



UNIVERSIDAD NACIONAL DE COLOMBIA

SEDE MEDELLÍN

Manual de Usuario Práctica 2

Grupo 1 - Equipo 4

Programación Orientada A Objetos

Integrantes:

Camilo Echeverri Castrillón
Juan Jose Marín Álvarez
Tomás Gutiérrez Orrego
Sebastián Olaya Pérez
Julián Orrego Martínez

Profesor:

Jaime Alberto Guzmán Luna

2022 - 2S

Contenido

1. Descripción general de la solución	3
1.1 Análisis	3
1.2 Diseño	3
1.3 Implementación	4
1.3.1 Base de datos.	4
1.3.2 Gestor de aplicación	4
1.3.3 Interfaz de usuario (IU)	5
2. Descripción del diseño estático del sistema en la especificación UML:	6
2.1 Usuario	6
2.2 Sala	6
2.3 Día	6
2.4 Película	7
2.5 Boleta	7
2.6 Horario	7
2.7 Tienda	7
2.8 Tienda Comida	7
2.9 Tienda UN	7
3. Descripción de la Implementación de características de programación orientada a objetos en el proyecto	8
3.1 Clase Abstracta, Herencia y Métodos Abstractos	8
3.2 Ligadura dinámica asociadas al modelo lógico de la aplicación.	8
3.4 Atributos de clase y métodos de clase	9
3.5 Uso de constante	9
3.6 Encapsulamiento	10
3.7 Sobrecarga de constructores	10
3.8 Manejo de referencias this para desambiguar	11
4. Descripción de las funcionalidades implementadas	11
4.1 Reembolso	11
4.2 Comprar boleta	11
4.3 Comprar comida	12
4.4 Comprar mercancía	13
4.5 Membresía o hacerse vip	13
5. Manual de usuario	14
Ejecutar	14
Interacción	14
Opción 1. Comprar boleta	14
Opción 2. Comprar comida	16
Opción 3. Comprar mercancía	17
Opción 4. Hacerse miembro VIP	18
Opción 5. Reembolso	18

1. Descripción general de la solución

1.1 Análisis

Al ver múltiples opciones, se encontró que uno de los sistemas de gestión que se ven implementados en la cotidianidad, es el sistema que manejan los cines para la gestión de su funcionamiento, por lo que se vió como una gran oportunidad para la implementación de la programación orientada con un sistema capaz de responder a las necesidades de un cine pequeño, permitiendo en sí que sea un entorno adecuado para aplicar los diferentes aspectos de la programación orientada a objetos.

1.2 Diseño

Para su diseño se tuvo en cuenta desde que el usuario ingresa por una boleta, hasta que hace compra de souvenirs. Además se planteó toda una experiencia de usuario, en la cual, se pueda elegir la película y dentro de la misma los horarios en la que se encuentra, la sala donde se presenta la película, dos tiendas en las cuales podrá comprar los elementos de su preferencia, así como beneficios adicionales.

Se toman en cuenta las siguientes características del cine:

- Hay dos tipos de membresía, siendo la VIP una membresía de pago
- Contará con una disponibilidad de 28 asientos por sala.
- Dispone de 7 productos diferentes para consumir dentro de las salas.
- Las personas con beneficios adicionales (Membresía VIP) contarán con un descuento en el saldo total de la compra.
- El usuario puede solicitar un reembolso antes de que la película se esté rodando.

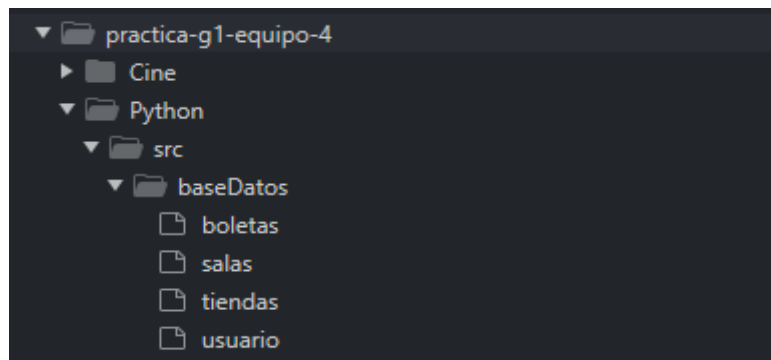
1.3 Implementación

El programa es implementado en su totalidad en lenguaje Python para su compilación y ejecución. También fue implementado el entorno de desarrollo VS Code, Sublime Text, etc.

Esta aplicación está constituida por 3 paquetes:

1.3.1 Base de datos

La base de datos (también conocida como capa de persistencia de datos) mantiene un comportamiento útil para mantener los objetos, que en palabras simples, es la encargada de leer, escribir y borrar objetos en el almacenamiento persistente, además se encuentran los objetos prediseñados para la prueba de la aplicación, lo que es de suma importancia en una aplicación de software. En este paquete encontraremos archivos **.txt** que contienen la serialización de objetos que nos permiten guardar listas. A continuación se anexa la estructura del archivo.

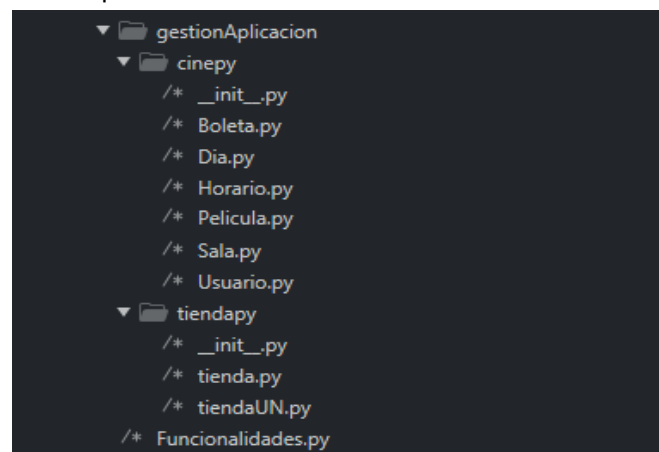


1.3.2 Gestión de aplicación

También conocida como capa lógica, hace parte primordial en la creación de una infraestructura de software, también es la encargada de iniciar la aplicación y derivar los paquetes. Permite la extensión general de servicios para todas las aplicaciones, además es la que engloba las clases usadas para un diagrama UML.

