

PROYECTO BIMESTRAL: **SISTEMA GESTIÓN CINEMAS LOJA**

Programación Orientada a Objetos

Docente:

- Ing. Pedro Daniel Irene Robalino

Estudiantes:

- Joel Domínguez Ochoa
- Axel Román Torres

Abril-Agosto 2025



UTPL
La Universidad Católica de Loja

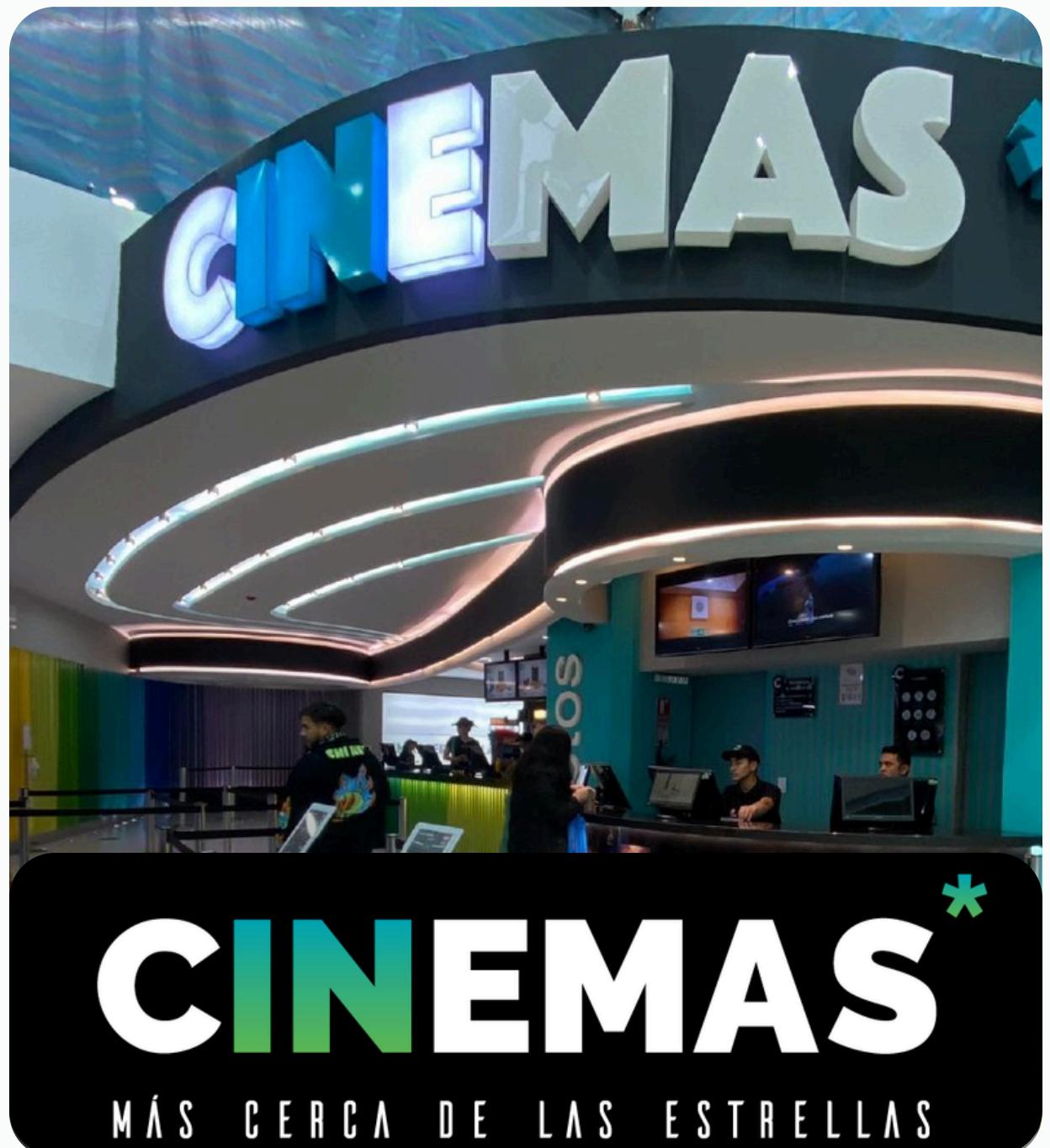


PLANTEAMIENTO DEL PROBLEMA

Desarrollar un sistema de gestión del CineMas-Loja con el objetivo de facturar boletos y snacks para las películas que se proyectan en sus salas. El sistema deberá manejar una cartelera de películas con horarios específicos, así como promociones especiales para determinadas funciones.

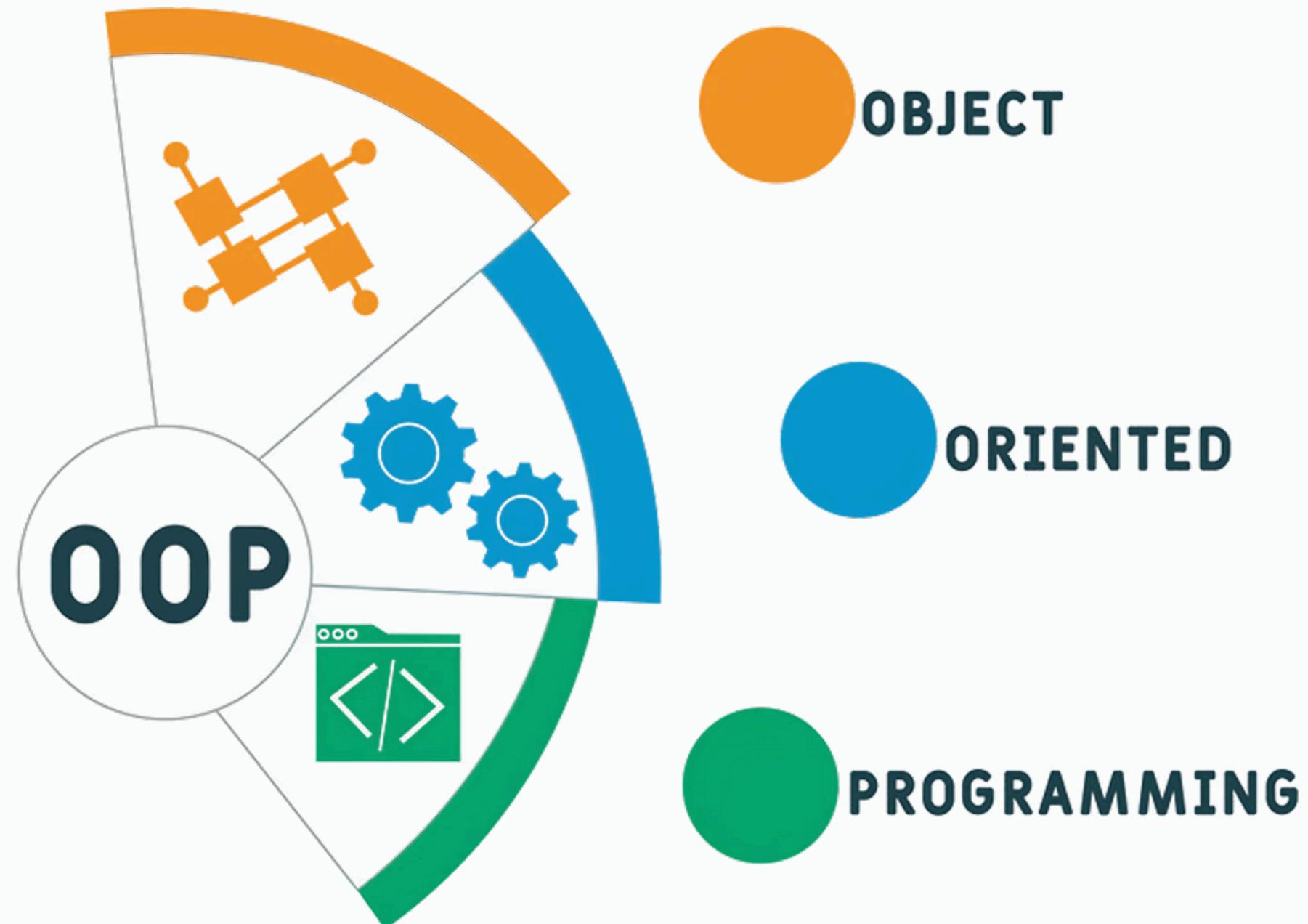
Características para considerar:

