



**Universidad Tecnológica del Perú**

**Facultad de Ingeniería**

**Carrera de Ingeniería de Sistemas y Software**

**“Sistema de Venta como soporte al proceso de gestión  
de ventas en la empresa Tambo en el año 2023”**

**Integrantes**

Yakelin Nicol Zavaleta Ramírez

Joaquín Miguel Segovia Mejía

Nelson Fabrizzio Pereira Vásquez

Sebastian Salomé Murillo

Fabricio Jefferson Salazar Ramos

**Curso:** Programación Orientada a Objetos

**Docente:** Ing. Hernán Peña Carnero

**Lima – Perú**

**2023 - I**

# ÍNDICE

<b>1. ASPECTOS GENERALES .....</b>	<b>3</b>
1.1. Introducción de la empresa .....	3
1.2. Datos generales de la empresa .....	3
1.3. Definición del problema .....	3
1.4. Definición de objetivos .....	4
<b>2. MARCO TEÓRICO .....</b>	<b>5</b>
2.1. Antecedentes .....	5
2.2. Fundamento Teórico .....	6
<b>3. DESARROLLO DE LA SOLUCIÓN .....</b>	<b>8</b>
3.1. Lista de Requerimientos funcionales .....	8
3.2. Lista de Requerimientos no funcionales .....	8
3.3. Matriz de casos de usos y Requerimientos .....	9
3.4. Especificación de los casos de uso .....	12
3.5. Diagrama de Clases .....	18
3.6. Presentación del Prototipo .....	24
3.7. Códigos de los formularios y pantallas de ejecución .....	27
<b>BIBLIOGRAFÍA .....</b>	<b>35</b>

## 1. ASPECTOS GENERALES

### 1.1. Introducción de la empresa

Tambo es una cadena peruana de tiendas de conveniencia perteneciente a la familia Lindley. En el año 2020 alcanzó la cifra de 400 locales, lo que la convierte en la más grande de su rubro en el país. El gerente general de la empresa, Luis Seminario, inauguró su primer local en el distrito de Comas, Lima, en el año 2015. Tambo inició la tendencia de colocar tiendas de conveniencia fuera de gasolineras (las cuales eran los únicos lugares donde se encontraban este tipo de locales, como “*Listo!*” y “*Viva!*”). Hoy en día es una de las marcas más conocidas por el poblador limeño común, ya que abundan sus locales a lo largo de la ciudad y son fáciles de encontrar para comprar gran variedad de productos.

### 1.2. Datos generales de la empresa

#### 1.2.1. Misión

La misión de tiendas Tambo es: Mejorar la calidad de nuestros clientes y trabajadores.

#### 1.2.2. Visión

“Ser la practi-tienda con más establecimientos en el país”.

#### 1.2.3. Objetivos de la empresa

- Ofrecer productos a los precios más competitivos del mercado.
- Satisfacer al público con menor poder adquisitivo.
- Fidelizar al público consumidor.

### 1.3. Definición del problema

#### 1.3.1. Descripción del Problema

La empresa Tambo, una cadena de tiendas de conveniencia en Perú, necesita agilizar su proceso de ventas y mejorar la experiencia del cliente. Actualmente, Tambo realiza sus ventas de manera manual, lo que genera errores, retrasos y pérdida de tiempo tanto para los empleados como para los clientes.

#### 1.3.2. Problema General

Tambo, carece de un sistema de ventas para mejorar su proceso de ventas y ofrecer una experiencia de compra más eficiente. A partir de ello, surge la siguiente

pregunta: ¿Cómo el sistema de ventas mejora el proceso de gestión de ventas de Tambo en el año 2023?

### **1.3.3. Problemas Específicos**

- ¿Cómo un sistema de ventas mejora la poca optimización del control de inventario?
- ¿Cómo un sistema de ventas resuelve los problemas de control de stock y dificultades para realizar pedidos de reposición de productos de manera oportuna?
- ¿Cómo un sistema de ventas mejora el impacto en los resultados financieros ocasionado por la falta de un sistema de ventas eficiente?

## **1.4. Definición de objetivos**

### **1.4.1. Objetivo General**

Desarrollar un sistema de ventas para la empresa Tambo, que permita mejorar la gestión de sus ventas y optimizar el control de inventario. Con la implementación de este sistema, se espera reducir los errores y retrasos en el proceso de ventas, mejorar la satisfacción del cliente y obtener información valiosa para la toma de decisiones. Además, se espera que este sistema de ventas ayude a Tambo a ser más competitiva en el mercado y a aumentar su rentabilidad.

### **1.4.2. Objetivo Especifico**

- Determinar como un sistema de ventas mejora la poca optimización del control de inventario.
- Determinar como un sistema de ventas resuelve los problemas de control de stock y dificultades para realizar pedidos de reposición de productos de manera oportuna.
- Determinar como un sistema de ventas mejora el impacto en los resultados financieros por ocasionado por la falta de un sistema de ventas eficiente.

### **1.4.3. Alcances y Limitaciones**

El desarrollo del sistema se basa en realizar un sistema que va a ayudar y facilitar la agilización del proceso de ventas y así lograr una eficaz manera de atender a los clientes. En términos generales, el alcance del sistema va a incluir un registro de ventas que permitirá el registro de ventas de los productos que se comercializan en Tambo, incluyendo información como el producto, el precio, la cantidad, el cliente y el medio de pago utilizado. Es importante destacar que el alcance del sistema puede variar en función

de los recursos para el desarrollo del proyecto, así como de las necesidades específicas de la empresa Tambo.

## **2. MARCO TEÓRICO**

### **2.1. Antecedentes**

#### **2.1.1. Antecedentes nacionales**

Yaurima (2019) en su investigación buscó aplicar la integración de un sistema contable con el sistema de ventas de un restaurante en la ciudad de Lima, Perú, para ello se realizó un levantamiento del total de información que dispone la empresa con el fin de saber la cantidad de parámetros necesarios para para realizar las consultas, filtros y validaciones. Finalmente, los resultados demostraron que la integración de los sistemas mejoró considerablemente la gestión contable de la empresa.

Quispe (2021) en su investigación buscó integrar un sistema de ventas en la botica Cloti en la ciudad de Cañete, Perú, para ello se empleó el enfoque cuantitativo, con un diseño no experimental. La población estuvo compuesta por los trabajadores de la botica Cloti, mientras tanto la muestra fueron 10 trabajadores del área de soporte técnico, a quienes se les aplico como instrumento un cuestionario. Finalmente, los resultados demostraron que la implementación de un sistema de ventas permitió mejorar el proceso de registro de ventas y contribuyo con la reducción del tiempo de atención, mejorando de esta forma la experiencia de compra en la botica Cloti.

#### **2.1.2. Antecedentes internacionales**

Martín (2019) en su investigación buscó diseñar e implementar un sistema de inventarios para el almacén de pinturas y ferretería Ferrecolor en la ciudad de Villavicencio, Colombia, para ello se empleó las metodologías de desarrollo de software, con un diseño no experimental. Finalmente, los resultados han demostrado que esta solución le ha permitido al almacén y su administrador, consultar y controlar la base de datos de su inventario, y gracias a ello enfocar los recursos y lograr un mejor crecimiento empresarial.

Cieri (2022) en su investigación buscó implementar una aplicación móvil para la gestión del ciclo de vida de automóviles de un concesionario en la ciudad de Las Palmas, España, para ello se emplearon las metodologías de desarrollo de software móvil, con un diseño no experimental. La población estuvo conformada por los trabajadores de la empresa concesionaria, mientras que la muestra fueron 15 trabajadores del área de soporte técnico, a quienes se les aplicó como instrumento un cuestionario. Finalmente, los resultados han demostrado que la aplicación ha permitido automatizar el proceso de la gestión del ciclo de vida de los automóviles de la concesionaria.