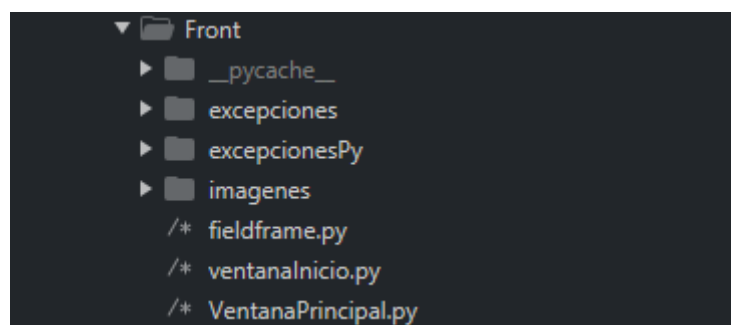
En este caso se decidió separar el **gestorAplicación** de la capa lógica en dos archivos secundarios: **cinepy** y **tiendapy**, cada archivo cuenta con clases, a su vez cada clase contiene atributos y métodos útiles y necesarios para el correcto desarrollo del proyecto y su respectivo CRM (Gestión de relaciones con el cliente).

A continuación se anexa el esquema:



1.3.3 Front o interfaz de usuario (GUI)

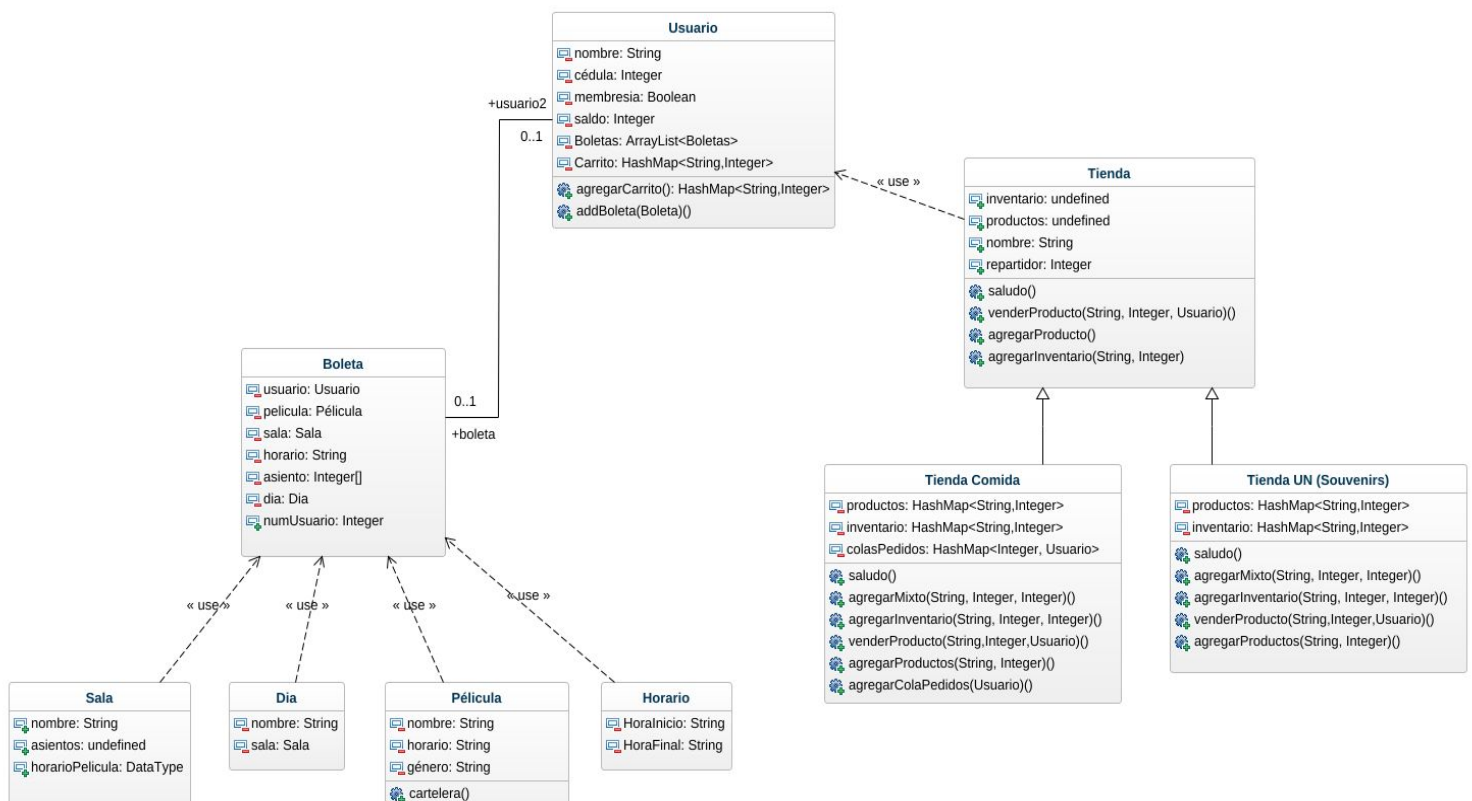
En esta capa se encuentran las funcionalidades del proyecto, en esencia son las clases que se encargan de la implementación de interfaces, las que tienen como función describir la lógica del comportamiento de los métodos. Su objetivo es permitir la interacción entre programa y usuario, mediante un sistema de toma de información.



Menú principal para el usuario.