- Gestión de cartelera.
- Facturación de boletos.
- Venta de snacks.
- Promociones especiales.
- Registro de ventas.



CINEMAS*
MÁS CERCA DE LAS ESTRELLAS

INTRODUCCIÓN



El proyecto "CineMas" es un sistema de gestión de ventas desarrollado en Java que demuestra la importancia de la **Programación Orientada a Objetos** (POO) y el diseño con UML. Basado en la **arquitectura MVC** y los principios **SOLID**, el sistema garantiza un código limpio y sólido, separando la lógica de negocio (controller), la persistencia de datos en SQLite (model) y la interfaz de usuario (view).

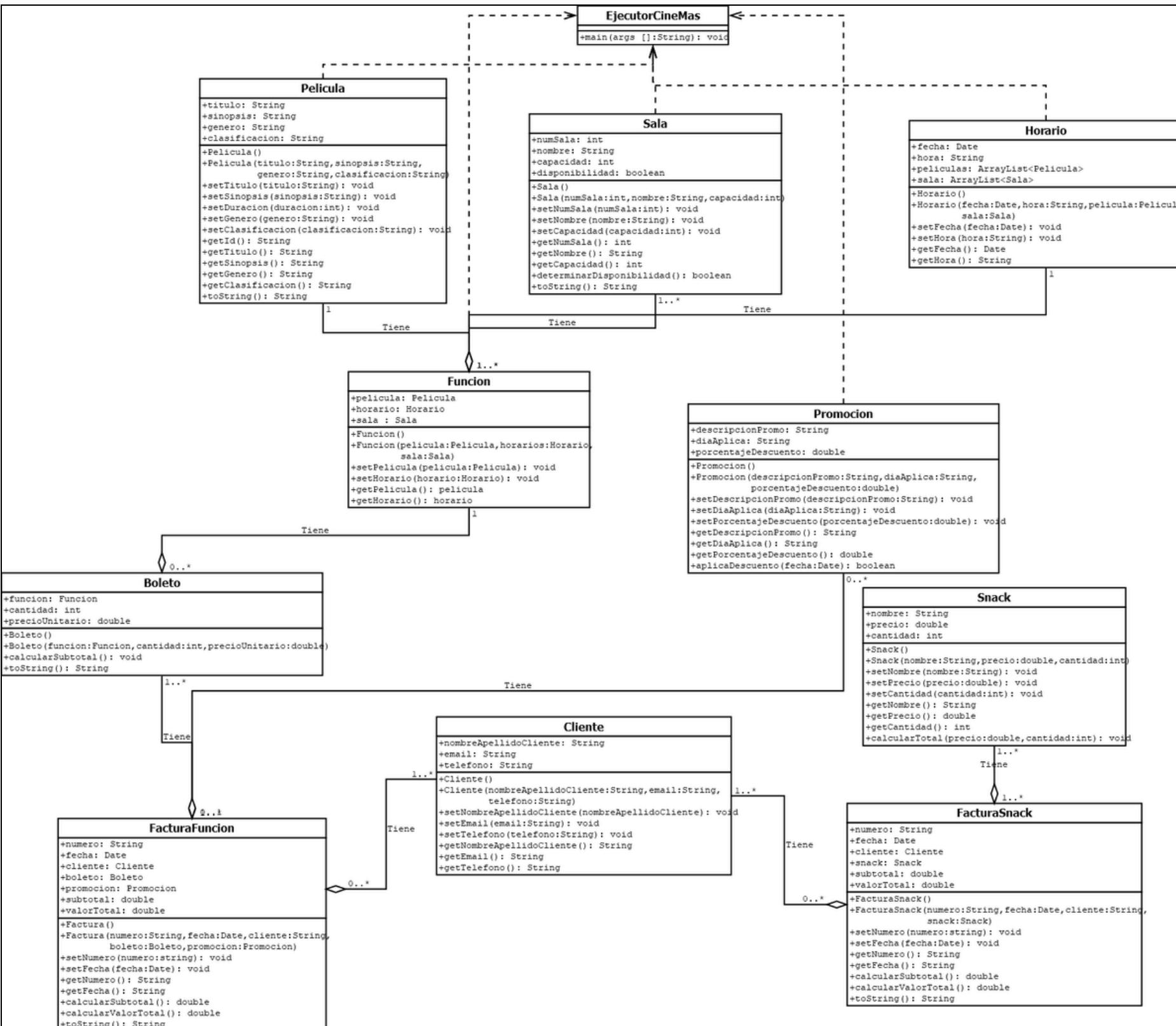
DESCRIPCION DEL PROYECTO

El sistema CineMas es una aplicación de consola que simula un punto de venta para un cine, permitiendo gestionar un horario de funciones dinámico donde las películas se asignan a múltiples salas.

El programa muestra la disponibilidad de asientos y permite un proceso de venta completo, desde la selección de boletos y snacks hasta la generación de facturas. Para garantizar que la información no se pierda entre sesiones, todas las transacciones y el estado de los asientos se guardan de forma persistente en una base de datos SQLite.



