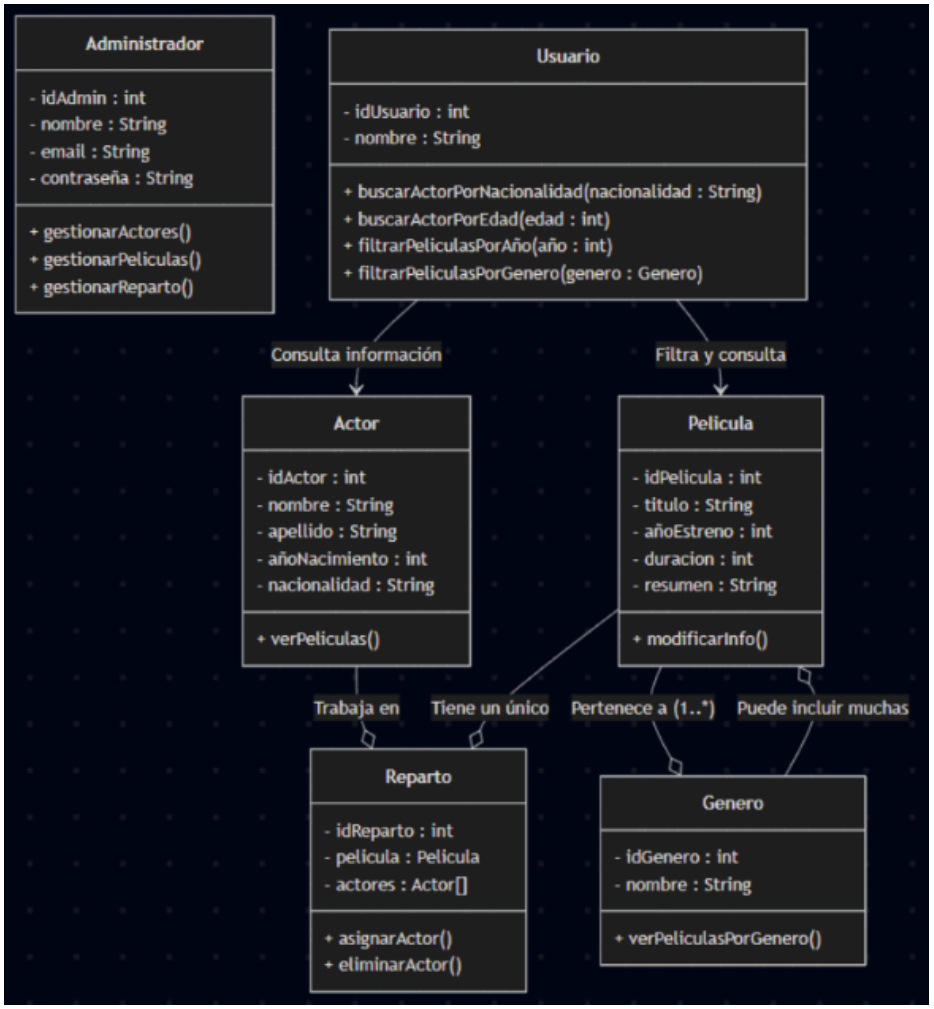
Sistema de Gestión de Películas y Actores/Actrices

La aplicación está diseñada para ayudar a gestionar información sobre actores, actrices, películas, géneros y el reparto de las películas. Su objetivo es facilitar la búsqueda y organización de datos relacionados con el cine, haciendo que sea más fácil para los administradores manejar esta información.