2. Descripción del diseño estático del sistema en la especificación UML:



2.1 Usuario

- La clase usuario tiene los atributos nombre, cédula, membresía, saldo, boleta, carrito
- El atributo membresía es de tipo booleano, no ayuda a chequear si el usuario es miembro VIP o no.
- El atributo saldo es un Integer, que representa el dinero que tiene a disposición el usuario.
- Boletas, es un ArrayList donde se guardan las boletas que compró el usuario.
- Carrito, es un ArrayList que almacena la comida que encargó el usuario.
- Los métodos *agregarCarrito* y *addBoleta* se encargan de alimentar los atributos Boleta y Carrito del usuario.

2.2 Sala

- El atributo nombre es el nombre de la sala.
- Asiento es un HashMap con
- Horario es un ArrayList con dos elementos de tipo Horario, que indican la hora de inicio y fin de la película.

2.3 Día

- Es un atributo que indica el nombre del día de la semana.
- Es un atributo que almacena la sala que está disponible ese día.

2.4 Película

- Es el nombre de la película.
- El atributo horario es un HashMap de key nombre de la película y de value el horario en el que se presenta.
- Es el género de la película.

2.5 Boleta

- Los atributos de la clase boleta están compuestos por sus simils de clase.
- Dentro está clase queda constatada toda la información del método *comprarEntrada*.

2.6 Horario

- Tiene dos únicos atributos, HoraInicio y HoraFinal. Estos dos atributos se encargan de establecer el tiempo en el que se presenta una película.

2.7 Tienda

- Es una *clase abstracta* que es la base de las demás tiendas.
- Entre sus métodos abstractos se encuentran *saludo* y *venderProducto*.
- Entre sus atributos de instancia se encuentran nombre, inventario y productos.

2.8 Tienda Comida

- Productos es el atributo que almacena los productos en un HashMap <String,Integer> (nombre del producto y precio).
- Inventario es el atributo que almacena los inventario en un HashMap <String,Integer> (nombre del producto y precio).

- Colas pedidos en un atributo que almacena los pedidos encargados en la funcionalidad *PedirAsiento*, lleva la cuenta de los pedidos con su respectivo usuario en un HashMap <Integer, Usuario>.
- Contiene funciones para agregar elementos a los diferentes atributos de Inventario y Productos.

2.9 Tienda UN

- Esta clase contiene los productos UNAL, su funcionamiento es similar al de *Tienda Comida*, con la excepción de que esta no tiene pedidos externos.
- Entre sus funciones están, venderProducto, agregarMixto que se encarga de agregar un producto al inventario y establecer su precio, llamada a la funciones agregarInventario y agregarProducto.

3. Descripción de la Implementación de características de programación orientada a objetos en el proyecto

3.1 Clase Abstracta, Herencia y Métodos Abstractos (En java y python)

```
package gestorAplicacion.tiendaAbst;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;
import gestorAplicacion.cine.Usuario;

public abstract class Tienda implements Serializable{

    private static final long serialVersionUID = 1L;

    protected String nombre;

    protected Map<String, Integer> inventario = new HashMap<String, Integer >();

    protected Map<String, Integer> productos = new HashMap<String, Integer >();

    protected int repartidores = 3;

    abstract public void saludo();

    abstract public void venderProducto(String nombre, Integer cantidad, Usuario usuario);
```

La clase tienda se creó para que tanto la tienda de comida y la tienda UN con souvenirs, tengan que implementar los métodos abstractos que se muestran (saludo y vender producto),

3.2 Ligadura dinámica asociadas al modelo lógico de la aplicación.

```
@Override
public void saludo() {
    System.out.println("Bienvenido a la tienda del cine UNAL");
}
```

```
@Override
public void saludo() {
    System.out.println("Bienvenido a la tienda de comida del cine UNAL");
}
```

```
public void venderProducto(String nombre, Integer cantidad, Usuario usuario) {
    if (inventario.get(nombre) >= cantidad) {
        inventario.put(nombre, inventario.get(nombre) - cantidad);
        if (usuario.verificarMembresia()) {
            usuario.setSaldo(usuario.getSaldo() - (int)((cantidad * inventario.get(nombre))*0.4));
        }
        else {
            usuario.setSaldo(usuario.getSaldo() - (cantidad * inventario.get(nombre)));
        }
    }
}
```

```
public void venderProducto(String nombre, Integer cantidad, Usuario usuario) {
    if (inventario.get(nombre) >= cantidad) {
        inventario.put(nombre, inventario.get(nombre) - cantidad);
        if (usuario.verificarMembresia()) {
            usuario.setSaldo(usuario.getSaldo() - (int)((cantidad * inventario.get(nombre))*0.3));
        }
        else {
            usuario.setSaldo(usuario.getSaldo() - (cantidad * inventario.get(nombre)));
        }
    }
}
```

Al irse a los métodos de cada tienda dependiendo del llamado, sea en tiendaUN o en tiendaComida se estaría aplicando la ligadura dinámica con los métodos saludo y venderProducto que aplicarían descuentos diferente.

3.4 Atributos de clase y métodos de clase


```

public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;

    private String Nombre;
    private int Cedula;
    private boolean Membresia;
    private int Saldo;
    private ArrayList<Boleta> Boletas = new ArrayList<Boleta>();
    private HashMap<String,Integer> Carrito = new HashMap<String,Integer>();
    private static int cantidadUsuarios;

    public static int getCantidadUsuarios() {
        return cantidadUsuarios;
    }
}

```

3.5 Uso de constante

```

public class Boleta implements Serializable{

    private static final long serialVersionUID = 1L;

    private Usuario usuario;
    private String pelicula;
    private Sala sala;
    private Horario horario;
    private String dia;
    //attribute Asiento as an array of Integers
    private int[] asiento;
    public final static int precio = 10000;
}

```

Se ve como se maneja como una constante el precio de las boletas del cine ya que solo cambiaría en main al un usuario volverse VIP

3.6 Encapsulamiento

```
public abstract class Tienda implements Serializable{

    private static final long serialVersionUID = 1L;

    protected String nombre;

    protected Map<String, Integer> inventario = new HashMap<String, Integer >();

    protected Map<String, Integer> productos = new HashMap<String, Integer >();

    protected int repartidores = 3;
```