## **2.2. Fundamento Teórico**

### **2.2.1. JAVA:**

Java es un lenguaje de programación de propósito general, orientado a objetos, desarrollado por Sun Microsystems en la década de 1990. El lenguaje de programación Java es conocido por su portabilidad, lo que significa que puede ser ejecutado en cualquier plataforma que tenga una máquina virtual Java (JVM) instalada. Además, Java es un lenguaje seguro que utiliza un sistema de gestión de memoria automatizado y tiene una amplia biblioteca de clases y métodos para realizar tareas comunes.

### **2.2.2. NetBeans**

NetBeans es un entorno de desarrollo integrado (IDE) que se utiliza para desarrollar aplicaciones en varios lenguajes de programación, incluyendo Java. NetBeans es una herramienta muy útil para desarrolladores de Java, ya que proporciona una amplia gama de características, como la edición de código, depuración, compilación, y muchas más. NetBeans también ofrece la posibilidad de trabajar con varios marcos de trabajo y tecnologías, como JUnit, Struts e Hibernate.

### **2.2.3. Base de Datos - SQL**

SQL (Structured Query Language) es un lenguaje utilizado para administrar y manipular datos en un sistema de gestión de bases de datos (DBMS). SQL se utiliza para realizar tareas como la creación, modificación y eliminación de tablas y registros en una base de datos. Algunas de las operaciones más comunes que se realizan con SQL son la selección, inserción, actualización y eliminación de datos.

#### **2.2.4. Lenguaje Unificado de Modelo – UML**

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se utiliza para representar y diseñar sistemas de software. UML proporciona una serie de herramientas para modelar sistemas de software, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia y muchos otros. Estas herramientas son útiles para los desarrolladores de software porque les permiten visualizar, diseñar y comunicar de manera efectiva las ideas y los conceptos que se necesitan para construir un sistema de software.

#### **2.2.5. Diagrama de Casos de Uso**

Un diagrama de casos de uso es una herramienta utilizada en el modelado de sistemas para identificar los requisitos funcionales y no funcionales de un sistema de software. Un diagrama de casos de uso muestra las relaciones entre los actores y los casos de uso de un sistema, lo que ayuda a los desarrolladores a comprender mejor los requisitos del sistema y las interacciones entre los usuarios y el sistema.

#### **2.2.6. Sistema de Ventas**

Un sistema de ventas es un conjunto de procesos y herramientas que se utilizan para gestionar el proceso de venta de un producto o servicio. Un sistema de ventas puede incluir desde la administración de inventarios hasta la gestión de relaciones con clientes, pasando por la gestión de facturación y cobranza. Un buen sistema de ventas puede ayudar a mejorar la eficiencia y la efectividad de una empresa al automatizar muchos de los procesos que se necesitan para vender productos o servicios.

### 3. DESARROLLO DE LA SOLUCIÓN

#### 3.1. Lista de Requerimientos funcionales

Para el desarrollo y correcto funcionamiento del Sistema de Ventas de Tambo, se necesitan las siguientes funcionalidades:

- **RF01:** El sistema permitirá el registro de usuarios, los cuales deberán proporcionar la una serie de datos de forma obligatoria. Entre ellos se destacan el correo electrónico y la contraseña. Además, debe validad que el correo electrónico del cliente no se encuentre previamente registrado en el sistema.
- **RF02:** El sistema debe permitir a los usuarios iniciar sesión empleando su correo electrónico y contraseña previamente registrados.
- **RF03:** El sistema debe permitir a los usuarios realizar compras de los productos seleccionados. Además, al seleccionar un producto debe permitir ingresar la cantidad del mismo.
- **RF04:** El sistema permitirá visualizar el resumen de los productos seleccionados por los usuarios. Asimismo, debe permitir quitar productos de la lista de compras si el usuario lo requiere.
- **RF05:** El sistema debe permitir a los usuarios realizar el pago de sus compras usando métodos de pago válidos, como lo pueden ser tarjetas bancarias. Para ello, el sistema deberá validar los métodos de pago seleccionados.
- **RF06:** Una vez finalizado y validado el proceso de pago, el sistema debe guardar la compra realizada por el usuario en el historial de compras, donde se visualizará: el código, fecha, hora e importe total de la compra.

#### 3.2. Lista de Requerimientos no funcionales

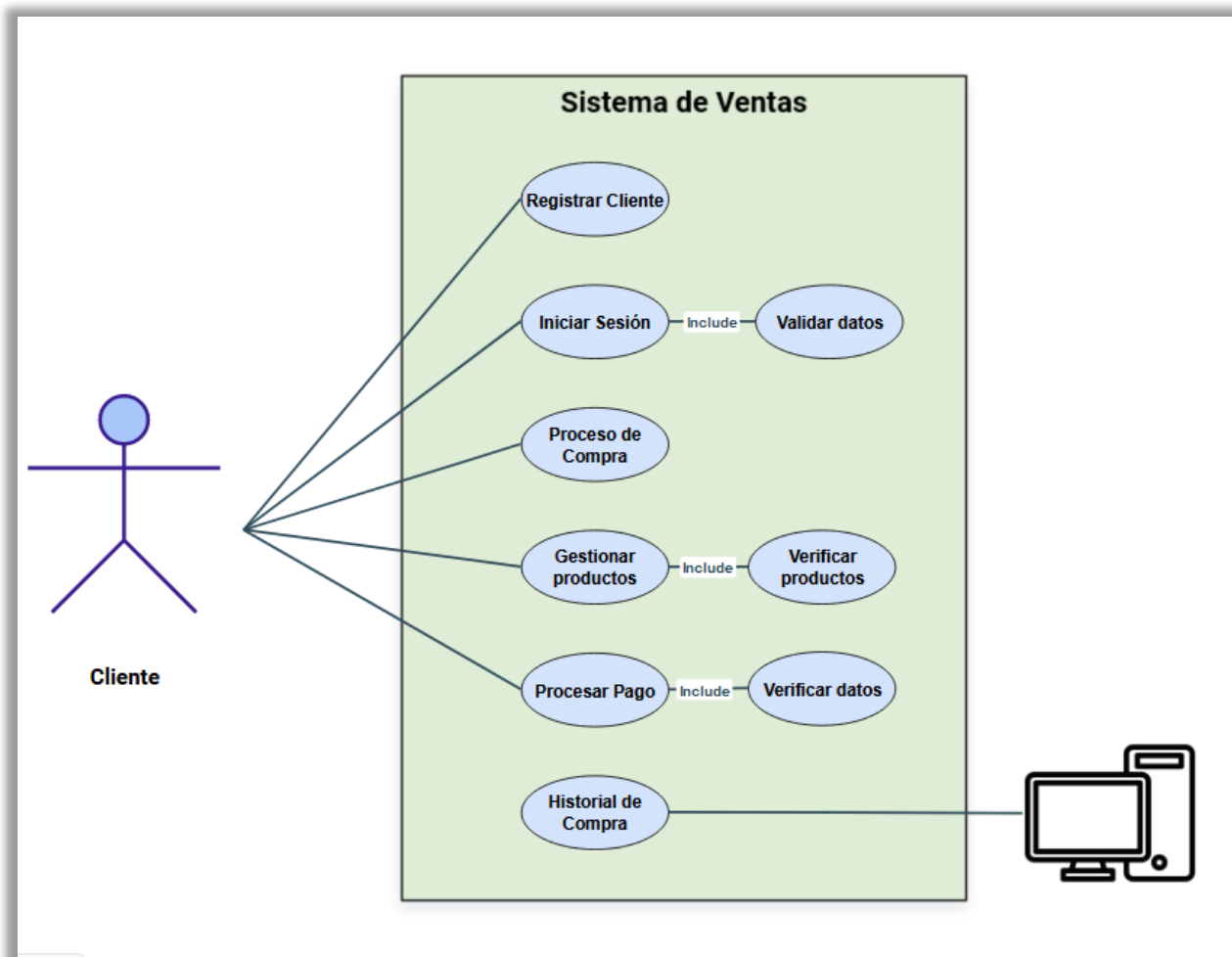
- **RNF01:** El sistema debe proporcionar un tiempo de respuesta rápido cuando el usuario realice acciones dentro del mismo. Esto garantizará una respuesta fluida y ofrecerá una experiencia satisfactoria al cliente.
- **RNF02:** El sistema debe ser escalable para que en un futuro pueda manejar un mayor volumen de usuarios, transacciones y carga de trabajo.
- **RNF03:** El sistema debe garantizar una alta disponibilidad, asegurando que este disponible las 24 horas del día.
- **RNF04:** El sistema debe ser compatible con una amplia gama de dispositivos y plataformas, garantizando que la interfaz del sistema se muestre adecuadamente.



