**UNIVERSIDAD MAYOR DE SAN ANDRÉS**

FACULTAD DE CIENCIAS PURAS Y NATURES

CARRERA DE INFORMATICA

**PROYECTO: SISTEMA DE GESTION DE INVENTARIO DE PRODUCTOS**



**Universitarios(a):** Enrique Iturri Rodríguez

Felix Rodríguez Cruz

Rene Hilari Bautista

Cesar Flores Laura

Jorge Ichuta Tuco

Gary Guarani Tarifa

Bladimir Maldonado

Asignatura: Inf-121

**Docente:** Rosalia López M.

**Fecha**:01/28/202

La Paz - Bolivia

**SISTEMA DE GESTION DE INVENTARIO DE PRODUCTOS**) para una mejor interacción con el usuario.

**1. Definición del Proyecto**

**1.1 Descripción General**

El proyecto consiste en un sistema de gestión de inventario que permite:

Administrar productos perecederos (como alimentos) y no perecederos (como electrónicos e indumentaria).

Registrar lotes para productos perecederos, incluyendo información como fecha de vencimiento y cantidad.

Gestionar operaciones básicas como agregar, eliminar, buscar y modificar productos.

Mostrar el inventario de manera organizada, con opciones para ordenar los productos por precio o cantidad de lotes.

**1.2 Objetivos**

Aplicar los conceptos fundamentales de POO, como herencia, polimorfismo y encapsulamiento.

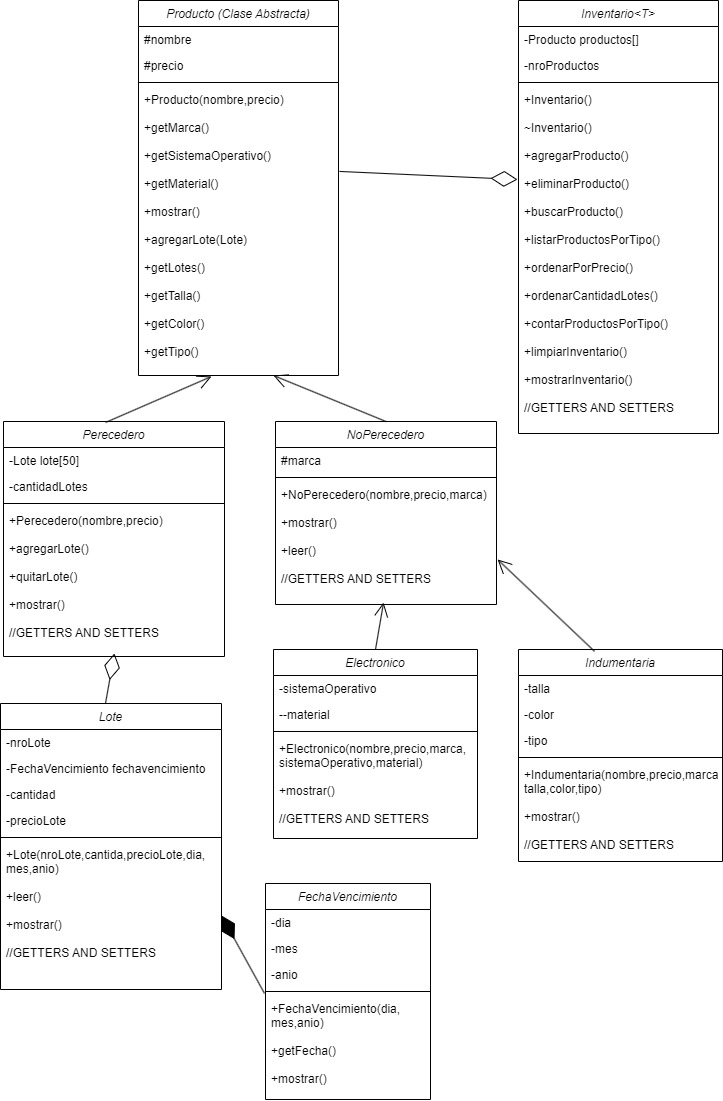
Diseñar un sistema modular y escalable que permita la gestión de diferentes tipos de productos.