Se puede ver como las veces en que se necesitan heredar los atributos se trabaja con protected y de resto con private

3.7 Sobrecarga de constructores

```
public Usuario() {this("Sin nombre", 0, 0);}

public Usuario (String Nombre, int Cedula, int Saldo) {
    this.Nombre = Nombre;
    this.Cedula = Cedula;
    this.Membresia = false;
    this.Saldo = Saldo;
}
```

Esto en caso de que el usuario no quiera identificarse.

```
public Sala ( Map<String, boolean[]> Asientos, Map<Pelicula, Horario> cartelera) {
    this.nombre = "Sala Unal";
    this.Asientos = Asientos;
    this.cartelera = cartelera;
}

public Sala (String nombre, Map<String, boolean[]> Asientos, Map<Pelicula, Horario> cartelera) {
    this.nombre = nombre;
    this.Asientos = Asientos;
    this.cartelera = cartelera;
}
```

En caso de que creen una sala sin nombre.**3.8 Manejo de referencias this para desambiguar**

```
class FieldFrame(Frame):
    def __init__(self, ventana, tituloCriterios = "", criterios = None, tituloValores
        super().__init__(ventana) #Lista de nombres de criterios
        self._tituloCriterios = tituloCriterios
        self._criterios = criterios
        self._tituloValores = tituloValores
        self._valores = valores
        self._habilitado = habilitado
```

Por ejemplo acá para diferenciar los que pasan por parámetro a los de la instancia

4. Descripción de las funcionalidades implementadas

4.1 Reembolso

La funcionalidad Reembolso es la encargada de reembolsar una boleta del usuario si así lo quiere, para ello depende del parámetro *usuario* de tipo Usuario, que se pasa por referencia.

Lo primero que hace el método es verificar si el usuario efectivamente tiene boletas bajo su nombre, si pasa la verificación muestra por pantalla las películas y las boletas que tiene para cada película, esto lo hace accediendo por medio de usuario a su atributo *boleta*.

Luego de imprimir las boletas se le pide al usuario que escoja el nombre de la película y el asiento que desea cancelar, se hace una nueva lista de booleanos que replica la lista anterior omitiendo el asiento reembolsado, se crea un nuevo HashMap para almacenar la película para luego reemplazarla en la original reemplazando el anterior y dejando libre la boleta reembolsada, luego de eso se le elimina el asiento de la boleta y si esta queda sin asientos se le eliminará igualmente.

Se le envía un parámetro de tipo usuario, desde de este se ingresa a boleta y se utilizan los sets y gets de asiento para modificarla, también es utilizada para acceder desde usuario al saldo y al efectuar el reembolso y modificarlo, la mayor parte de las interacciones son con la clase boletas, ya que esta guarda la información más importante como lo es, el nombre de la película, asientos reservados.

4.2 Comprar boleta

La compra de la boleta básicamente es el medio por el cual el usuario puede elegir el horario de la película y elegir un asiento a gusto, además si el usuario cuenta con membresía, se le realizará el descuento en su compra. Para el desarrollo se requirió una interacción con Usuario y Día, uno para obtener el booleano que define el estado de la membresía del usuario y el otro para la selección del horario/día respectivamente, es importante resaltar que cada película está asignada a un horario, por lo tanto esa interacción con Día.

Son varios los objetos creados al comprar una boleta; *Pelicula* crea los objetos *pelicula* el cual recibe de argumentos el nombre de la película y el género al que pertenece, luego *Horario* crea los objetos *horarios*, los cuales reciben argumentos de hora de inicio y hora de fin, luego se crean las carteleras, las cuales son diccionarios que permiten ubicar una película con el horario en el que se presenta, luego se crea un nuevo arreglo de booleanos que permite ubicar los asientos, luego se crean las salas que reciben los parámetros de Nombre, el cual es "Sala", los asientos y la cartelera de películas, por último se establecen los días y se crea un array en donde los ingresa, luego el método *comprarEntrada* verifica primero si se es VIP y luego pregunta cuántas entradas desea comprar, luego el día en que quiere ver la películas, allí se le muestran las películas que se ofrecen ese día y se pide ingresar el nombre, de ésta y luego que escoja los asientos según las entradas que quiso comprar, luego confirma el asiento y le muestra la factura que contiene el saldo actual.

4.3 Comprar comida

En esta funcionalidad el usuario comprador podrá adquirir todos los productos que desee consumir, además podrá ver en el menú tanto de las comidas como su precio. Este proceso se hace por medio de una transacción que descontará el saldo del usuario dependiendo del costo de cada producto, si el comprador no tiene el saldo suficiente no podrá comprar ningún producto. Al realizar dicha compra se imprime la información del usuario, su saldo restante y cantidad del producto.

TiendaComida crea un nuevo objeto *tiendacomida* al cual se le pasa como argumento un nombre, en este caso "Tienda de comida"; luego al ser una clase que hereda de *TiendaUN*, que a su vez hereda de *Tienda*, la cual posee el método *setNombre*, se le cambia éste a "Comida Unal"; luego se añaden los productos que se desean vender por medio del método *agregarMixto* al cual se le agregan los 7 productos ofrecidos con los parámetros, nombre, cantidad máxima de productos ofrecidos por el cine y el precio de cada producto; luego desde el método *comprarComida* se muestran los productos para que el usuario pueda comprar lo que desee.

4.4 Comprar mercancia

En esta funcionalidad el usuario comprador podrá adquirir todos los productos que desee llevar relacionados con la UN, además podrá ver en el menú tanto de los productos como su precio. Este proceso se hace por medio de una transacción que descontará el saldo del usuario dependiendo del costo de cada producto, si el comprador no tiene el saldo suficiente no podrá comprar ningún producto. Al realizar dicha compra se imprime la información del usuario, su saldo restante y cantidad del producto.