UML - PRIMER BIMESTRE



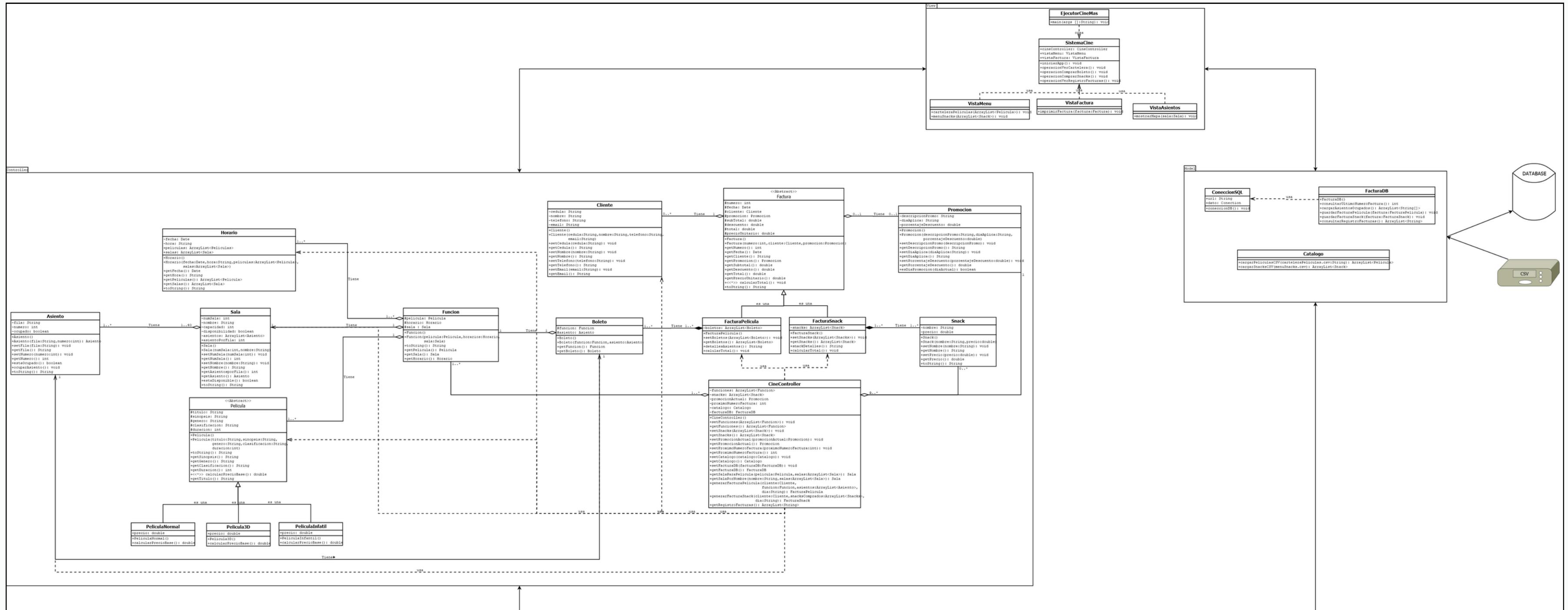
- Falta de Arquitectura MVC.
- Falta Encapsulamiento.
- Falta de Aprovechamiento de la Herencia.
- Bajo uso del Polimorfismo.
- Acoplamiento elevado.
- Ausencia de Principios SOLID

¿QUE MEJORAMOS?





UML - SEGUNDO BIMESTRE



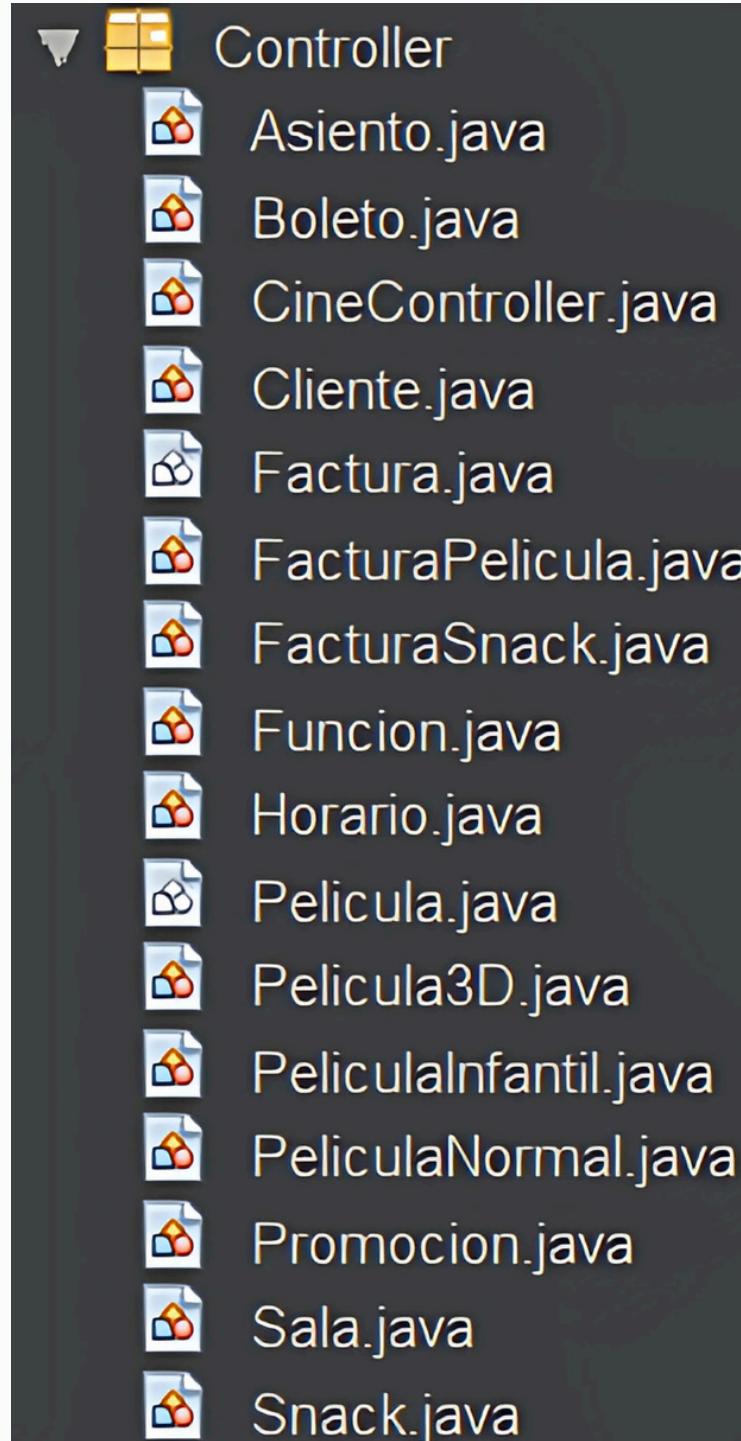


CODIGO

PAQUETE CONTROLLER

Este paquete es el núcleo lógico del sistema, conteniendo las clases de negocio (Pelicula, Factura, etc.) y las reglas que las gobiernan.

La clase CineController gestiona el estado de la aplicación y las operaciones, respondiendo a las solicitudes de la view y usando el model para la persistencia, manteniendo así la lógica de negocio separada de las otras capas.