Implementar una interfaz gráfica de usuario (GUI) para facilitar la interacción con el sistema.

Utilizar estructuras de datos y algoritmos para ordenar y buscar productos de manera eficiente.

**2. Análisis y Diseño**

**2.1 Diagrama UML**



**2.2 Principios de Diseño**

Herencia: Las clases Electrónico e Indumentaria heredan de NoPerecedero, que a su vez hereda de Producto.

Polimorfismo: Se utiliza para implementar métodos como mostrar() y agregarLote() en las clases derivadas.

Encapsulamiento: Los atributos de las clases están encapsulados y se accede a ellos mediante métodos getter y setter.

Manejo de colecciones: Se utilizan arreglos para gestionar la lista de productos y lotes.

Interfaz gráfica: Se implementa una GUI utilizando Java Swing para facilitar la interacción con el usuario.

**3. Implementación en Java**

**3.1 Estructura del Proyecto**

El proyecto está organizado en los siguientes paquetes y clases:

Interfaz: Contiene las clases principales del sistema, como Producto, Perecedero, Electrónico, Indumentaria, Lote e Inventario.

InterfazGRafica: Clase que implementa la interfaz gráfica de usuario utilizando Java Swing.

**3.2 Código Fuente**

##### **Clase Inventario**

class Inventario<T extends Producto> {

private T[] productos;

private int nroProductos;

@SuppressWarnings("unchecked")

public Inventario() {

productos = (T[]) new Producto[100];

nroProductos = 0;

}

public void agregarProducto(T producto) {

if (nroProductos < productos.length) {

productos[nroProductos] = producto;

nroProductos++;

} else {

System.out.println("Inventario lleno, no se puede agregar mas productos.");

}

}

public boolean eliminarProducto(String nombre) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equals(nombre)) {

for (int j = i; j < nroProductos - 1; j++) {

productos[j] = productos[j + 1];

}

productos[nroProductos - 1] = null;

nroProductos--;

System.out.println("Producto eliminado: " + nombre);

return true;

}

}

System.out.println("Producto no encontrado: " + nombre);

return false;

}

public T buscarProducto(String nombre) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equalsIgnoreCase(nombre)) {

return productos[i];

}

}

System.out.println("Producto no encontrado: " + nombre);

return null;

}

public void listarProductosPorTipo(Class<?> tipo) {

System.out.println("Productos del tipo: " + tipo.getSimpleName());

for (int i = 0; i < nroProductos; i++) {

//Pregunta si el tipo de clase es Perecedero o No perecedero en la posicion i

if (tipo.isInstance(productos[i])) {

System.out.println("Nombre: "+ productos[i].getNombre());

productos[i].mostrar();

System.out.println("-------------------------");

}

}

}

public void ordenarPorPrecio() {

for (int i = 0; i < nroProductos - 1; i++) {

for (int j = 0; j < nroProductos - i - 1; j++) {

Producto p1 = productos[j];

Producto p2 = productos[j + 1];

if (p1.getPrecio() > p2.getPrecio()) {

T aux = productos[j];

productos[j] = productos[j + 1];

productos[j + 1] = aux;

}

}

}

System.out.println("Productos ordenados por precio.");

}

public void ordenarCantidadLotes() {

for (int i = 0; i < nroProductos - 1; i++) {

for (int j = 0; j < nroProductos - i - 1; j++) {

if (productos[j] instanceof Perecedero && productos[j + 1] instanceof Perecedero) {

Perecedero p1 = (Perecedero) productos[j];

Perecedero p2 = (Perecedero) productos[j + 1];

if (p1.getCantidadLotes() > p2.getCantidadLotes()) {

T aux = productos[j];

productos[j] = productos[j + 1];

productos[j + 1] = aux;

}

}

}

}

System.out.println("Productos ordenados por cantidad de lotes.");