TiendaUN crea un nuevo objeto *tiendaUN* al cual se le pasa como argumento un nombre, en este caso "Tienda UN"; luego se añaden los productos que se desean vender por medio del método *agregarMixto* al cual se le agregan los 4 productos ofrecidos con los parámetros, nombre, cantidad máxima de productos ofrecidos por el cine y el precio de cada producto; luego desde el método *comprarMercancia* se muestran los productos para que el usuario pueda comprar lo que desee.

4.5 Membresía o hacerse vip

Con la intención de mejorar la experiencia del usuario en cuanto a costos, se implementa una función en la cual el usuario podrá suscribirse por un valor de 30000, mostrando por pantalla los descuentos tanto de la tienda Un como de la tienda de comida, por lo que al volverse VIP, al ingresar de nuevo tendrá descuentos en las demás funcionalidades.

Usamos las clases *Usuario*, *TiendaComida* y *TiendaUN* para esta funcionalidad.

Con *Usuario* obtenemos la información del usuario sobre el estado de su suscripción, si este ya se encuentra suscrito imprime un mensaje haciéndole saber al usuario. De lo contrario, usa las clases *TiendaComida* y *TiendaUN* para imprimir todos los beneficios que tendría en el precio de los productos y además de un pequeño descuento en el costo de la boleta. Por consola, le pregunta si quiere adquirir estos beneficios. Si la respuesta es afirmativa, el método va al saldo del usuario y le descuenta el valor de \$30.000, que es lo que cuesta la suscripción, si la respuesta es negativa agradece al usuario por su tiempo y lo manda al menú.

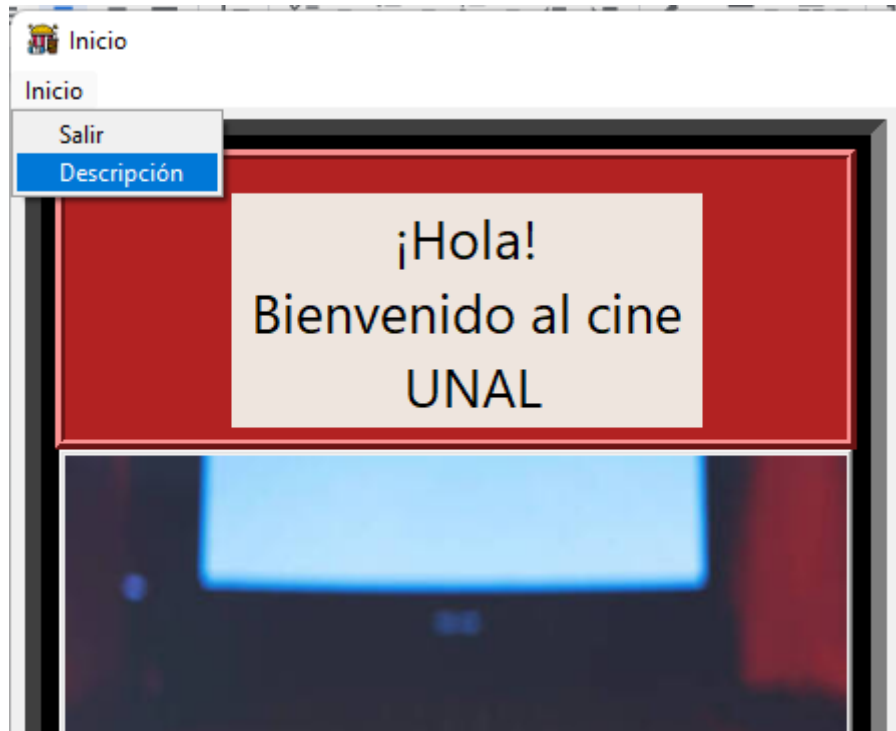
5. Manual de usuario

Menú principal



Al ejecutar la aplicación, encontrarás una dos ventanas y un botón de interacción en la parte superior izquierda que dice **Inicio**:

- Botón de inicio:



El menú dispone de las opción salir y descripción, una para abandonar la aplicación y otra para obtener una breve descripción del producto respectivamente.

- Ventana derecha:



Al clicar “Los integrantes del grupo 4” se desplegarán 4 fotografías dando a conocer una breve presentación de los desarrolladores del cine, añadiendo una breve descripción.

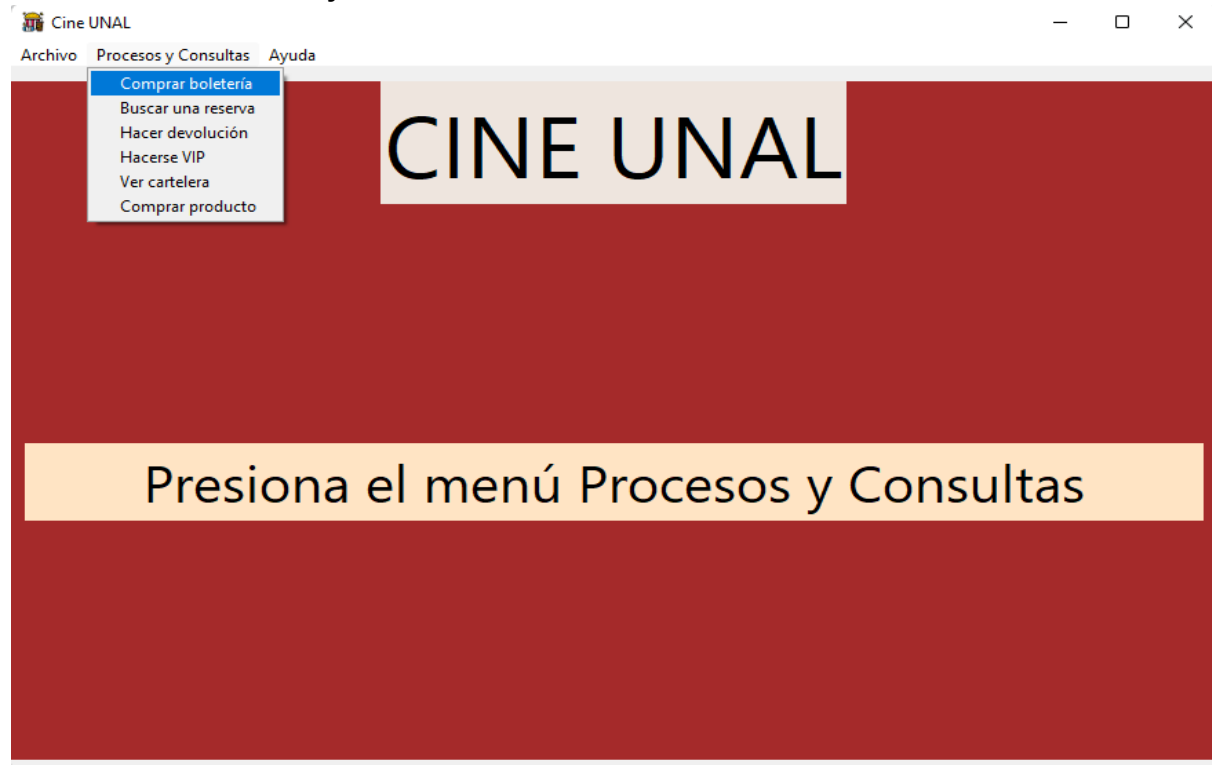
- Ventana izquierda:

Inicio



La ventana izquierda es el inicio del menú principal, al clicar se ejecutará una nueva ventana con los menús desplegables a continuación:

Menú de inicio Procesos y Consultas



Al ingresar al menú de inicio, se puede desplegar las opciones de Procesos y Consultas donde se puede realizar el trámite deseado.

- Comprar boletería:

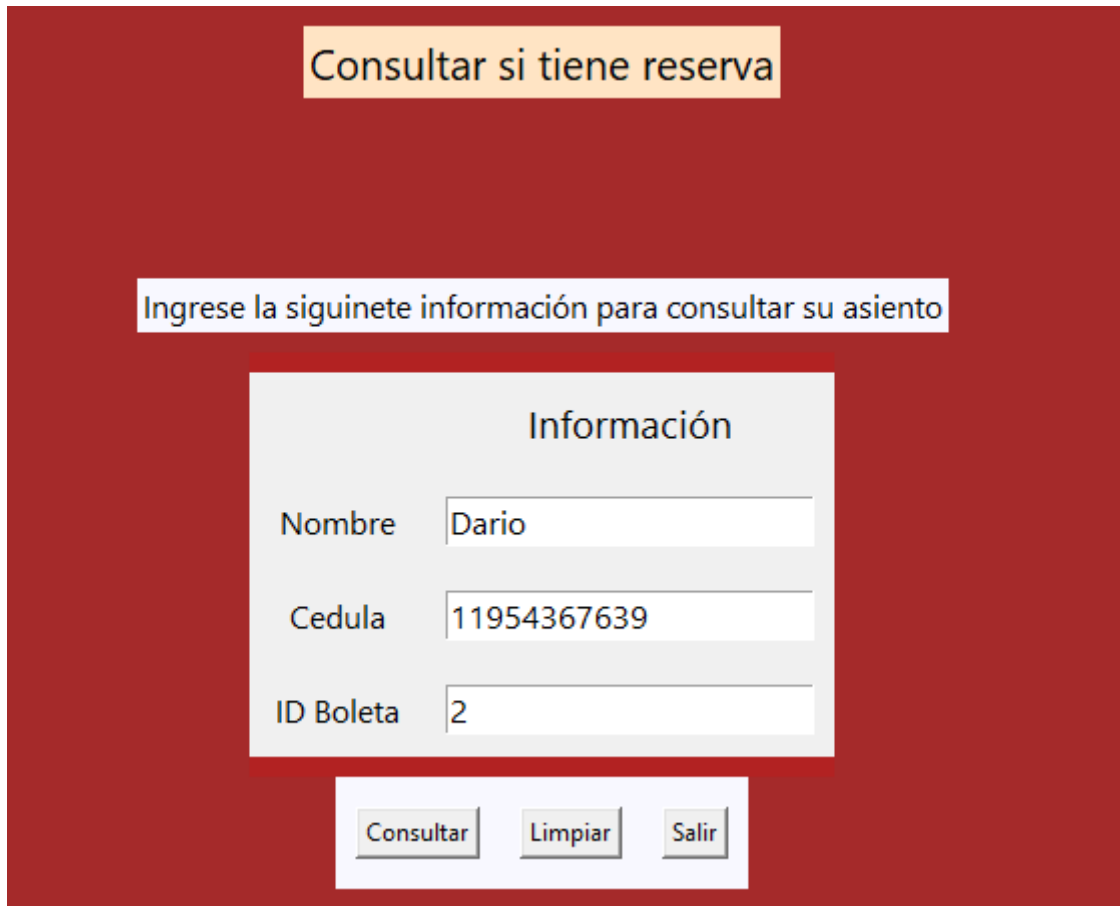
The screenshot displays the 'Comprar tus boletas' form. At the top, a yellow button says 'Compra tus boletas'. Below it, a white box contains the instruction 'Digite los campos para comprar las boletas'. The form itself is titled 'Información Personal' and contains the following fields:

Información Personal	
Nombre	Dario
Cedula	11954367639
Cantidad de boletas	2
Dia de la funcion	Martes
Nombre de la pelicula	Avengers

At the bottom of the form are three buttons: 'Comprar', 'Limpiar', and 'Salir'. To the right of the form, a small notification window titled 'Compra de boletas' shows a blue information icon and the text 'Compra exitosa', with an 'Aceptar' button.

La sección Compra tus boletas es el espacio para ingresar tus datos, la cantidad de boletas, día y nombre de la película tal cual como se muestra en la imagen. Al rellenar los datos si se desea hacer la compra se debe clicar en “Comprar”.

- Buscar una reserva:



Consultar si tiene reserva

Ingrese la siguiente información para consultar su asiento

Información	
Nombre	<input type="text" value="Dario"/>
Cedula	<input type="text" value="11954367639"/>
ID Boleta	<input type="text" value="2"/>

Al rellenar los campos es necesario que cada uno de los datos coincidan con los datos del comprador de la boleta, incluida la ID de la boleta, tal cual como se muestra. Al finalizar se presiona Consultar si se desea consultar los datos de la boleta.