### 3.3. Matriz de casos de usos y Requerimientos

#### 3.3.1. Diagrama de Casos de Uso

A continuación, se presenta el diagrama de Casos de Uso, donde se podrá apreciar las interacciones que podrá realizar el usuario con el sistema de Ventas de Tambo.



Fuente: Elaboración Propia

### 3.3.2. Matrices de Casos de Uso

#### Registro de Cliente

ID	E	Nombre CP	Datos de Entrada	Resultado Esperado
P1	E1	Registro Exitoso	Datos: DNI, nombre, etc. Email: *****@prueba.com Contraseña: *****	Cliente registrado correctamente. Los datos son almacenados para su posterior uso.
P2	E2	Registro Invalido	Datos: DNI, nombre, etc. Email: Contraseña: *****	Cliente no registrado en el sistema.

#### Inicio de Sesión

ID	O	Nombre CP	Datos de Entrada	Resultado Esperado
P1	O1	Inicio de Sesión Exitoso	Usuario: *****@prueba.com Contraseña: 123456	Se permite el ingreso y se le redirecciona al apartado de proceso de compra.
P2	O2	Inicio de Sesión Invalido	Usuario: *****@prueba.com Contraseña: 654321	No se permite el ingreso al sistema y se sugiere revisar los datos proporcionados.
P3	O3	Inicio de Sesión como invitado	Usuario: N/A Contraseña: N/A	Se permite el ingreso y se le redirecciona al apartado de proceso de compra.

#### Proceso de Compra

ID	O	Nombre CP	Datos de Entrada	Resultado Esperado
P1	O1	Selección de Productos	Producto 1: ***** Producto 2: ***** Producto 3: *****	El cliente selecciona los productos que desea.

### Gestión de Productos

ID	O	Nombre CP	Datos de Entrada	Resultado Esperado
P1	O1	Revisión de Productos	Producto 1: ***** Producto 2: ***** Producto 3: *****	El cliente revisa los productos que desea y comprueba el precio total.
P2	O2	Eliminación de Productos	Producto 1: ***** Producto 2: *****	El cliente elimina uno o varios productos de su lista de compra y se actualiza el precio total.

### Proceso de Pago

ID	O	Nombre CP	Datos de Entrada	Resultado Esperado
P1	O1	Pago Exitoso	Datos del cliente: ***** Tarjeta: ***** FV: **/** CCV: ***	Pago realizado exitosamente. Los datos proporcionados son correctos.
P2	O2	Pago Rechazado	Datos del cliente: ***** Tarjeta: ***** FV: **/** CCV: N/A	Pago rechazado. Algún dato ha sido ingresado incorrectamente.

### Historial de Compra

ID	O	Nombre CP	Datos de Entrada	Resultado Esperado
P1	O1	Historial actualizado	Código: ***** Fecha: **/**/** Hora: **: ** Importe total: S/. *****	Pago realizado exitosamente. Los datos proporcionados son correctos.
P2	O2	Historial no actualizado	Código: ***** Fecha: **/**/** Hora: **: ** Importe total: S/. *****	Pago rechazado. Algún dato ha sido ingresado incorrectamente.

### 3.4. Especificación de los casos de uso

#### 3.4.1. Caso de Uso: Registro del Cliente

<b>ID</b>	CU-01	
<b>Nombre</b>	Registro de Cliente	
<b>Descripción</b>	El cliente se dispone a registrarse en el sistema de compra, lo cual le proporcionará ciertos beneficios.	
<b>Actores</b>	Clientes de Tambo	
<b>Precondición</b>	El cliente deberá tener un correo electrónico valido y actualizado.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El cliente ingresa los siguientes atributos: DNI, nombre, apellidos y número de contacto, en sus respectivos campos.
	<b>2</b>	El cliente ingresa su correo electrónico dentro del campo correspondiente a la misma.
	<b>3</b>	El cliente crea e ingresa su contraseña.
	<b>4</b>	El sistema verifica que todos los campos estén completos y que no haya datos repetidos en la base de datos.
	<b>5</b>	Finaliza el caso de uso.
<b>Postcondición</b>	El cliente se ha registrado correctamente en el sistema y se dispone a iniciar sesión con sus credenciales.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	<b>4</b>	<b>4.1.</b> El sistema muestra un mensaje de error, debido a que algún campo está incompleto. <b>4.2.</b> El sistema muestra de error, debido a que algún dato es invalido. <b>4.3.</b> El sistema muestra un mensaje de error, debido a que algún dato ya está registrado en la base de datos.
<b>Anotaciones</b>		

### 3.4.2. Caso de Uso: Inicio de Sesión

<b>ID</b>	CU-02	
<b>Nombre</b>	Inicio de Sesión	
<b>Descripción</b>	El cliente se dispone a iniciar sesión con su correo electrónico y contraseña para ingresar al sistema.	
<b>Actores</b>	Clientes de Tambo	
<b>Precondición</b>	El cliente deberá estar registrado en el sistema de Tambo.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El cliente ingresa su correo electrónico dentro del campo que contiene el mismo nombre.
	<b>2</b>	El cliente ingresa su contraseña dentro del campo correspondiente a la misma.
	<b>3</b>	El sistema verifica que todos los campos estén completos y que no haya datos repetidos en la base de datos.
	<b>4</b>	Finaliza el caso de uso.
<b>Postcondición</b>	El cliente se ha iniciado sesión correctamente en el sistema y se dispone a iniciar con el Proceso de Compra.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	<b>3</b>	<b>3.1.</b> El sistema muestra un mensaje de error, debido a que algún campo está incompleto. <b>3.2.</b> El sistema muestra de error, debido a que algún dato es invalido.
<b>Anotaciones</b>	Si el cliente lo requiere, puede iniciar sesión como invitado.	