PAQUETE CONTROLLER

```
public class Boleto implements Serializable{
    protected Funcion funcion;
    protected Asiento asiento;

    public Boleto(Funcion funcion, Asiento asiento) {
        this.funcion = funcion;
        this.asiento = asiento;
    }
    public Funcion getFuncion() {
        return funcion; }
    public AsientogetAsiento() {
        return asiento; }
}
```

```
public class Snack implements Serializable {

    private String nombre;
    private double precio;

    public Snack(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getNombre() {
        return nombre;
    }
    public void setPrecio(double precio) {
        this.precio = precio;
    }
    public double getPrecio() {
        return precio;
    }

    @Override
    public String toString() {
        return String.format("%-25s | Precio: $%.2f", this.nombre, this.precio);
    }
}
```

```
public class Cliente implements Serializable {

    private String cedula;
    private String nombre;
    private String telefono;
    private String email;

    public Cliente(String cedula, String nombre, String telefono, String email) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.telefono = telefono;
        this.email = email;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }
    public String getCedula() {
        return cedula;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getNombre() {
        return nombre;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }
    public String getTelefono() {
        return telefono;
    }

    public void setEmail(String email) {
        this.email = email;
    }
    public String getEmail() {
        return email;
    }
}
```

```
public class Funcion {

    protected Pelicula pelicula;
    protected Sala sala;
    protected Horario horario;

    public Funcion(Pelicula pelicula, Sala sala, Horario horario) {
        this.pelicula = pelicula;
        this.sala = sala;
        this.horario = horario;
    }

    public Pelicula getPelicula() {
        return pelicula;
    }

    public Sala getSala() {
        return sala;
    }

    public Horario getHorario() {
        return horario;
    }

    @Override
    public String toString() {
        return String.format("%-30s | Sala: %-15s | Horario: %s",
            pelicula.getTitulo(), sala.getNombre(), horario.getHora());
    }
}
```



CLASES ABSTRACTA-POLIMORFISMO

```
public abstract class Pelicula implements Serializable {  
  
    protected String titulo;  
    protected String sinopsis;  
    protected String genero;  
    protected String clasificacion;  
    protected int duracion;  
  
    public Pelicula(String titulo, String sinopsis, String genero,  
                    String clasificacion, int duracion) {  
        this.titulo = titulo;  
        this.sinopsis = sinopsis;  
        this.genero = genero;  
        this.clasificacion = clasificacion;  
        this.duracion = duracion;  
    }  
  
    public String getTitulo() {  
        return titulo;  
    }  
  
    public String getSinopsis() {  
        return sinopsis;  
    }  
  
    public String getGenero() {  
        return genero;  
    }  
  
    public String getClasificacion() {  
        return clasificacion;  
    }  
  
    public int getDuracion() {  
        return duracion;  
    }  
  
    public abstract double getPrecioBase();  
  
    @Override  
    public String toString() {  
        return String.format("%-32s | Genero: %-10s | Clas.: %-5s | Dur.: %d min",  
                            this.titulo, this.genero, this.clasificacion, this.duracion);  
    }  
}
```

```
public class Pelicula3D extends Pelicula implements Serializable {  
  
    public double precio;  
  
    public Pelicula3D(String titulo, String sinopsis,  
                      String genero, String clasificacion, int duracion) {  
        super(titulo, sinopsis, genero, clasificacion, duracion);  
        this.precio = 10.00;  
    }  
  
    @Override  
    public double getPrecioBase() {  
        return this.precio;  
    }  
}
```

```
public class PeliculaNormal extends Pelicula implements Serializable{  
  
    public double precio;  
  
    public PeliculaNormal(String titulo, String sinopsis,  
                          String genero, String clasificacion, int duracion) {  
        super(titulo, sinopsis, genero, clasificacion, duracion);  
        this.precio = 7.50;  
    }  
    @Override  
    public double getPrecioBase() {  
        return this.precio;  
    }  
}
```

```
public class PeliculaInfantil extends Pelicula implements Serializable {  
  
    public double precio;  
  
    public PeliculaInfantil(String titulo, String sinopsis,  
                           String genero, String clasificacion, int duracion) {  
        super(titulo, sinopsis, genero, clasificacion, duracion);  
        this.precio = 5.00;  
    }  
  
    @Override  
    public double getPrecioBase() {  
        return this.precio;  
    }  
}
```



CINECONTROLLER.JAVA

Es el componente central que gestiona la operación. Inicializa el estado de la aplicación, expone los métodos de negocio y procesa las solicitudes de la view, traduciendo las acciones del usuario en operaciones de negocio y manteniendo la lógica completamente aislada.

```
public class CineController implements Serializable {
    private ArrayList<Funcion> funciones;
    private ArrayList<Snack> snacks;
    private Promocion promocionActual;
    private int proximoNumeroFactura;
    private Catalogo catalogo;
    private FacturaDB facturaDB;

    public CineController() {
        this.catalogo = new Catalogo();
        this.facturaDB = new FacturaDB();

        this.proximoNumeroFactura = facturaDB.consultarUltimoNumeroFactura() + 1;

        ArrayList<Pelicula> peliculas = catalogo.cargarPelículasCSV("carteleraPelículas.csv");
        this.snacks = catalogo.cargarSnacksCSV("menuSnacks.csv");

        ArrayList<Sala> salas = new ArrayList<>();
        salas.add(new Sala(1, "Sala Premium"));
        salas.add(new Sala(2, "Sala 3D"));
        salas.add(new Sala(3, "Sala Infantil"));

        this.funciones = new ArrayList<>();
        for (Pelicula peli : peliculas) {
            Sala salaAsignada = getSalaParaPelicula(peli, salas);
            Horario horario1 = new Horario(new Date(), "19:30", peli, salaAsignada);
            funciones.add(new Funcion(peli, salaAsignada, horario1));
            if (peli instanceof PeliculaNormal) {
                Horario horario2 = new Horario(new Date(), "22:00", peli, salaAsignada);
                funciones.add(new Funcion(peli, salaAsignada, horario2));
            }
        }
    }
}
```

```
private Sala getSalaParaPelicula(Pelicula pelicula, ArrayList<Sala> salas) {
    if (pelicula instanceof Pelicula3D) {
        return getSalaPorNombre("Sala 3D", salas);
    } else if (pelicula instanceof PeliculaInfantil) {
        return getSalaPorNombre("Sala Infantil", salas);
    } else {
        return getSalaPorNombre("Sala Premium", salas);
    }
}

private Sala getSalaPorNombre(String nombre, ArrayList<Sala> salas) {
    for (Sala sala : salas) {
        if (sala.getNombre().equals(nombre)) {
            return sala;
        }
    }
    return null;
}

public FacturaPelícula generarFacturaPelícula(Cliente cliente, Función función, ArrayList<Asiento> asientos, String día) {
    ArrayList<Boleto> boletos = new ArrayList<>();
    for (Asiento asiento : asientos) {
        asiento.ocupar();
        boletos.add(new Boleto(function, asiento));
    }

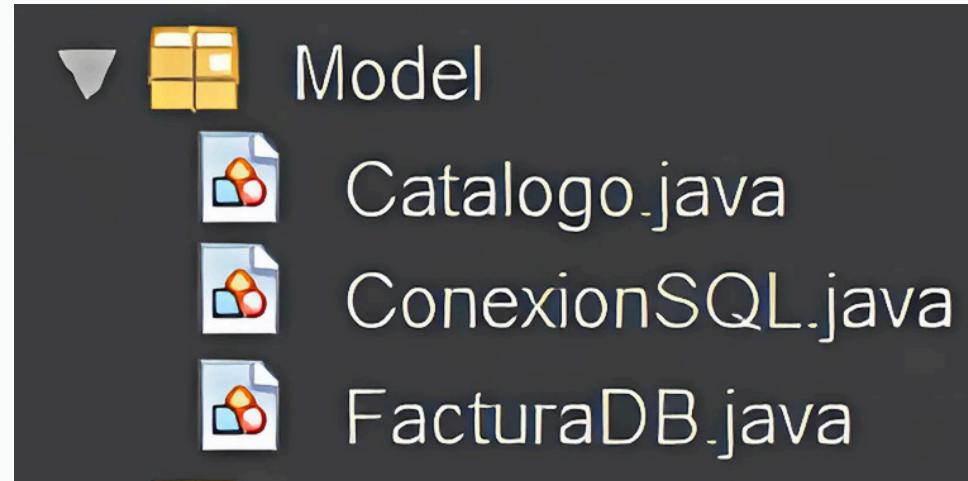
    FacturaPelícula factura = new FacturaPelícula(proximoNumeroFactura, cliente, boletos, this.promocionActual);
    factura.calcularTotal(día);
    facturaDB.guardarFacturaPelícula(factura);

    proximoNumeroFactura++;
    return factura;
}
```