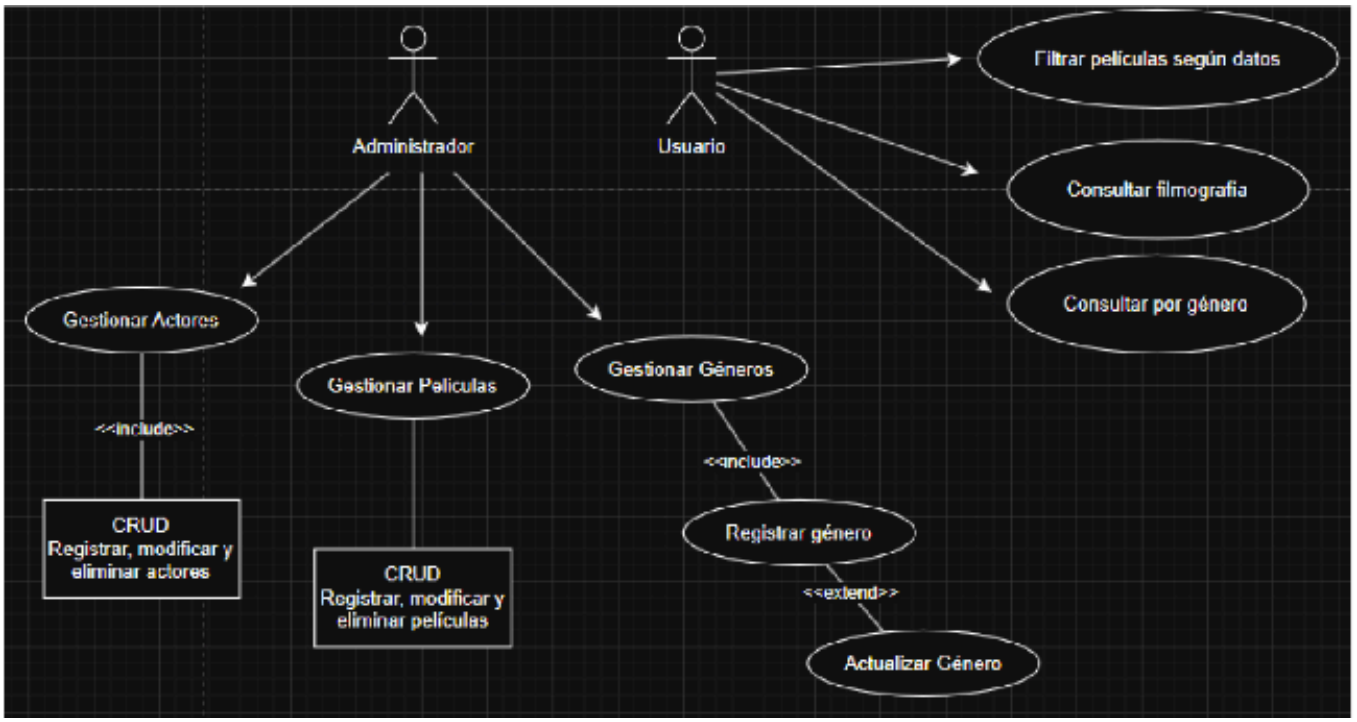
La aplicación soluciona el problema de no tener un lugar centralizado donde se pueda gestionar toda la información sobre películas, actores y su reparto. Lo que hace que los administradores realicen tareas como agregar, modificar o eliminar datos.

El Sistema de Gestión de Películas y Actores/Actrices permite a los administradores gestionar actores, películas, géneros y el reparto de forma fácil. La aplicación se conecta a una base de datos para guardar información, y da opciones para buscar como usuario información de películas y actores.

Diagrama de Clases:
Muestra las principales partes del sistema, como **Actor**, **Película**, **Reparto**, **Género**, y cómo se relacionan entre sí.



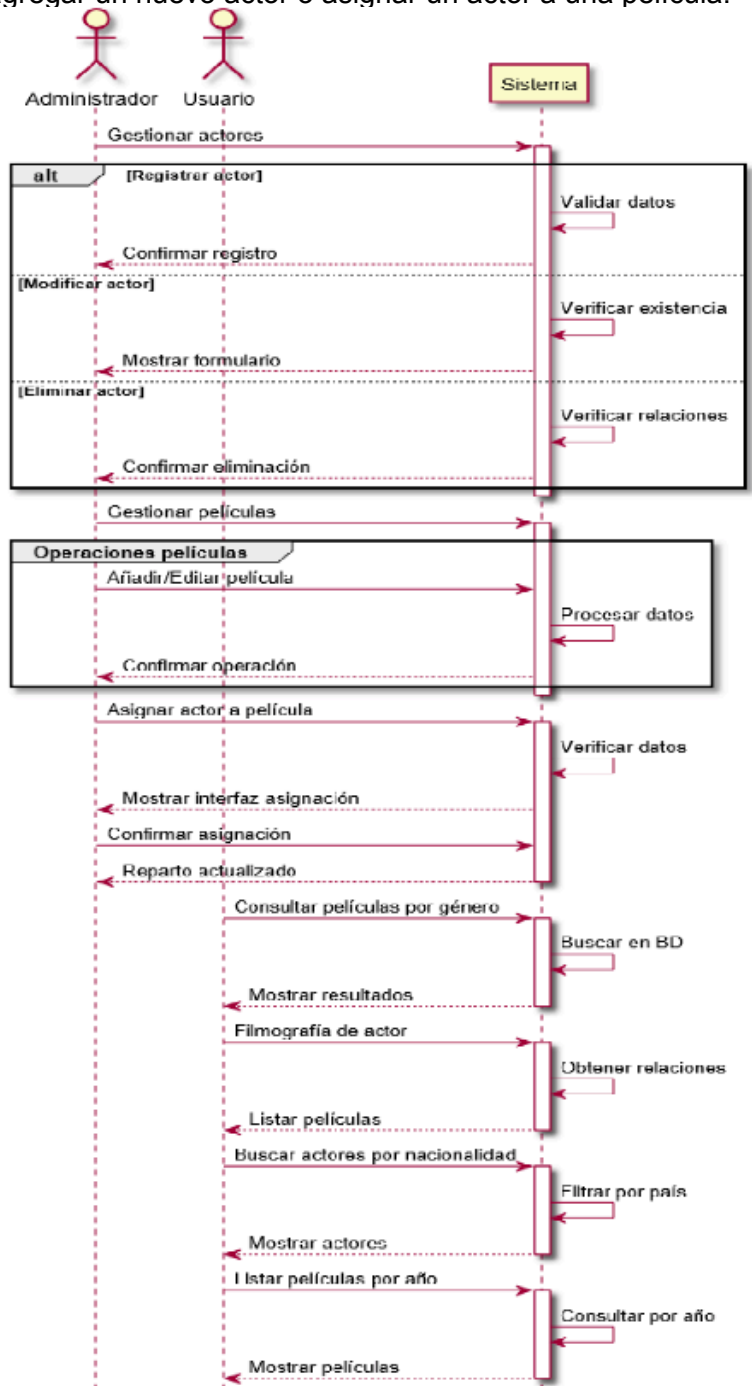
Casos de Uso:
Describe las acciones que pueden realizar los usuarios, como gestionar actores, películas, y el reparto, así como consultar información.