### 3.4.3. Caso de Uso: Inicio de Sesión

<b>ID</b>	CU-03	
<b>Nombre</b>	Proceso de compra	
<b>Descripción</b>	El cliente se dispone a seleccionar los productos de su gusto, los cuales estarán divididos por categorías.	
<b>Actores</b>	Clientes de Tambo	
<b>Precondición</b>	El cliente debe iniciar sesión en el sistema o en su defecto ingresar como invitado.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	1	El cliente selecciona la categoría del producto que desea.
	2	El cliente escoge los productos que desea y los agrega a su lista de compra.
	3	El sistema realiza las operaciones correspondientes para mostrar el precio parcial de los productos seleccionados.
	4	Finaliza el caso de uso.
<b>Postcondición</b>	El cliente se ha seleccionado los productos que necesita y se dispone a pasar al apartado de Gestión de Productos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	2	2.1. El sistema muestra un mensaje de error, debido a que el producto no tiene stock.
<b>Anotaciones</b>		

### 3.4.4. Caso de Uso: Inicio de Sesión

<b>ID</b>	CU-04	
<b>Nombre</b>	Gestión de Productos	
<b>Descripción</b>	El cliente se dispone a revisar los productos que han sido seleccionado, así mismo, durante este proceso el usuario podrá eliminar productos si lo requiere.	
<b>Actores</b>	Clientes de Tambo	
<b>Precondición</b>	El cliente debe haber seleccionado productos durante el Proceso de Compra.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El cliente revisa los productos que ha seleccionado.
	<b>2</b>	El cliente visualiza en pantalla el resumen de precio total por los productos seleccionados.
	<b>3</b>	El cliente, conforme con su lista de productos, presiona un botón que le redireccionará al apartado de Proceso de Pago.
	<b>4</b>	Finaliza el caso de uso.
<b>Postcondición</b>	El cliente ha verificado los productos que tiene en su lista de compra y se dispone a pasar al apartado de Proceso de Pago.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	<b>1.1.</b> El cliente elimina uno o varios productos de su lista de compra. <b>1.2.</b> El sistema actualiza el precio total de compra, el cual será mostrado en pantalla-
<b>Anotaciones</b>		

### 3.4.5. Caso de Uso: Proceso de Pago

<b>ID</b>	CU-05	
<b>Nombre</b>	Proceso de Pago	
<b>Descripción</b>	El cliente se dispone a pagar el importe de los productos que ha seleccionado previamente.	
<b>Actores</b>	Clientes de Tambo	
<b>Precondición</b>	El cliente debe haber revisado y aceptado su lista de compras durante el proceso Gestión de Productos.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El sistema completa automáticamente los datos del cliente que ha iniciado sesión en el sistema.
	<b>2</b>	El cliente selecciona el método de pago que utilizará para realizar el pago.
	<b>3</b>	El cliente realiza el pago del importe final, mediante una tarjeta bancaria o en su defecto en efectivo.
	<b>4</b>	Finaliza el caso de uso.
<b>Postcondición</b>	Una vez completado el proceso de pago, el sistema guarda la compra realizada por el cliente en el apartado de Historial de Compra.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	<b>3</b>	<b>3.1.</b> El cliente ingresa uno o varios datos bancarios erróneos. <b>3.2.</b> El sistema rechaza el pago, debido a los datos erróneos proporcionados.
<b>Anotaciones</b>		



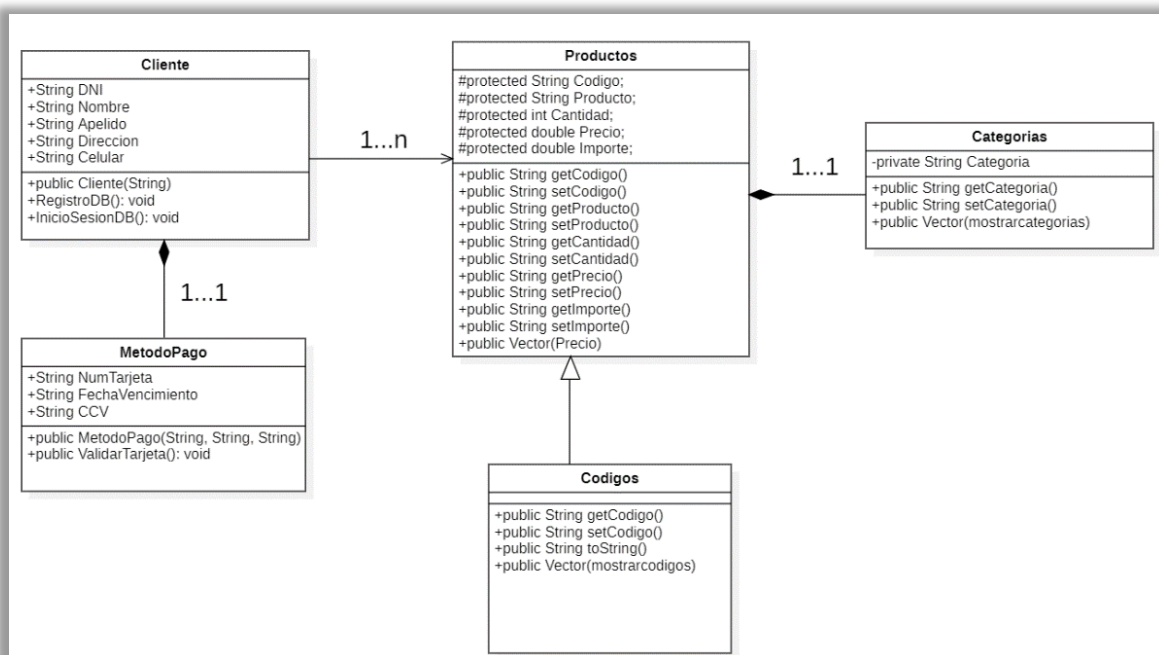
### 3.4.6. Caso de Uso: Historial de Compra

<b>ID</b>	CU-06	
<b>Nombre</b>	Historial de Compra	
<b>Descripción</b>	El sistema guarda la información de la compra realizada por el cliente en su historial de compra.	
<b>Actores</b>	Sistema de Compra de Tambo	
<b>Precondición</b>	El cliente debe haber realizado exitosamente el pago de los productos que ha seleccionado previamente.	
<b>Flujo Principal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El sistema verifica que el pago se haya realizado correctamente.
	<b>2</b>	El sistema reúne los datos de la compra realizada por el cliente.
	<b>3</b>	El sistema guarda los datos de la compra realizada por el cliente en el apartado Historial de Compras, el cual es único de cada cliente.
	<b>4</b>	Finaliza el caso de uso.
<b>Postcondición</b>	El cliente visualiza su historial de compras actualizado con los detalles de los productos adquiridos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	<b>1.1.</b> El sistema rechaza el método de pago. <b>1.2.</b> El sistema no actualiza el historial de compras del cliente.
<b>Anotaciones</b>		

### 3.5. Diagrama de Clases

A continuación, se presenta el diagrama de clases UML del sistema de compras. Este diagrama es una representación visual de las entidades principales y las relaciones entre ellas en nuestro sistema. Al diseñar el diagrama, tuvimos en cuenta la estructura y la lógica de nuestro sistema, así como las interacciones entre las diferentes entidades.

- Relación de asociación entre Clientes y Productos, un cliente puede tener acceso a diferentes productos.
- Relación de herencia entre Códigos y Productos, Códigos hereda atributos de la clase Productos.
- Relación de composición entre Productos y Categorías, un producto pertenece a una categoría específica.
- Relación de composición entre Clientes y Método Pago, un cliente tiene un método de pago específico.