- Hacer devolución:

The screenshot shows a web form titled "Solicita la devolución" on a red background. Below the title is a white instruction box: "Ingrese su documento para saber si se puede realizar la devolución de su dinero". The form itself is a light gray box with two input fields: "Cedula" with the value "1193436739" and "ID Boleta" with the value "2". At the bottom of the form are three buttons: "Devolución", "Limpiar", and "Salir".

Al rellenar los campos (se debe especificar la ID de la boleta) se debe presionar el botón Devolución.

- Hacerse VIP:

The screenshot shows a web form titled "Hazte miembro VIP" on a red background. Below the title is a white instruction box: "Ingrese su documento para volverse VIP". The form is a light gray box with a single input field labeled "Cedula". At the bottom of the form are three buttons: "VIP", "Limpiar", and "Salir".

El menú VIP da pie a una credencial que genera descuentos en las compras de los diferentes productos del cine, solo se debe rellenar el campo Cédula y presionar el botón VIP para acceder a comprar la membresía.

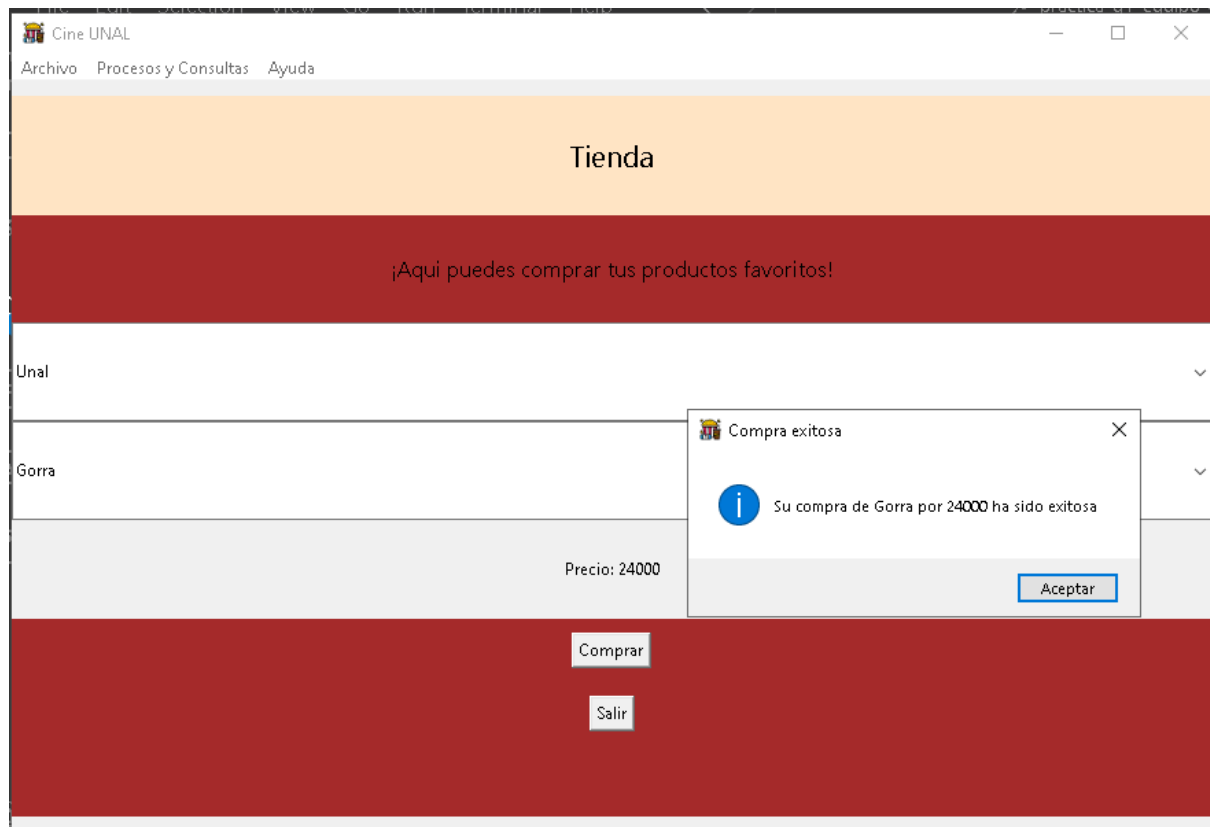
- Ver Cartelera:

The screenshot shows a web application window titled 'Cine UNAL'. It has a menu bar with 'Archivo', 'Procesos y Consultas', and 'Ayuda'. The main content area is divided into two columns. The left column has a header 'Por favor seleccione el día' with a dropdown menu showing 'Jueves'. Below this is a red box with the text 'La Cartelera el día de hoy' and a 'Salir' button at the bottom. The right column has a dark red header with the text 'Jueves' and a light red box below it containing the following movie schedule:

Película	Horario
RANGO	14:00-16:00
EL PASEO	16:00-20:00
HALLOWEEN	20:00-22:00
REVENANT	22:00-24:00