NOMBRE: Filtrar películas según datos
ROL: Usuario
DESCRIPCIÓN: Permite buscar películas según criterios como género, año, actor, etc.
PRECONDICIONES: NO
FLUJO: 1. El usuario ingresa los criterios de búsqueda. 2. El sistema muestra las películas que coinciden con los filtros.
POSTCONDICIONES: Se muestra una lista de películas filtradas.

NOMBRE: Consultar Filmografía y por Género
ROL: Usuario
DESCRIPCIÓN: Permite ver la lista de películas en las que ha participado un actor.
PRECONDICIONES: NO
FLUJO: 1. El usuario selecciona un actor. 2. El sistema muestra su filmografía. 3. El usuario vuelve al menú inicial y selecciona un género. 4. El sistema muestra las películas asociadas.
POSTCONDICIONES: NO

Diagrama de Secuencia:
Muestra cómo interactúan los diferentes elementos del sistema cuando se realiza una acción, como agregar un nuevo actor o asignar un actor a una película.



Resumen de funcionalidades principales

- **Permite gestionar actores**: agregar, modificar y eliminar actores.
- **Permite gestionar películas**: agregar, modificar y eliminar películas.
- **Permite gestionar géneros**: agregar nuevos géneros y consultar películas por género.
- **Permite gestionar el reparto**: asignar actores a películas y modificar sus roles.

Ofrece opciones de búsqueda para encontrar actores por nacionalidad o edad y listar películas por año o género, buscar películas por título, nombre de actor o año de estreno.

Ejemplos concretos de lo que permite hacer el sistema

- Un administrador puede registrar un nuevo actor ingresando su nombre y otros datos.
- Se pueden consultar todas las películas en las que ha trabajado un actor específico.
- Los administradores pueden cambiar la información de una película, como su duración o resumen.
- Se puede asignar un actor a una película y modificar su papel en el reparto.

Usuarios previstos

- **Usuarios sin inicio de sesión**: Pueden buscar información sobre películas y actores.

- **Administrador**: Tiene acceso completo para gestionar toda la información, incluyendo el reparto y poder realizar también consultas.

Hardware y software empleados

- **Hardware:** Java 22 y una base de datos MySQL.
- **Software:** Java, MySQL, y Eclipse.

Estructura de paquetes (DAO, DTO...)

- **DAO**: Son las partes del sistema que se encargan de acceder a la base de datos.
- **DTO**: Son las clases que representan los datos que se manejan en la aplicación.
- **Connection**: Contiene la clase que conecta la base de datos creada con el proyecto a partir de librerías SQL implementadas.
- **Aplicación**: Contiene la clase principal que inicia el sistema.

Uso de colecciones, excepciones, funciones

Uso de Colecciones:

- **ArrayList**: lo uso para almacenar listas de actores, películas y repartos. (**CÓDIGO)**

```
private static List<Actor> actores = new ArrayList<>();
private static List<Pelicula> peliculas = new ArrayList<>();
private static List<Reparto> repartos = new ArrayList<>();
```

- **ResultSet:** lo uso para ejecutar consultas SQL en los métodos de gestión y consultas.

```
ResultSet resultado = statement.executeQuery();
        while (resultado.next()) {
}
```

Uso de Excepciones:
Se manejan excepciones para controlar errores al interactuar con la base de datos. En los métodos que realizan operaciones CRUD, utilice bloques **try-catch** para capturar **SQLException** y mostrar mensajes de error al usuario.