}

public int contarProductosPorTipo(Class<?> tipo) {

int contador = 0;

for (int i = 0; i < nroProductos; i++) {

if (tipo.isInstance(productos[i])) {

contador++;

}

}

System.out.println("Numero de productos de tipo " + tipo.getSimpleName() + ": " + contador);

return contador;

}

public void limpiarInventario() {

for (int i = 0; i < nroProductos; i++) {

productos[i] = null;

}

nroProductos = 0;

System.out.println("Inventario limpiado.");

}

public boolean actualizarPrecio(String nombre, double nuevoPrecio) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equalsIgnoreCase(nombre)) {

Producto producto = productos[i];

producto.setPrecio(nuevoPrecio);

//producto.precio = nuevoPrecio;

System.out.println("Precio actualizado para " + nombre + ": " + nuevoPrecio);

return true;

}

}

System.out.println("Producto no encontrado o no es NoPerecedero: " + nombre);

return false;

}

public void mostrarInventario() {

if(nroProductos==0){

System.out.println("El inventario esta vacio");

}

for (int i = 0; i < nroProductos; i++) {

System.out.println("\nNombre: "+ productos[i].getNombre());

productos[i].mostrar();

System.out.println("----------------------------------");

}

}

public T[] getProductos() {

return productos;

}

public int getNroProductos() {

return nroProductos;

}

}

class Inventario<T extends Producto> {

private T[] productos;

private int nroProductos;

@SuppressWarnings("unchecked")

public Inventario() {

productos = (T[]) new Producto[100];

nroProductos = 0;

}

public void agregarProducto(T producto) {

if (nroProductos < productos.length) {

productos[nroProductos] = producto;

nroProductos++;

} else {

System.out.println("Inventario lleno, no se puede agregar mas productos.");

}

}

public boolean eliminarProducto(String nombre) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equals(nombre)) {

for (int j = i; j < nroProductos - 1; j++) {

productos[j] = productos[j + 1];

}

productos[nroProductos - 1] = null;

nroProductos--;

System.out.println("Producto eliminado: " + nombre);

return true;

}

}

System.out.println("Producto no encontrado: " + nombre);

return false;

}

public T buscarProducto(String nombre) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equalsIgnoreCase(nombre)) {

return productos[i];

}

}

System.out.println("Producto no encontrado: " + nombre);

return null;

}

public void listarProductosPorTipo(Class<?> tipo) {

System.out.println("Productos del tipo: " + tipo.getSimpleName());

for (int i = 0; i < nroProductos; i++) {

//Pregunta si el tipo de clase es Perecedero o No perecedero en la posicion i

if (tipo.isInstance(productos[i])) {

System.out.println("Nombre: "+ productos[i].getNombre());

productos[i].mostrar();

System.out.println("-------------------------");

}

}

}

public void ordenarPorPrecio() {

for (int i = 0; i < nroProductos - 1; i++) {

for (int j = 0; j < nroProductos - i - 1; j++) {

Producto p1 = productos[j];

Producto p2 = productos[j + 1];

if (p1.getPrecio() > p2.getPrecio()) {

T aux = productos[j];

productos[j] = productos[j + 1];

productos[j + 1] = aux;

}

}

}

System.out.println("Productos ordenados por precio.");

}

public void ordenarCantidadLotes() {

for (int i = 0; i < nroProductos - 1; i++) {

for (int j = 0; j < nroProductos - i - 1; j++) {

if (productos[j] instanceof Perecedero && productos[j + 1] instanceof Perecedero) {

Perecedero p1 = (Perecedero) productos[j];

Perecedero p2 = (Perecedero) productos[j + 1];

if (p1.getCantidadLotes() > p2.getCantidadLotes()) {

T aux = productos[j];

productos[j] = productos[j + 1];

productos[j + 1] = aux;

}

}

}

}

System.out.println("Productos ordenados por cantidad de lotes.");

}

public int contarProductosPorTipo(Class<?> tipo) {

int contador = 0;

for (int i = 0; i < nroProductos; i++) {

if (tipo.isInstance(productos[i])) {

contador++;