CODIGO

PAQUETE MODEL



Este paquete es la capa de persistencia de datos del sistema, responsable de toda la comunicación con las fuentes de datos (SQLite y archivos CSV), aislando al resto de la aplicación de cómo se guardan o leen los datos.

Contiene clases especializadas como ConexionSQL para la conexión, Catalogo para leer los archivos iniciales, y FacturaDB que guarda y lee toda la información de las transacciones.



CONEXION SQL

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexionSQL {
    private String url;
    public String dr = "\"C:\\\\Users\\\\jcvei\\\\OneDrive\\\\Escritorio\\\\2 CICLO \""
        + "COMPUTACION\\\\AAB2_Proyecto-go4\\\\Solucion_Codigo\\\\CineMas-Loja\\\\"
        + "dataBase\\\\baseDatosCinemas.db\"";
    public ConexionSQL() {
        this.url = "jdbc:sqlite:dataBase/dbCinemasFacturas.db";
    }

    public Connection conectar() {
        try {
            return DriverManager.getConnection(this.url);
        } catch (SQLException e) {
            System.out.println("No se pudo conectar a la base de datos: " + e.getMessage());
            return null;
        }
    }
}
```

Name	Type	Schema
Tables (2)		
FacturaPelicula		CREATE TABLE "FacturaPelicula"
numero	INTEGER	"numero" INTEGER
cliente	TEXT	"cliente" TEXT
email	TEXT	"email" TEXT
telefono	TEXT	"telefono" TEXT
pelicula	TEXT	"pelicula" TEXT
asientos	TEXT	"asientos" TEXT
sala	TEXT	"sala" TEXT
horario	TEXT	"horario" TEXT
cantidadBoletos	INTEGER	"cantidadBoletos" INTEGER
precioUnitario	REAL	"precioUnitario" REAL
subTotal	REAL	"subTotal" REAL
descuento	REAL	"descuento" REAL
total	REAL	"total" REAL
FacturaSnacks		CREATE TABLE "FacturaSnacks"
numero	INTEGER	"numero" INTEGER
cliente	TEXT	"cliente" TEXT
email	TEXT	"email" TEXT
telefono	TEXT	"telefono" TEXT
snack	TEXT	"snack" TEXT
precioUnitario	REAL	"precioUnitario" REAL
subTotal	REAL	"subTotal" REAL
descuento	REAL	"descuento" REAL
total	REAL	"total" REAL



CATALOGO.JAVA

```
public ArrayList<Pelicula> cargarPeliculasCSV(String ruta) {
    ArrayList<Pelicula> lista = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
        br.readLine();
        String linea;
        while ((linea = br.readLine()) != null) {
            String[] datos = linea.split(";");
            if (datos.length < 6) continue;

            String tipo = datos[0].trim();
            String titulo = datos[1].trim();
            String sinopsis = datos[2].trim();
            String genero = datos[3].trim();
            String clasificacion = datos[4].trim();
            int duracion = Integer.parseInt(datos[5].trim());

            Pelicula peli = null;
            if (tipo.equalsIgnoreCase("Normal")) {
                peli = new PeliculaNormal(titulo, sinopsis, genero, clasificacion, duracion);
            } else if (tipo.equalsIgnoreCase("3D")) {
                peli = new Pelicula3D(titulo, sinopsis, genero, clasificacion, duracion);
            } else if (tipo.equalsIgnoreCase("Infantil")) {
                peli = new PeliculaInfantil(titulo, sinopsis, genero, clasificacion, duracion);
            }
            if (peli != null) {
                lista.add(peli);
            }
        }
    } catch (IOException | NumberFormatException e) {
        System.out.println("Error al leer el archivo de peliculas: " + e.getMessage());
    }
    return lista;
}
```

A	B	C
1	Nombre	Precio
2	PopCorn	3
3	Soda Pequeña	1,25
4	Soda Grande	2
5	Granizado	2,5
6	Nachos	3,5
7	HotDog	2,75



A	B	C	D	E	F
1	Tipo	Titulo	Sinopsis	Genero	Clasificacion Duracion
2	Normal	Avengers: End Game	"Heroes intentan revertir el chasquido de Thanos"	Accion	PG-13 180
3	3D	El gato con Botas	"El gato aventurero lucha por su última vida"	Animacion	APT 95
4	Infantil	Lilo y Stitch	"Una niña hawaiana adopta un alienígena"	Infantil	APT 90
5	Normal	los 4 Fantasticos	"Cuatro personas obtienen poderes en el espacio"	Accion	PG-13 100
6	Normal	Superman	"El hombre de acero salva al mundo"	Accion	APT 100
7	3D	Como entrenar a tu dragon	"Un vikingo y su dragón desafían las reglas"	Infantil	APT 90
8	Infantil	ZooTopia 2	"Animales en una ciudad moderna enfrentan nuevos retos"	Infantil	APT 92
9	Normal	Megan 2.0	"Una niña robot con inteligencia peligrosa"	Suspens	PG-15 105



```
public ArrayList<Snack> cargarSnacksCSV(String ruta) {
    ArrayList<Snack> lista = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
        br.readLine();
        String linea;
        while ((linea = br.readLine()) != null) {
            String[] datos = linea.split(";");
            if (datos.length < 2) continue;