Fuente: Elaboración Propia

### 3.5.1. Clase Cliente

```
public class Cliente {
    String DNI;
    String Nombre;
    String Apellido;
    String Direccion;
    String Celular;
    String Email;
    String Pass;

    public Cliente(String DNI, String Nombre, String Apellido, String Direccion, String Celular, String Email, String Pass) {
        this.DNI = DNI;
        this.Nombre = Nombre;
        this.Apellido = Apellido;
        this.Direccion = Direccion;
        this.Celular = Celular;
        this.Email = Email;
        this.Pass = Pass;
    }

    public Cliente(String Email, String Pass) {
        this.Email = Email;
        this.Pass = Pass;
    }
}
```

```
47 void RegistroDB(){
48
49     if(DNI.isEmpty() || Nombre.isEmpty() || Apellido.isEmpty() || Direccion.isEmpty() || Celular.isEmpty() || Email.isEmpty() || Pass.isEmpty())
50         JOptionPane.showMessageDialog(parentComponent: null, message: "Debes rellenar todos los campos");
51     else{
52         try{
53             Connection con=Conexion.getConnection();
54             PreparedStatement ps=con.prepareStatement(sql: "INSERT INTO CLIENTE (DNI, Nombre, Apellido, Direccion, Celular, Email ,Pass) VALUES(?,?";
55
56             ps.setString(parameterIndex: 1, x: DNI);
57             ps.setString(parameterIndex: 2, x: Nombre);
58             ps.setString(parameterIndex: 3, x: Apellido);
59             ps.setString(parameterIndex: 4, x: Direccion);
60             ps.setString(parameterIndex: 5, x: Celular);
61             ps.setString(parameterIndex: 6, x: Email);
62             ps.setString(parameterIndex: 7, x: Pass);
63             ps.executeUpdate();
64             JOptionPane.showMessageDialog(parentComponent: null , message: "Registro guardado, proceda a iniciar sesión");
65         }catch(SQLException ex){
66             JOptionPane.showMessageDialog(parentComponent: null , message: ex.toString());
67         }
68         Login LoginFrame = new Login();
69         LoginFrame.setVisible(b: true);
70         LoginFrame.pack();
71         LoginFrame.setLocationRelativeTo(null); //Centra el formulario
72     }
73 }
74 }
```

```
void InicioSeionDB(){
    String url="SELECT * "+FROM CLIENTE WHERE Email='"+Email+"'and Pasa='"+Pass+"'";
    try{
        Connection con=Conexion.getConnection();
        PreparedStatement ps=con.prepareStatement(sql: url); //Precompila la consulta de SQL
        ResultSet rs=ps.executeQuery();

        if(rs.next()){
            //Si existe el Email
            String em=rs.getString(columnLabel: "Email");
            String pa=rs.getString(columnLabel: "Pass");
            String nom=rs.getString(columnLabel: "Nombre");
            String ape=rs.getString(columnLabel: "Apellido");

            if(Pass.equals(em)){
                //Si la contraseña es igual, se redirecciona a siguiente proceso

                JOptionPane.showMessageDialog(parentComponent: null, message: "Bienvenido");

                Login Lg=new Login();
                Lg.setVisible(x: false);

                MenuPrincipal Menu=new MenuPrincipal();
                Menu.setVisible(b: true);
                Menu.setLocationRelativeTo(null);

                nom=nom.toUpperCase();
                ape=ape.toUpperCase();
                Menu.jlbDNI_Historial.setText(nom+" "+ape);
            }else{
                JOptionPane.showMessageDialog(parentComponent: null, message: "La contraseña es incorrecta");
            }
        }else{
            //El Email no existe
            JOptionPane.showMessageDialog(parentComponent: null, message: "El correo electrónico no está registrado");
        }
    }catch(SQLException ex){
        JOptionPane.showMessageDialog(parentComponent: null, message: toString());
    }
}
```

### 3.5.2. Clase Productos

```

21 public class Productos {
22     protected StringCodigo;
23     protected StringProducto;
24     protected intCantidad;
25     protected doublePrecio;
26     protected doubleImporte;
27
28     public String getCodigo() {
29         returnCodigo;
30     }
31
32     public void setCodigo(StringCodigo) {
33         this.Codigo =Codigo;
34     }
35
36     public String getProducto() {
37         returnProducto;
38     }
39
40     public void setProducto(StringProducto) {
41         this.Producto =Producto;
42     }
43
44     public int getCantidad() {
45         returnCantidad;
46     }
47
48     public void setCantidad(intCantidad) {
49         this.Cantidad =Cantidad;
50     }
51
52     public double getPrecio() {
53         returnPrecio;
54     }
55
56     public double getImporte() {
57         returnImporte;
58     }
59
60     public void setImporte(doubleImporte) {
61         this.Importe =Importe;
62     }
63
64     public Vector<Productos> Precio(doublePrecio){
65         PreparedStatement ps=null;
66         ResultSet rs=null;
67         Conexion cx=new Conexion();
68         Connection con=cx.getConexion();
69
70         Vector<Productos> datos= new Vector<Productos>();
71         Productos dat=null;
72
73         try{
74             String sql="SELECT Precio FROM Producto where Codigo='"+Codigo+"'";
75             ps=con.prepareStatement(sql);
76             rs=ps.executeQuery();
77
78             dat=new Productos();
79
80             while(rs.next()){
81                 dat=new Productos();
82                 dat.setCodigo(rs.getString(columnLabel: "Codigo"));
83                 datos.add(e: dat);
84             }
85
86             rs.close();
87         }catch(SQLException ex){
88             System.err.println(x: ex.toString());
89         }
90         return datos;
91     }
92 }

```

```

public double getImporte() {
    returnImporte;
}

public void setImporte(doubleImporte) {
    this.Importe =Importe;
}

public Vector<Productos> Precio(doublePrecio){
    PreparedStatement ps=null;
    ResultSet rs=null;
    Conexion cx=new Conexion();
    Connection con=cx.getConexion();

    Vector<Productos> datos= new Vector<Productos>();
    Productos dat=null;

    try{
        String sql="SELECT Precio FROM Producto where Codigo='"+Codigo+"'";
        ps=con.prepareStatement(sql);
        rs=ps.executeQuery();

        dat=new Productos();

        while(rs.next()){
            dat=new Productos();
            dat.setCodigo(rs.getString(columnLabel: "Codigo"));
            datos.add(e: dat);
        }

        rs.close();
    }catch(SQLException ex){
        System.err.println(x: ex.toString());
    }
    return datos;
}

```

### 3.5.3. Clase Categorías

```
public class Categorías {
    private String Categoria;

    public String getCategoria() {
        return Categoria;
    }

    public void setCategoria(String Categoria) {
        this.Categoria = Categoria;
    }

    public String toString(){
        return Categoria;
    }
}
```

```
public Vector<Categorías> mostrarcategorías(){
    PreparedStatement ps=null;
    ResultSet rs=null;
    Conexion cx=new Conexion();
    Connection con=cx.getConexion();

    Vector<Categorías> datos= new Vector<Categorías>();
    Categorías dat=null;

    try{
        String sql="SELECT Categoria FROM CATEGORIA";
        ps=con.prepareStatement(sql);
        rs=ps.executeQuery();

        dat=new Categorías();
        dat.setCategoria(Categoria:"Seleccione categoria");
        datos.add(e: dat);

        while(rs.next()){
            dat=new Categorías();
            dat.setCategoria(Categoria:rs.getString(columnLabel: "Categoria"));
            datos.add(e: dat);
        }

        rs.close();
    }catch(SQLException ex){
        System.err.println(x: ex.toString());
    }
    return datos;
}
```

### 3.5.4. Clase Códigos

```
public class Códigos extends Productos {
    //private String Codigo;

    @Override
    public String getCodigo() {
        return Codigo;
    }

    @Override
    public void setCodigo(String Codigo) {
        this.Codigo = Codigo;
    }

    @Override
    public String toString(){
        return Codigo;
    }

    public Vector<Códigos> mostrarCódigos(String Categoria){
        PreparedStatement ps=null;
        ResultSet rs=null;
        Conexion cx=new Conexion();
        Connection con=cx.getConexion();

        Vector<Códigos> datos= new Vector<Códigos>();
        Códigos dat=null;

        try{
            String sql="SELECT Codigo FROM Producto where Categoria='"+Categoria+"'";
            ps=con.prepareStatement(sql);
            rs=ps.executeQuery();

            dat=new Códigos();
            dat.setCodigo(Codigo:"Seleccione código");
            datos.add(e: dat);
        }
```

```
            dat=new Códigos();
            dat.setCodigo(Codigo:"Seleccione código");
            datos.add(e: dat);

            while(rs.next()){
                dat=new Códigos();
                dat.setCodigo(Codigo:rs.getString(columnLabel: "Codigo"));
                datos.add(e: dat);
            }
            rs.close();
        }catch(SQLException ex){
            System.err.println(x: ex.toString());
        }
        return datos;
    }
}
```