}

}

System.out.println("Numero de productos de tipo " + tipo.getSimpleName() + ": " + contador);

return contador;

}

public void limpiarInventario() {

for (int i = 0; i < nroProductos; i++) {

productos[i] = null;

}

nroProductos = 0;

System.out.println("Inventario limpiado.");

}

public boolean actualizarPrecio(String nombre, double nuevoPrecio) {

for (int i = 0; i < nroProductos; i++) {

if (productos[i].getNombre().equalsIgnoreCase(nombre)) {

Producto producto = productos[i];

producto.setPrecio(nuevoPrecio);

//producto.precio = nuevoPrecio;

System.out.println("Precio actualizado para " + nombre + ": " + nuevoPrecio);

return true;

}

}

System.out.println("Producto no encontrado o no es NoPerecedero: " + nombre);

return false;

}

public void mostrarInventario() {

if(nroProductos==0){

System.out.println("El inventario esta vacio");

}

f public T[] getProductos() {

return productos;

}

public int getNroProductos() {

return nroProductos;

}

**}**

Clase Producto

abstract class Producto {

protected String nombre;

protected double precio;

public Producto(String nombre,double precio) {

this.nombre = nombre;

this.precio = precio;

}

public double getPrecio() {

return precio;

}

public void setPrecio(double precio) {

this.precio = precio;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public abstract String getMarca();

public abstract String getSistemaOperativo();

public abstract String getMaterial();

public abstract void mostrar();

public abstract void agregarLote(Lote lote);

public abstract Lote[] getLotes();

public abstract String getTalla();

public abstract String getColor();

public abstract String getTipo();

}

Clase perecedero

class Perecedero extends Producto {

private Lote[] lotes = new Lote[50];

private int cantidadLotes;

public Perecedero(String nombre,double precio) {

super(nombre,precio);

this.cantidadLotes=0;

}

@Override

public void agregarLote(Lote lote){

this.lotes[this.cantidadLotes]=lote;

this.cantidadLotes++;

}

public Lote quitarLote(){

int aux = this.cantidadLotes;

this.cantidadLotes--;

return this.lotes[aux];

}

@Override

public Lote[] getLotes() {

return lotes;

}

@Override

public String getMarca() {

return null;

}

@Override

public void mostrar(){

System.out.println("Precio del producto: "+precio);

System.out.println("Numero de lotes: "+cantidadLotes);

for (int i = 0; i < this.cantidadLotes; i++) {

lotes[i].mostrar();

}

}

public int getCantidadLotes() {

return cantidadLotes;

}

@Override

public String getSistemaOperativo() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getMaterial() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getTalla() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getColor() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getTipo() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

}

Clase Lote

public class Lote {

private String nroLote;

private FechaVencimiento fechaVencimiento;

private int cantidad;

private double precioLote;

public Lote(String nroLote, int cantidad,double precioLote, int dia, int mes , int anio) {

this.nroLote = nroLote;

this.precioLote=precioLote;

this.fechaVencimiento = new FechaVencimiento(dia, mes, anio);

this.cantidad = cantidad;

}

public String getNroLote() {

return nroLote;

}

public void setNroLote(String nroLote) {

this.nroLote = nroLote;

}

public FechaVencimiento getFechaVencimiento() {

return fechaVencimiento;

}

public void setFechaVencimiento(FechaVencimiento fechaVencimiento) {

this.fechaVencimiento = fechaVencimiento;

}

public int getCantidad() {

return cantidad;

}

public void setCantidad(int cantidad) {

this.cantidad = cantidad;

}

public void leer() {

Scanner lee = new Scanner(System.in);

System.out.print("Ingrese el número de lote: ");

this.nroLote = lee.next();

System.out.print("Ingrese la cantidad: ");

this.cantidad = lee.nextInt();

}

public void mostrar() {

System.out.println("Codigo de Lote: " + this.nroLote);

System.out.println("Cantidad: " + this. cantidad);

System.out.println("Precio del lote: "+this.precioLote);

this.fechaVencimiento.mostrar();