            String nombre = datos[0].trim();
            double precio = Double.parseDouble(datos[1].trim().replace(",","."));
            lista.add(new Snack(nombre, precio));
        }
    } catch (IOException | NumberFormatException e) {
        System.out.println("Error al leer el archivo de snacks: " + e.getMessage());
    }
    return lista;
}
```



FACTURADB.JAVA

```
public void guardarFacturaPelicula(FacturaPelicula factura) {
    String sql = "INSERT INTO FacturaPelicula(numero, cliente, email, telefono, "
        + "pelicula, asientos, sala, horario, cantidadBoletos, precioUnitario, subTotal, descuento, total) "
        + "VALUES(?,?,?,?,?,?,?,?,?,?,?,?)";
    ConexionSQL conector = new ConexionSQL();
    try (Connection conn = conector.conectar(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
        if (conn != null) {
            pstmt.setInt(1, factura.getNumero());
            pstmt.setString(2, factura.getCliente().getNombre());
            pstmt.setString(3, factura.getCliente().getEmail());
            pstmt.setString(4, factura.getCliente().getTelefono());
            pstmt.setString(5, factura.getBoletos().get(0).getFuncion().getPelicula().getTitulo());
            pstmt.setString(6, factura.getDetallesAsientos());
            pstmt.setString(7, factura.getBoletos().get(0).getFuncion().getSala().getNombre());
            pstmt.setString(8, factura.getBoletos().get(0).getFuncion().getHorario().getHora());
            pstmt.setInt(9, factura.getBoletos().size());
            pstmt.setDouble(10, factura.getPrecioUnitario());
            pstmt.setDouble(11, factura.getSubtotal());
            pstmt.setDouble(12, factura.getDescuento());
            pstmt.setDouble(13, factura.getTotal());
            pstmt.executeUpdate();
        }
    } catch (SQLException e) {
        System.out.println("Error al guardar la factura de película: " + e.getMessage());
    }
}
```

```
public void guardarFacturaSnack(FacturaSnack factura) {
    String sql = "INSERT INTO FacturaSnacks(numero, cliente, email, telefono, "
        + "snack, precioUnitario, subTotal, descuento, total) "
        + "VALUES(?,?,?,?,?,?,?,?)";
    ConexionSQL conector = new ConexionSQL();
    try (Connection conn = conector.conectar(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
        if (conn != null) {
            pstmt.setInt(1, factura.getNumero());
            pstmt.setString(2, factura.getCliente().getNombre());
            pstmt.setString(3, factura.getCliente().getEmail());
            pstmt.setString(4, factura.getCliente().getTelefono());
            pstmt.setString(5, factura.snackDetalles());
            pstmt.setDouble(6, factura.getPrecioUnitario());
            pstmt.setDouble(7, factura.getSubtotal());
            pstmt.setDouble(8, factura.getDescuento());
            pstmt.setDouble(9, factura.getTotal());
            pstmt.executeUpdate();
        }
    } catch (SQLException e) {
        System.out.println("Error al guardar la factura de snack: " + e.getMessage());
    }
}
```

Numero	Nombre	Apellido	Filter	Apellido	Nombre	Filter	Funcion	Filter	Sala	Filter	Asientos	Filter	Horario	Filter	CantidadBoletos	Filter	PrecioUnitario	Filter	SubTotal	Filter	Descuento	Filter
1	Jose	Velasquez	09654213	Jose	Velasquez	Avengers: End Game	A1, A2, A3	Sala Premium	19:30	3	7.5	22.5	11.									
3	Javier	Jaramillo	09554423	Javier	Jaramillo	Avengers: End Game	B2	Sala Premium	19:30	1	7.5	7.5	3.									
4	Mario	Rojas	098562342	Mario	Rojas	Avengers: End Game	C5, C4, C3	Sala Premium	19:30	3	7.5	22.5	11.									
5	Domenica	Zapata	09562137	Domenica	Zapata	Lilo y Stitch	A1, A2	Sala Infantil	19:30	2	5.0	10.0	5									
9	Sebastian	Ojeda	09854513	Sebastian	Ojeda	Megan 2.0	E1, E2	Sala Premium	22:00	2	7.5	15.0	7									
11	Daniel	Lopez	098842156	Daniel	Lopez	El gato con Botas	A3, A2	Sala 3D	19:30	2	10.0	20.0	10									

Numero	Nombre	Apellido	Filter	Apellido	Nombre	Filter	Funcion	Filter	Sala	Filter	Asientos	Filter	Horario	Filter	CantidadBoletos	Filter	PrecioUnitario	Filter	SubTotal	Filter	Descuento	Filter
1	Valentina	Carrion	09584723	Valentina	Carrion	PopCorn, PopCorn, Soda Pequeña, Soda...													3.0	12.0	0.0	12.0
2	Domenica	Zapata	09562137	Domenica	Zapata	PopCorn, Soda Pequeña													3.0	4.25	0.0	4.25
3	Victoria	Carrion	0954786321	Victoria	Carrion	Nachos, Nachos, Granizado, Soda ...													3.5	10.75	0.0	10.75
4	Paulina	Gonzales	0956217781	Paulina	Gonzales	PopCorn, PopCorn, Soda Pequeña, Soda...													3.0	17.5	0.0	17.5
5	Marco	Castillo	098562314	Marco	Castillo	PopCorn, Soda Pequeña, HotDog, Soda ...													3.0	9.0	0.0	9.0
6	Flor	Velez	09856231	Flor	Velez	PopCorn, Soda Pequeña, Soda Grande, ...													3.0	9.0	4.5	4.5
7	Antonio	Carrillo	0984213645	Antonio	Carrillo	PopCorn													3.0	3.0	1.5	1.5



FACTURADB.JAVA

```
public ArrayList<String> consultarRegistroFacturas() {
    ArrayList<String> registro = new ArrayList<>();
    String sql = "SELECT numero, cliente, total, pelicula AS item FROM FacturaPelicula "
        + "UNION ALL SELECT numero, cliente, total, snack AS item FROM FacturaSnacks ORDER BY numero";
    ConexionSQL conector = new ConexionSQL();
    try (Connection conn = conector.conectar(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            registro.add(String.format("Factura Nro: %05d | Cliente: %-20s | Total: $%-8.2f | Compra: %s",
                rs.getInt("numero"),
                rs.getString("cliente"),
                rs.getDouble("total"),
                rs.getString("item")));
        }
    } catch (SQLException e) {
        System.out.println("Error al consultar registro de facturas: " + e.getMessage());
    }
    return registro;
}
```