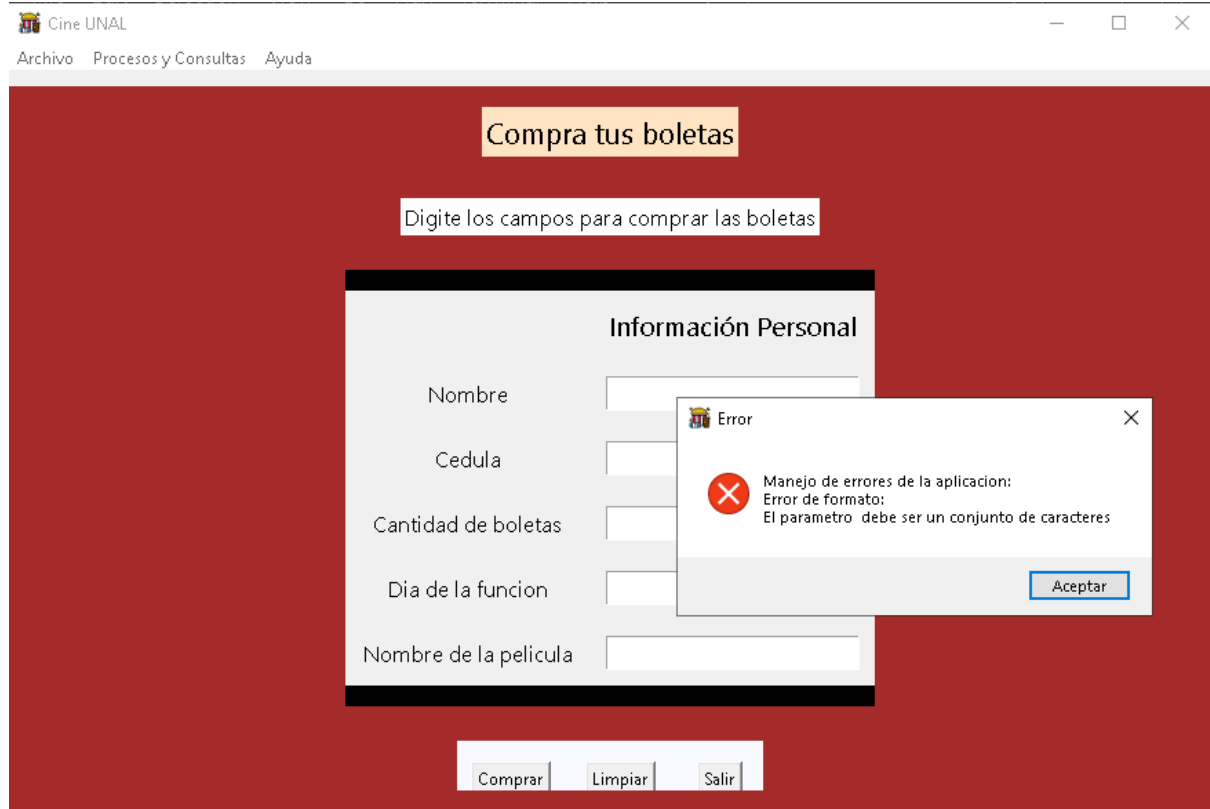
El menú Ver Cartelera permite visualizar entre las diferentes opciones de películas con sus respectivas fechas.

- Comprar producto

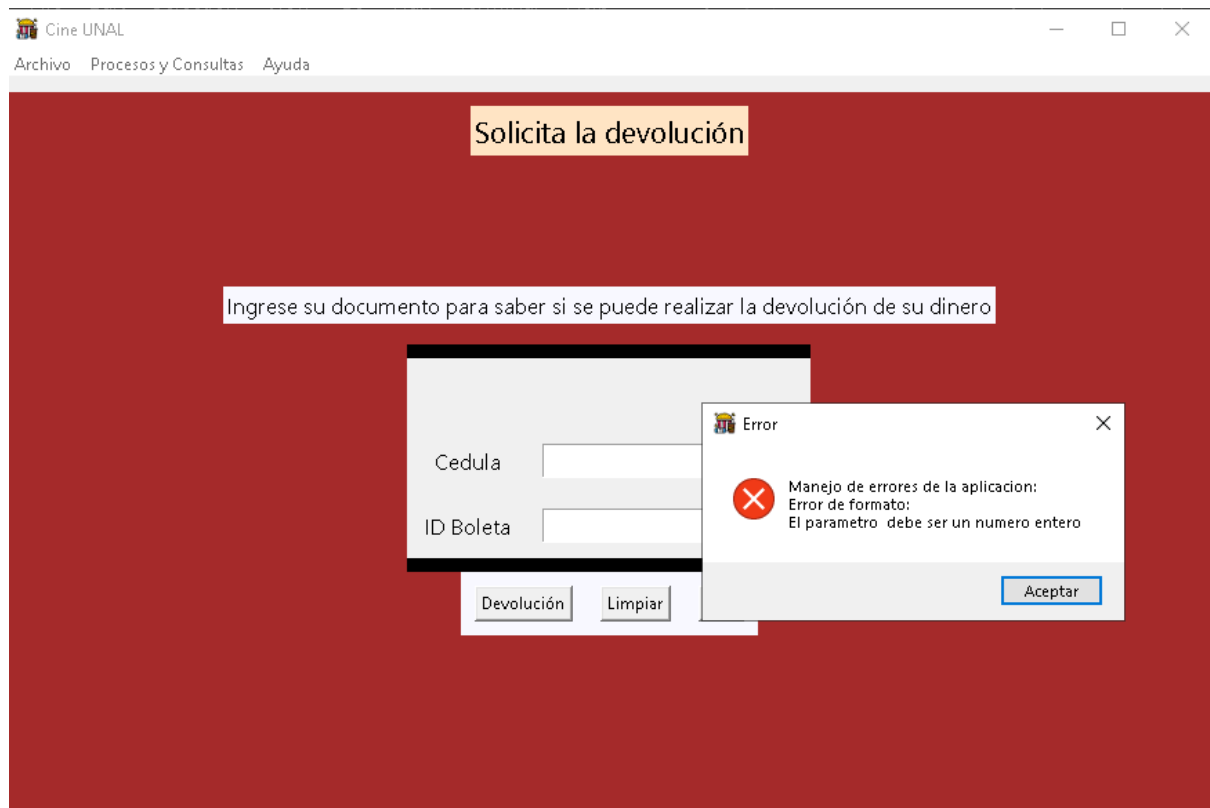


Al querer realizar una compra, se debe seleccionar el tipo de tienda y el producto para que salga su precio. Al finalizar se debe seleccionar el botón Comprar y se efectuará el pago.

Excepciones



Cuando no se ingresa los caracteres cuando son requeridos aparecerá este error.



Cuando no se ingresa un número entero cuando es requerido aparecerá este error.