### 3.5.5. Clase MétodoPago

```
public class MetodoPago {

    String NumTarjeta;
    String FechaVencimiento;
    String CCV;

    public MetodoPago(String NumTarjeta, String FechaVencimiento, String CCV) {
        this.NumTarjeta = NumTarjeta;
        this.FechaVencimiento = FechaVencimiento;
        this.CCV = CCV;
    }
}
```

```
public void ValidarTarjeta(){
    String url="SELECT NumTarjeta, FechaVencimiento, CCV FROM TARJETA WHERE NumTarjeta='"+NumTarjeta+"' "+and FechaVencimiento='"+
        FechaVencimiento+"' and CCV='"+CCV+"'";

    try{
        Connection con=Conexion.getConexion();
        PreparedStatement ps=con.prepareStatement(sql:url); //Precompila la consulta de SQL
        ResultSet rs=ps.executeQuery();

        if(rs.next()){
            String nt=rs.getString(columnLabel: "NumTarjeta");
            String fv=rs.getString(columnLabel: "FechaVencimiento");
            String ccv=rs.getString(columnLabel: "CCV");

            if(NumTarjeta.equals(anObject: nt) && FechaVencimiento.equals(anObject: fv) && CCV.equals(anObject: ccv)){
                JOptionPane.showMessageDialog(parentComponent: null, message: "COMPRA PROCESADA!");

                //Agregamos los datos de la compra al historial de compras del cliente
                String url1="INSERT INTO ";
                PreparedStatement pt=con.prepareStatement(sql:url1); //Precompila la consulta de SQL
                ResultSet st=pt.executeQuery();
            }else{
                JOptionPane.showMessageDialog(parentComponent: null, message: "PAGO RECHAZADO, REVISE LOS DATOS INGRESADOS");
            }
        }
    }catch(SQLException ex){
        System.err.println(x: ex);
    }
}
```

### 3.6. Presentación del Prototipo

#### 3.6.1. Prototipo – Registro de Usuario

The screenshot shows a window titled "Registrar" with a light gray background. At the top left, there is a small icon of a document with a pencil. Below the title, the word "Datos" is displayed. The form contains six input fields stacked vertically, each with a label to its left: "Nombres", "Apellidos", "E-mail", "Teléfono", "Contraseña", and "Confirmar Contraseña". At the bottom right of the form, there are two buttons: "Cancelar" (light gray) and "Registrarse" (pink).

#### 3.6.2. Prototipo – Inicio de Sesión

The screenshot shows a window titled "Inicio Sesión" with a light gray background. At the top center, the title "Inicio Sesión" is displayed. Below the title, there are two input fields stacked vertically, each with a label to its left: "E-mail" and "Contraseña". At the bottom center of the form, there is a single button labeled "Iniciar Sesión" (light gray).



### 3.6.3. Prototipo – Proceso de Compra

**PRODUCTOS**

CÓDIGO	DESCRIPCIÓN	CANTIDAD	PRECIO
--------	-------------	----------	--------

Producto:

Cantidad:

AGREGAR ELIMINAR

GUARDAR ACTUALIZAR

TOTAL:

### 3.6.4. Prototipo – Gestión de Productos

**GESTION DE PRODUCTOS**

CÓDIGO	CANTIDAD	PRECIO	DESCRIPCIÓN	PROVEEDOR
--------	----------	--------	-------------	-----------

Código:

Cantidad:

Precio:

Descripción:

Proveedor:

AGREGAR ELIMINAR

GUARDAR ACTUALIZAR

### 3.6.5. Prototipo – Proceso de Pago

**Cliente**

Nombres  Calle

Apellidos  Ciudad  Postal Code

DNI  País

**Forma de Pago**

☐ Tarjer de Crédito/ Debito ☐ Efectivo

Número de Tarjeta

Fecha de Expiración

Código de seguridad

**Datos de Compra**

**Id Compra** 00000000

**Fecha** 00/00/0000

**Monto** 00000000

Cancelar Siguiete

### 3.6.6. Prototipo – Historial de Compras

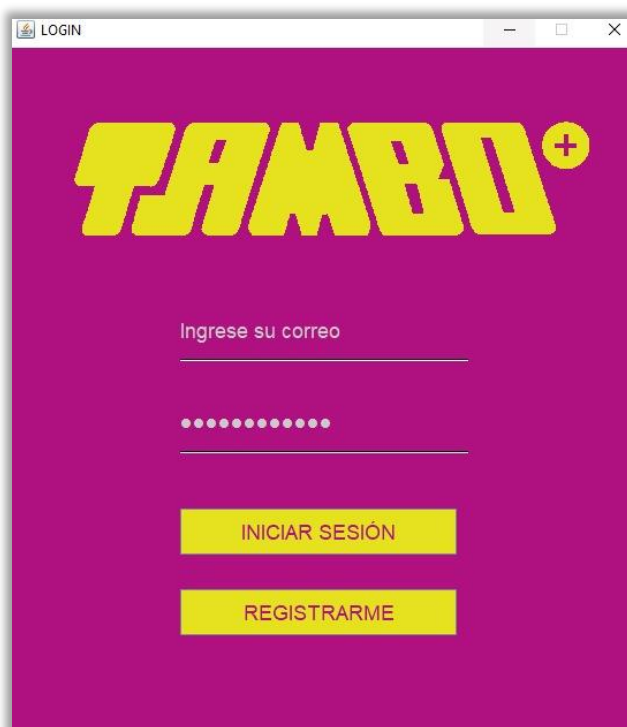
**Historial de Compras**

N° de Recibo	Fecha	Empleado	Cliente	Total
00000000	08/11/2002	Jose Aguilar De la Cruz Guerra	Jose Aguilar De la Cruz Guerra	2863.156
00000000	08/11/2002	Jose Aguilar De la Cruz Guerra	Jose Aguilar De la Cruz Guerra	2863.156
00000000	08/11/2002	Jose Aguilar De la Cruz Guerra	Jose Aguilar De la Cruz Guerra	2863.156