```
(4) Ver Registro de Facturas
(5) Salir
>> Seleccione una opcion: 4

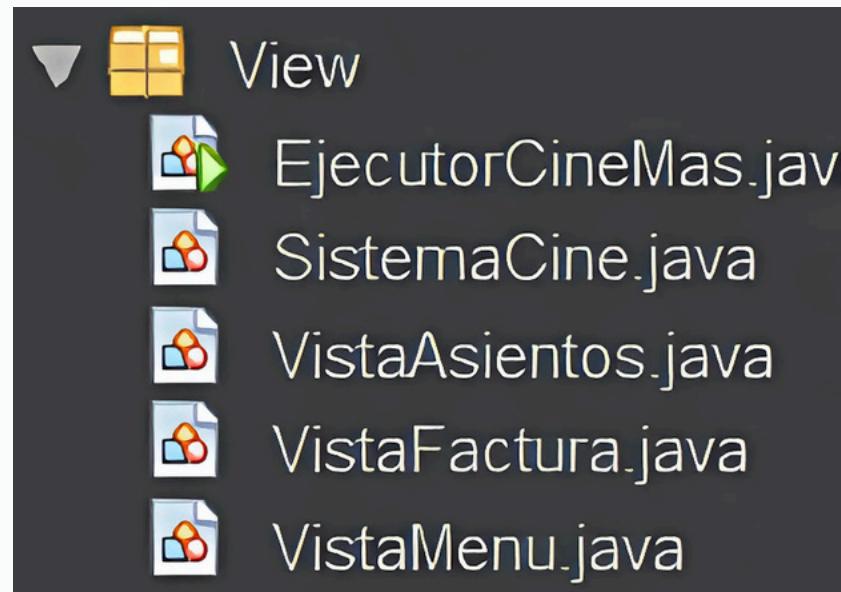
| REGISTRO DE FACTURAS |
Factura Nro: 00001 | Cliente: Jose Velasquez | Total: $11,25 | Compra: Avengers: End Game
Factura Nro: 00002 | Cliente: Valentina Carrion | Total: $12,00 | Compra: PopCorn, PopCorn, Soda Pequeña, Soda Grande, HotDog
Factura Nro: 00003 | Cliente: Javier Jaramillo | Total: $3,75 | Compra: Avengers: End Game
Factura Nro: 00004 | Cliente: Mario Rojas | Total: $11,25 | Compra: Avengers: End Game
Factura Nro: 00005 | Cliente: Domenica Zapata | Total: $5,00 | Compra: Lilo y Stitch
Factura Nro: 00006 | Cliente: Domenica Zapata | Total: $4,25 | Compra: PopCorn, Soda Pequeña
Factura Nro: 00007 | Cliente: Victoria Carrion | Total: $10,75 | Compra: Nachos, Nachos, Granizado, Soda Pequeña
Factura Nro: 00008 | Cliente: Paulina Gonzales | Total: $17,50 | Compra: PopCorn, PopCorn, Soda Pequeña, Soda Pequeña, Nachos, HotDog, HotDog
Factura Nro: 00009 | Cliente: Sebastian Ojeda | Total: $7,50 | Compra: Megan 2.0
Factura Nro: 00010 | Cliente: Marco Castillo | Total: $9,00 | Compra: PopCorn, Soda Pequeña, HotDog, Soda Grande
Factura Nro: 00011 | Cliente: Daniel Lopez | Total: $10,00 | Compra: El gato con Botas
Factura Nro: 00012 | Cliente: Flor Velez | Total: $4,50 | Compra: PopCorn, Soda Pequeña, Soda Grande, HotDog
Factura Nro: 00013 | Cliente: Antonio Carrillo | Total: $1,50 | Compra: PopCorn

Presione Enter para volver al menu principal |> |
```



CODIGO

PAQUETE VIEW



Este paquete es la capa de presentación del sistema, responsable de toda la interacción con el usuario. Su función es mostrar la información y capturar las entradas de forma clara, aislando completamente la interfaz de la lógica de negocio.

Está compuesto por clases especializadas: SistemaCine actúa como el motor que gestiona el menú y el flujo de operaciones, mientras que VistaMenu, VistaAsientos y VistaFactura se encargan únicamente de "dibujar" la información en la consola. La clase EjecutorCineMas sirve como el punto de entrada que inicia la aplicación.



PAQUETE VIEW

```
public class SistemaCine {  
    private CineController controller;  
    private Scanner scanner;  
    private VistaMenu vistaMenu;  
    private VistaAsientos vistaAsientos;  
    private VistaFactura vistaFactura;  
  
    public SistemaCine() {  
        this.controller = new CineController();  
        this.scanner = new Scanner(System.in);  
        this.vistaMenu = new VistaMenu();  
        this.vistaAsientos = new VistaAsientos();  
        this.vistaFactura = new VistaFactura();  
    }  
  
    public void iniciarApp() {  
        int opcion;  
        do {  
            System.out.println("-----");  
            System.out.println("|           MENU PRINCIPAL - CineMas           |");  
            System.out.println("-----");  
            System.out.println("(1) Comprar Boleto(s) de Pelicula");  
            System.out.println("(2) Comprar Snacks");  
            System.out.println("(3) Ver Cartelera de Peliculas");  
            System.out.println("(4) Ver Registro de Facturas");  
            System.out.println("(5) Salir");  
            System.out.print(">> Seleccione una opcion: ");  
        } while (opcion < 1 || opcion > 5);  
        controller.ejecutarOpcion(opcion);  
    }  
}
```

```
private void operacionComprarBoleto() {
    vistaMenu.mostrarFunciones(controller.getFunciones());
    System.out.print("Seleccione el numero de la Pelicula: ");
    int opcionFuncion = Integer.parseInt(scanner.nextLine()) - 1;
    Funcion funcion = controller.getFunciones().get(opcionFuncion);

    System.out.print("Cuantos boletos desea comprar?: ");
    int cantidadBoletos = Integer.parseInt(scanner.nextLine());

    ArrayList<Asiento> asientosSeleccionados = new ArrayList<>();
    for (int i = 0; i < cantidadBoletos; i++) {
        vistaAsientos.mostrarMapa(funcion.getSala());
        System.out.printf("Seleccione asiento para el boleto %d de %d (ej. A5): ", (i + 1), cantidadBoletos);
        String seleccionAsientoStr = scanner.nextLine().toUpperCase().trim();
        String fila = seleccionAsientoStr.substring(0, 1);
        int numero = Integer.parseInt(seleccionAsientoStr.substring(1));
        Asiento asiento = funcion.getSala().getAsiento(fila, numero);

        if (asiento == null || asiento.estaOcupado()) {
            System.out.println("\n[ ERROR ] Asiento no valido u ocupado.");
            return;
        }
        asiento.ocupar();
        asientosSeleccionados.add(asiento);
    }

    System.out.print("Ingrese su nombre: ");
    String nombreCliente = scanner.nextLine();
    System.out.print("Ingrese su email: ");
    String emailCliente = scanner.nextLine();
    System.out.print("Ingrese su telefono: ");
    String telefonoCliente = scanner.nextLine();
    Cliente cliente = new Cliente("9999999999", nombreCliente, emailCliente, telefonoCliente);