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public double getPrecioLote() {

return precioLote;

}

}

Clase FechaVencimiento

class FechaVencimiento {

private int dia;

private int mes;

private int año;

public FechaVencimiento(int dia, int mes, int año) {

this.dia = dia;

this.mes = mes;

this.año = año;

}

public String getFecha() {

return dia + "/" + mes + "/" + año;

}

public void mostrar(){

System.out.println("Fecha: "+dia + "/" + mes + "/" + año);

}

}

Clase NoPerecedero

class NoPerecedero extends Producto {

public String marca;

public NoPerecedero(String nombre, double precio,String marca) {

super(nombre,precio);

this.marca = marca;

}

@Override

public String getMarca() {

return marca;

}

public void setMarca(String marca) {

this.marca = marca;

}

@Override

public void mostrar(){

System.out.println("Marca: "+marca);

System.out.println("Precio:"+precio);

}

@Override

public void agregarLote(Lote lote) {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getSistemaOperativo() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getMaterial() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public Lote[] getLotes() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getTalla() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getColor() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

@Override

public String getTipo() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

}

Clase indumentaria

class Indumentaria extends NoPerecedero {

private String talla;

private String color;

private String tipo;

public Indumentaria(String nombre, double precio, String marca, String talla, String color, String tipo) {

super(nombre, precio, marca);

this.talla = talla;

this.color = color;

this.tipo = tipo;

}

@Override

public String getMarca() {

return marca;

}

@Override

public String getTalla() {

return talla;

}

@Override

public String getColor() {

return color;

}

@Override

public String getTipo() {

return tipo;

}

@Override

public void mostrar(){

super.mostrar();

System.out.println("Talla: "+talla);

System.out.println("Color: "+color);

System.out.println("Tipo: "+tipo);

}

}

Clase Electronico

class Electronico extends NoPerecedero {

private String sistemaOperativo;

private String material;

public Electronico(String nombre, double precio,String marca, String sistemaOperativo, String material) {

super(nombre, precio, marca);

this.sistemaOperativo = sistemaOperativo;

this.material = material;

}

@Override

public String getSistemaOperativo() {

return sistemaOperativo;

}

@Override

public String getMaterial() {

return material;

}

@Override

public void mostrar(){

super.mostrar();

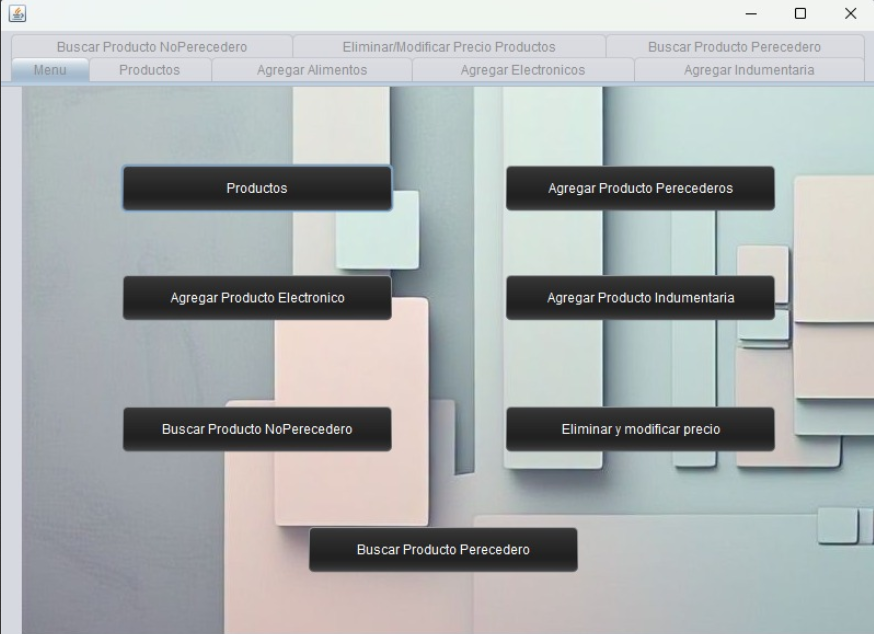
System.out.println("Sistema Operativo: "+sistemaOperativo);