Excepciones en el método crearActor():

```
try {
    PreparedStatement statement = conexion.prepareStatement(
"INSERT INTO actor (nombre, apellidos, año_nacimiento, nacionalidad) VALUES (?, ?, ?,
?)");
    // Código para establecer parámetros y ejecutar la consulta
} catch (SQLException e) {
    System.out.println("Error al crear actor: " + e.getMessage());
}
```

En consultas:

```
        try {
        PreparedStatement statement = conexion.prepareStatement("SELECT * FROM
actor");
        ResultSet resultado = statement.executeQuery(); // Procesa resultados
} catch (SQLException e) {
 System.out.println("Error al consultar actores: " + e.getMessage());
}
```

Validación, si se pone un número incorrecto da error:

```
if (!scanner.hasNextInt()) {
    System.out.println("Error: Debe ingresar un número válido.");
    scanner.nextLine();
    return;
}
```

Funciones:
Las funciones están organizadas en métodos que permiten realizar operaciones específicas como los CRUD en las gestiones de Administrador, **gestionActores(), gestionPeliculas(), gestionReparto()**

Implementar búsquedas específicas: **consultarPeliculasPorAño(), consultarActoresPorNombre(),** etc.

Para salir del sistema: **confirmarSalida()**

Forma de ejecutar la aplicación:
Establece conexión con la base de datos y solicita identificación del usuario o del administrador

```
// Conexion a la base de datos (Comprobación):
private static Connection conexion;

private static final String ADMIN_NAME = "Marina"; // NOMBRE DEL ADMINISTRADOR

public static void main(String[] args) throws SQLException {
    conexion = conexionBaseDatos.getConnection();
    if (conexion != null) {
        System.out.println("Conexión a la base de datos establecida correctamente.");
    } else {
        System.out.println("No se pudo establecer la conexión a la base de datos.");
    }

    System.out.print("\n Ingrese su nombre para acceder al sistema: ");
    String usuario = scanner.nextLine();
    boolean esAdmin = ADMIN_NAME.equalsIgnoreCase(usuario);
```

Muestra el menú de opciones según tipo (administrador o usuario) Captura y valida la selección

```
while (!salir) {
    System.out.println("\n------ SISTEMA DE GESTIÓN CINEMATOGRÁFICA ------");
    if (esAdmin) {
        System.out.println("1. Gestión de Actores");
        System.out.println("2. Gestión de Películas");
        System.out.println("3. Gestión de Reparto");
    }

    System.out.println("4. Consultas");
    System.out.println("5. Salir");
    System.out.print("Seleccione una opción: ");

    int opcion = scanner.nextInt();
    scanner.nextLine();
```

Operaciones: Para cada opción, llama a la función correspondiente y maneja errores

```
switch (opcion) {
    case 1 -> {
        if (esAdmin) gestionActores();
        else System.out.println("Acceso denegado. Solo el administrador puede gestionar actores.");
    }
    case 2 -> {
        if (esAdmin) gestionPeliculas();
        else System.out.println("Acceso denegado. Solo el administrador puede gestionar películas.");
    }
    case 3 -> {
        if (esAdmin) gestionReparto();
        else System.out.println("Acceso denegado. Solo el administrador puede gestionar reparto.");
    }
    case 4 -> realizarConsultas();
    case 5 -> salir = confirmarSalida();
    default -> System.out.println("Opción no válida. Inténtelo de nuevo.");
}
```

Salida: Confirma antes de cerrar en el menú inicial y sale del scanner.

```
        System.out.println("\nSistema cerrado. ¡Hasta pronto!");
        scanner.close();
    }

    /**
     * Confirma si el usuario desea salir del sistema.
     * @return true si desea salir, false en caso contrario
     */
    private static boolean confirmarSalida() {
        System.out.print("\n¿Está seguro que desea salir? (SI/NO): ");
        String respuesta = scanner.nextLine();
        return respuesta.equalsIgnoreCase("SI");
    }
```

Aprendizajes durante el desarrollo

- Aprendí la importancia de manejar errores para que la aplicación funcione sin problemas.
- Puedo entender cómo organizar el código para que sea más fácil de entender y mantener.
- Y pude mejorar un poco la técnica de la conexión entre Java y bases de datos.

Propuestas de ampliación o mejora futura

- Implementar un sistema de autenticación para usuarios sin inicio de sesión.
- Añadir funcionalidades de búsqueda avanzada para películas y actores.
- Integrar una interfaz gráfica de usuario.