    FacturaPelicula factura = controller.generarFacturaPelicula(cliente, funcion, asientosSeleccionados, "Martes");

    vistaFactura.imprimir(factura);
}
```

```
public class VistaAsientos {
    public void mostrarMapa(Sala sala) {
        System.out.println("-----");
        System.out.println("|      SELECCION DE ASIENTOS      |");
        System.out.println("-----");
        System.out.println("          [ PANTALLA ]          ");
        int asientosPorFila = sala.getAsientosPorFila();
        for (int i = 0; i < sala.getAsientos().size(); i++) {
            Asiento asiento = sala.getAsientos().get(i);
            if (i % asientosPorFila == 0) {
                System.out.println();
                System.out.print(asiento.getFila() + "  ");
            }
            System.out.print(asiento.estaOcupado() ? "[X]" : "[ ]");
        }
        System.out.println("\n-----");
    }
}

public class VistaMenu {
    public void mostrarFunciones(ArrayList<Funcion> funciones) {
        System.out.println("-----");
        System.out.println("||           Cartelera de Funciones : CineMas           ||");
        System.out.println("-----");
        for (int i = 0; i < funciones.size(); i++) {
            System.out.printf("%-3d. %s\n", (i + 1), funciones.get(i));
        }
        System.out.println("-----");
    }

    public void mostrarSnacks(ArrayList<Snack> snacks) {
        System.out.println("-----");
        System.out.println("||           SNACKS : CineMas           ||");
        System.out.println("-----");
        for (int i = 0; i < snacks.size(); i++) {
            System.out.printf("%d. %s\n", (i + 1), snacks.get(i));
        }
        System.out.println("-----");
    }
}
```



VISTA DEL PROGRAMA

```
-----  
|      MENU PRINCIPAL - CineMas      |  
-----  
(1) Comprar Boleto(s) de Pelicula  
(2) Comprar Snacks  
(3) Ver Cartelera de Peliculas  
(4) Ver Registro de Facturas  
(5) Salir  
>> Seleccion una opcion: 1  
-----  
||          Cartelera de Funciones : CineMas           ||  
-----  
1 . Avengers: End Game    | Sala: Sala Premium | Horario: 19:30  
2 . Avengers: End Game    | Sala: Sala Premium | Horario: 22:00  
3 . El gato con Botas     | Sala: Sala 3D      | Horario: 19:30  
4 . Lilo y Stitch         | Sala: Sala Infantil | Horario: 19:30  
5 . los 4 Fantasticos    | Sala: Sala Premium | Horario: 19:30  
6 . los 4 Fantasticos    | Sala: Sala Premium | Horario: 22:00  
7 . Superman               | Sala: Sala Premium | Horario: 19:30  
8 . Superman               | Sala: Sala Premium | Horario: 22:00  
9 . Como entrenar a tu dragon | Sala: Sala 3D      | Horario: 19:30  
10 . Zootopia 2            | Sala: Sala Infantil | Horario: 19:30  
11 . Megan 2.0              | Sala: Sala Premium | Horario: 19:30  
12 . Megan 2.0              | Sala: Sala Premium | Horario: 22:00  
-----
```

```
Seleccione el numero de la Pelicula: 4  
Cuantos boletos desea comprar?: 2  
-----  
|      SELECCION DE ASIENTOS      |  
-----  
[ PANTALLA ]  
  
A [X][X][-][-][-][-][-]  
B [-][-][-][-][-][-][-]  
C [-][-][-][-][-][-][-]  
D [-][-][-][-][-][-][-]  
E [-][-][-][-][-][-][-]  
-----  
Seleccione asiento para el boleto 1 de 2 (ej. A5): C1  
-----  
|      SELECCION DE ASIENTOS      |  
-----  
[ PANTALLA ]  
  
A [X][X][-][-][-][-][-]  
B [-][-][-][-][-][-][-]  
C [X][-][-][-][-][-][-]  
D [-][-][-][-][-][-][-]  
E [-][-][-][-][-][-][-]  
-----  
Seleccione asiento para el boleto 2 de 2 (ej. A5): C2  
Ingrrese su nombre: Alejandro Rodriguez  
Ingrrese su email: alejandroRodri@gmail.com  
Ingrrese su telefono: 0988554523
```



VISTA DEL PROGRAMA

```
| MENU PRINCIPAL - CineMas |
|-----|
|(1) Comprar Boleto(s) de Pelicula
|(2) Comprar Snacks
|(3) Ver Cartelera de Peliculas
|(4) Ver Registro de Facturas
|(5) Salir
>> Seleccion una opcion: 2
|-----|
|| SNACKS : CineMas ||
|-----|
1. PopCorn | Precio: $3,00
2. Soda Pequeña | Precio: $1,25
3. Soda Grande | Precio: $2,00
4. Granizado | Precio: $2,50
5. Nachos | Precio: $3,50
6. HotDog | Precio: $2,75
|-----|
Seleccione un snack por numero (o escriba 'fin' para terminar): 1
'PopCorn' agregado al carrito.
Seleccione un snack por numero (o escriba 'fin' para terminar): 3
'Soda Grande' agregado al carrito.
Seleccione un snack por numero (o escriba 'fin' para terminar): 4
'Granizado' agregado al carrito.
Seleccione un snack por numero (o escriba 'fin' para terminar): fin
Ingrese su nombre: Maria Villavicencio
Ingrese su email: mariavilavic19@gmail.com
Ingrese su telefono: 0988745653
```

```
=====
FACTURA CINEMAS
=====
Factura Nro: 00014
Fecha: 22/07/2025 01:56:08
-----
CLIENTE:
Nombre: Alejandro Rodriguez
Cedula: 9999999999
Email: 0988554523
Telefono: Alejandro Rodriguez
-----
DETALLES:
- 2 Boleto(s) para: Lilo y stitch
  Sala: Sala Infantil, Horario: 19:30
  Asientos: c1, c2
-----
SUBTOTAL: $ 10,00
Promo Aplicada: Martes Loco
DESCUENTO: -$ 5,00
-----
TOTAL A PAGAR: $ 5,00
-----
Gracias por su compra!
```

```
=====
FACTURA CINEMAS
=====
Factura Nro: 00015
Fecha: 22/07/2025 02:00:53
-----
CLIENTE:
Nombre: Maria Villavicencio
Cedula: 9999999999
Email: 0988745653
Telefono: Maria Villavicencio
-----
DETALLES:
- Snack: PopCorn $ 3,00
- Snack: Soda Grande $ 2,00
- Snack: Granizado $ 2,50
-----
SUBTOTAL: $ 7,50
Promo Aplicada: Martes Loco
DESCUENTO: -$ 3,75
-----
TOTAL A PAGAR: $ 3,75
-----
Gracias por su compra!
```



CONCLUSIONES

El sistema CineMas cumple con sus objetivos, demostrando una gestión eficiente de ventas. La arquitectura MVC y los principios SOLID han sido clave para lograr un código modular y fácil de mantener.

El programa gestiona correctamente la persistencia de datos en SQLite, conservando el estado de ventas y ocupación de salas entre sesiones.

Como futuras mejoras, se podría implementar una interfaz gráfica (GUI), añadir validaciones de datos más robustas y expandir la lógica de negocio con más funcionalidades.



**MUCHAS GRACIAS
POR VER ESTA PRESENTACIÓN**