Cerrar

### 3.7. Códigos de los formularios y pantallas de ejecución

#### 3.7.1. Inicio de Sesión



##### 3.7.1.1. Botón Iniciar Sesión

```
private void jbtnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    // BOTÓN INICIAR SESIÓN

    String Email=this.jtxtEmail.getText();
    String Pass=this.jtxtPass.getText();

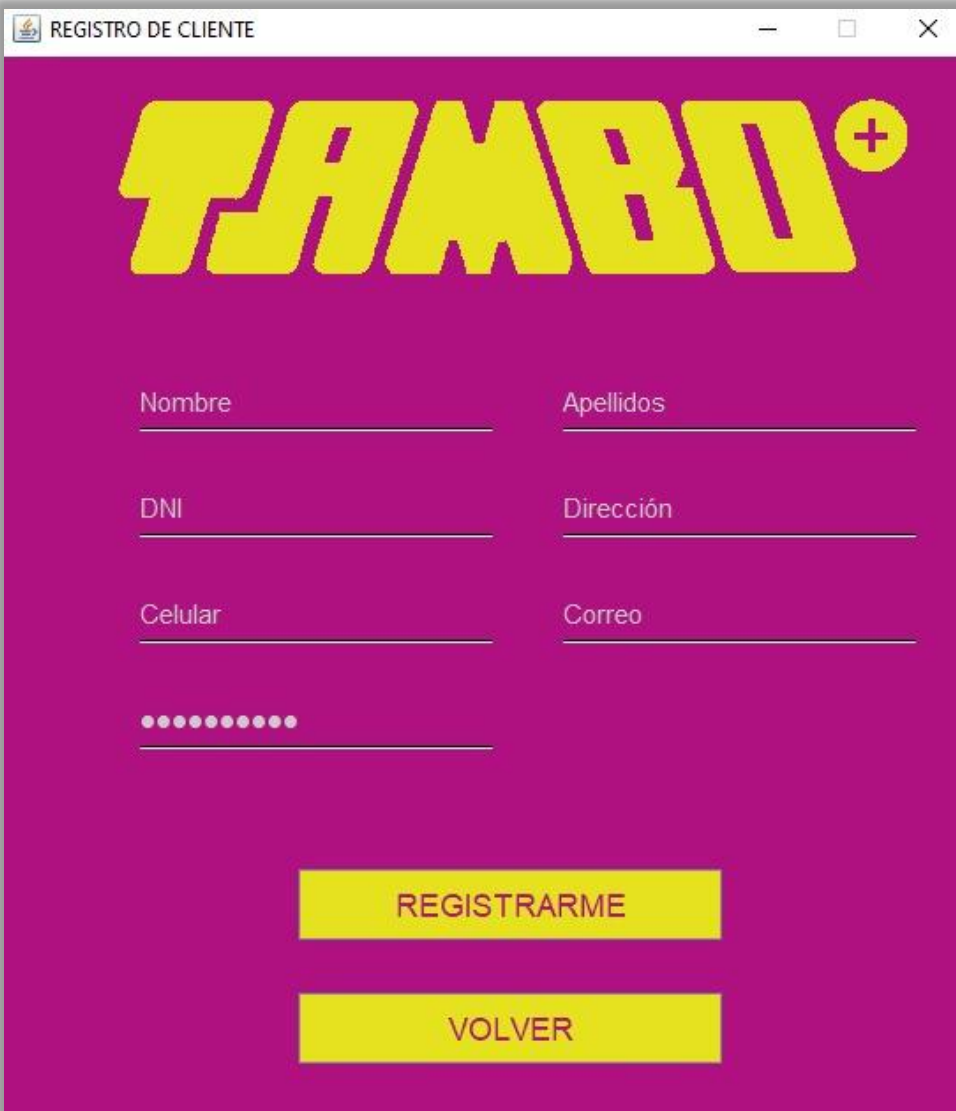
    MenuPrincipal Mp=new MenuPrincipal();
    Mp.setEmail(Email: jtxtEmail.getText());

    Cliente cl=new Cliente(Email, Pass);
    cl.InicioSeionDB();
    this.dispose();
}
```

### 3.7.1.2. Botón Registrarme

```
private void jButtonRegistroActionPerformed(java.awt.event.ActionEvent evt) {  
    // BOTÓN REGISTRARME  
  
    Registro RegistroFrame = new Registro();  
    RegistroFrame.setVisible(b: true);  
    RegistroFrame.pack();  
    RegistroFrame.setLocationRelativeTo(c: null); //Centra el formulario  
    this.dispose();  
}
```

### 3.7.2. Registro de Usuario



REGISTRO DE CLIENTE

**TAMBO** +

Nombre

Apellidos

DNI

Dirección

Celular

Correo

REGISTRARME

VOLVER

### 3.7.2.1. Botón Registrarme

```
private void jbtnRegistrarmeActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String DNI=this.jtxtDNI.getText();
    String Nombre=this.jtxtNombre.getText();
    String Apellido=this.jtxtApellido.getText();
    String Direccion=this.jtxtDireccion.getText();
    String Celular=this.jtxtCelular.getText();
    String Email=this.jtxtEmail.getText();
    String Pass=this.jtxtPass.getText();

    Cliente cl=new Cliente(DNI, Nombre, Apellido, Direccion, Celular, Email, Pass);

    cl.RegistroDB();

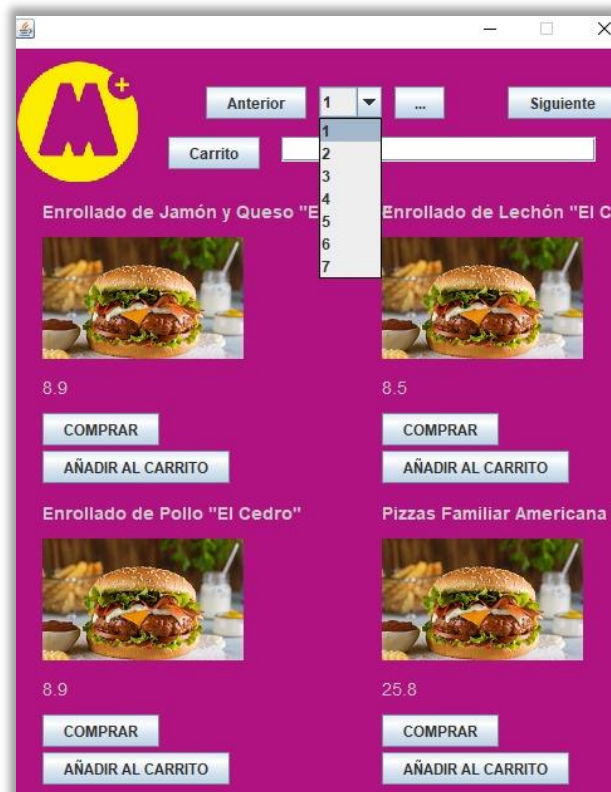
    this.dispose();
}
```

### 3.7.2.2. Botón Volver

```
private void jbtnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    Login LoginFrame = new Login();
    LoginFrame.setVisible(b: true);
    LoginFrame.pack();
    LoginFrame.setLocationRelativeTo(c: null); //Centra el formulario
    this.dispose();
}
```

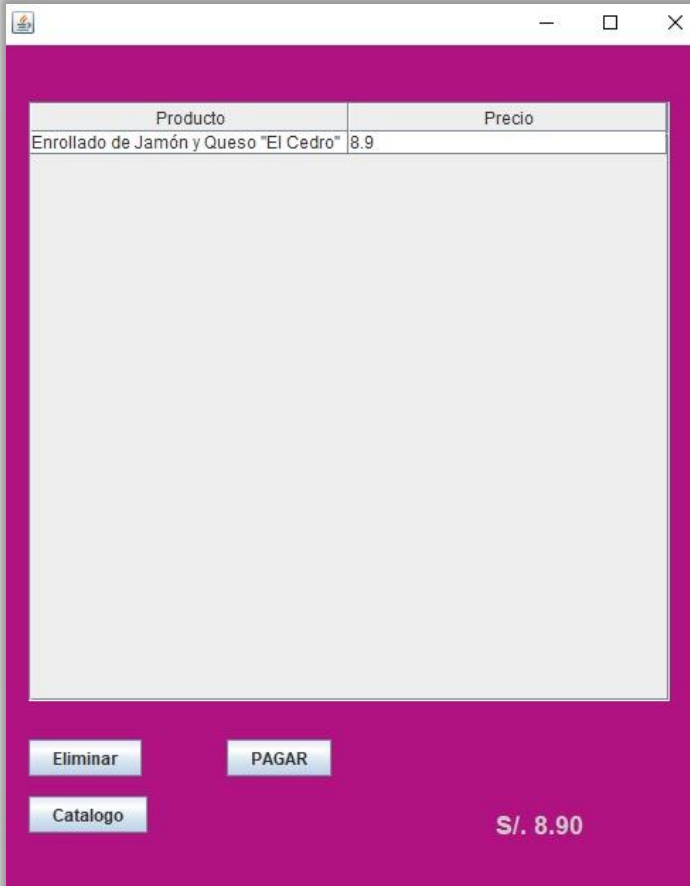
### 3.7.3. Menú Principal – Comprar



### 3.7.3.1. Botón Carrito

```
private void jButtonAgregarCarritoActionPerformed(java.awt.event.ActionEvent evt) {  
    Productos Carrito=new Productos();  
  
    Carrito.setCodigo(Codigo:jcbxCodigos.getSelectedItem().toString());  
    Carrito.setProducto(Producto: jlbProducto.getText());  
    Carrito.setPrecio(Precio:precio);  
    Carrito.setCantidad(Cantidad: cantidad);  
  
    importe=Math.round(precio*cantidad);  
  
    Carrito.setImporte(Importe: importe);  
  
    jspCantidad.setValue(value: 1);  
  
    ListaVentas.add(e: Carrito);  
    ActualizarTabla();  
    BorrarVenta();  
}
```

### 3.7.4. Menú Principal – Carrito de compras



Producto	Precio
Enrollado de Jamón y Queso "El Cedro"	8.9

Eliminar      PAGAR

Catalogo

S/. 8.90

#### 3.7.4.1. Botón pagar productos

```
private void jButtonPagarActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    jTabbedPane1.setSelectedIndex(index: 2);  
}
```

#### 3.7.5. Menú Principal – Selección de método de Pago

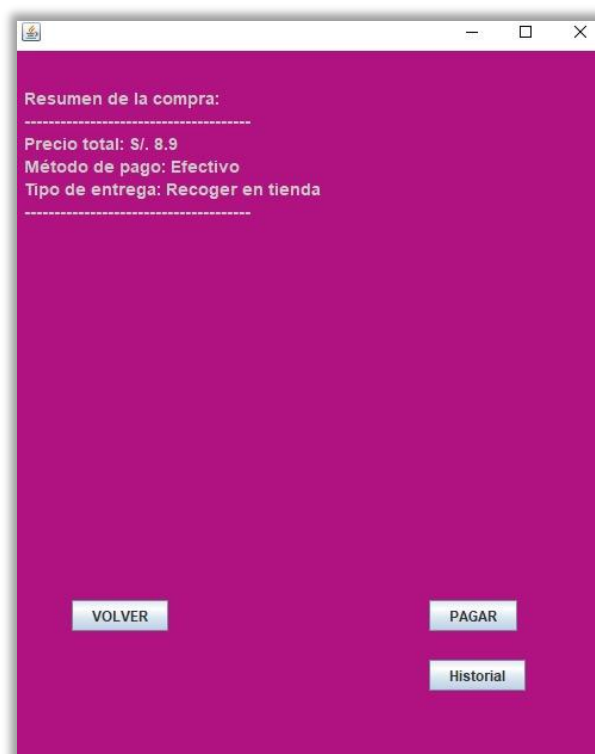


### 3.7.6. Menú Principal – Método de Entrega

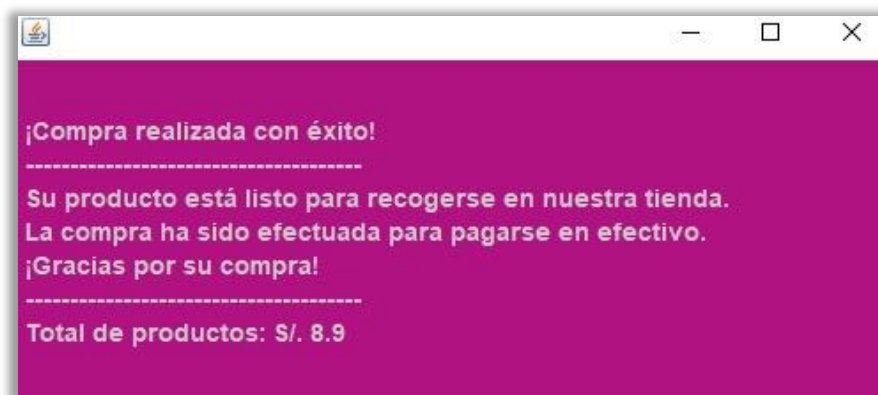




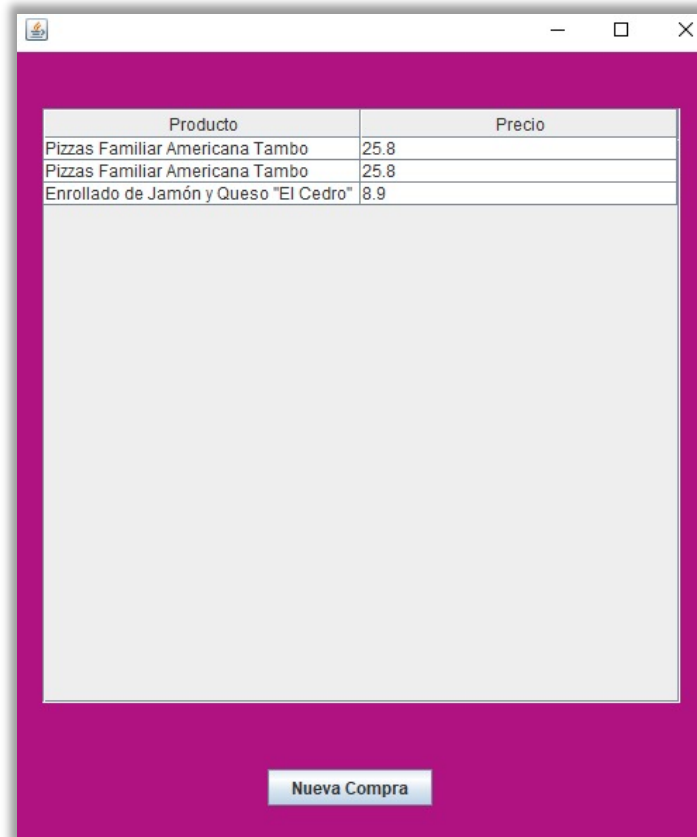
### 3.7.7. Menú Principal – Apartado de Historial de Compras



### 3.7.8. Menú Principal – Apartado de Compra Finalizada



### 3.7.9. Menú Principal – Apartado del Historial de Compras



---

## BIBLIOGRAFÍA

- (2015). *Tambo+: La tienda de conveniencia que mezcla el canal tradicional con lo moderno*. PerúRetail. <https://www.peru-retail.com/tambo-la-tienda-de-conveniencia-que-mezcla-el-canal-tradicional-con-lo-moderno/>
- Vásquez, A. (2020). *Tambo supera las 400 tiendas y afirma llegará a 600 para el bicentenario*. Mercado Negro. <https://www.mercadonegro.pe/retail/tambo-supera-las-400-tiendas-y-afirma-llegara-a-600-para-el-bicentenario/>
- Yaurima, K. (2019). *Integración del sistema contable con el sistema de ventas de un restaurante* [Tesis de grado, Universidad Tecnológica del Perú]. Repositorio Institucional de la Universidad Tecnológica del Perú. <https://repositorio.utp.edu.pe/handle/20.500.12867/1967>
- Quispe, F. (2021). *Implementación del sistema de ventas en botica Cloti – Cañete; 2021* [Tesis de grado, Universidad Católica de los Ángeles de Chimbote]. Repositorio institucional de la Universidad Católica de los Ángeles de Chimbote. <https://repositorio.uladech.edu.pe/handle/20.500.13032/31801>
- Martín, E. (2019). *Diseño e implementación de sistemas de inventarios para el almacén de pinturas y ferretería Ferrecolor* [Tesis de grado, Universidad Cooperativa de Colombia]. Repositorio Universidad Cooperativa de Colombia. <https://repository.ucc.edu.co/items/76116c33-46b9-4373-8408-60ed648fc6a0>
- Cieri, L. (2022). *Aplicación móvil para la gestión del Ciclo de Venta de automóviles de un concesionario (km cero y Segunda Mano)* [Tesis de grado, Universidad de Las Palmas de Gran Canaria]. Repositorio Institucional de la ULPGC. <https://accedacris.ulpgc.es/handle/10553/118377>