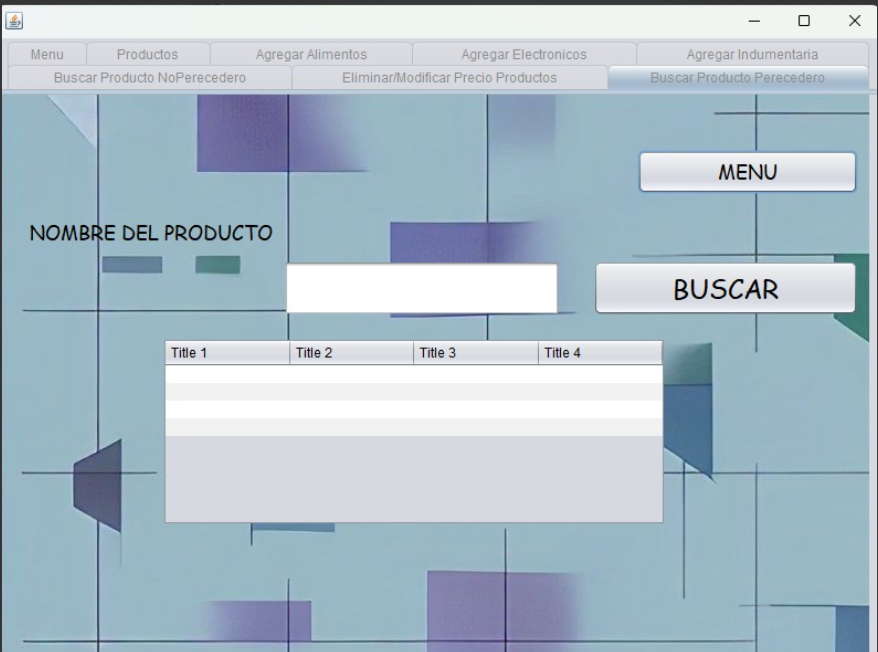
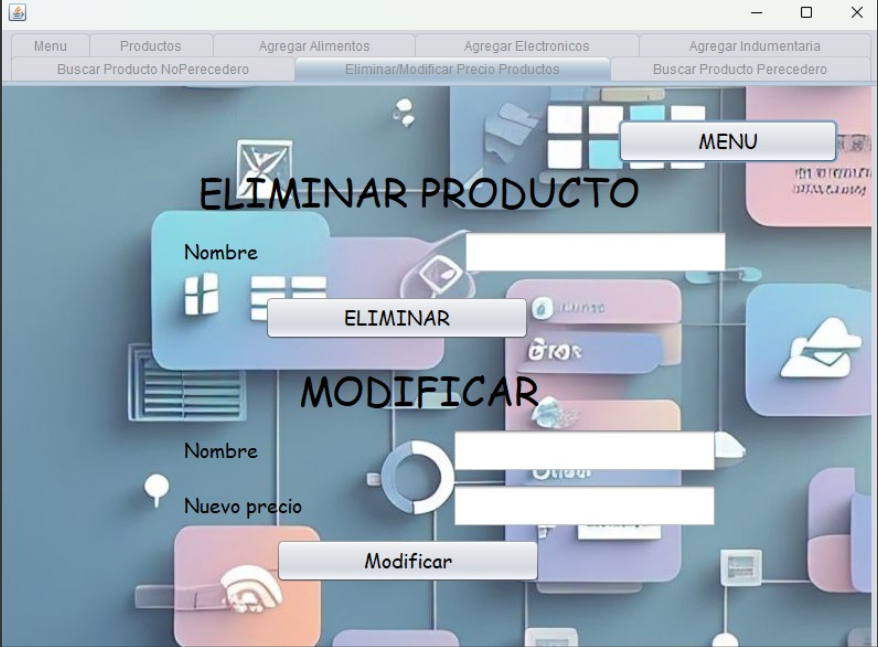
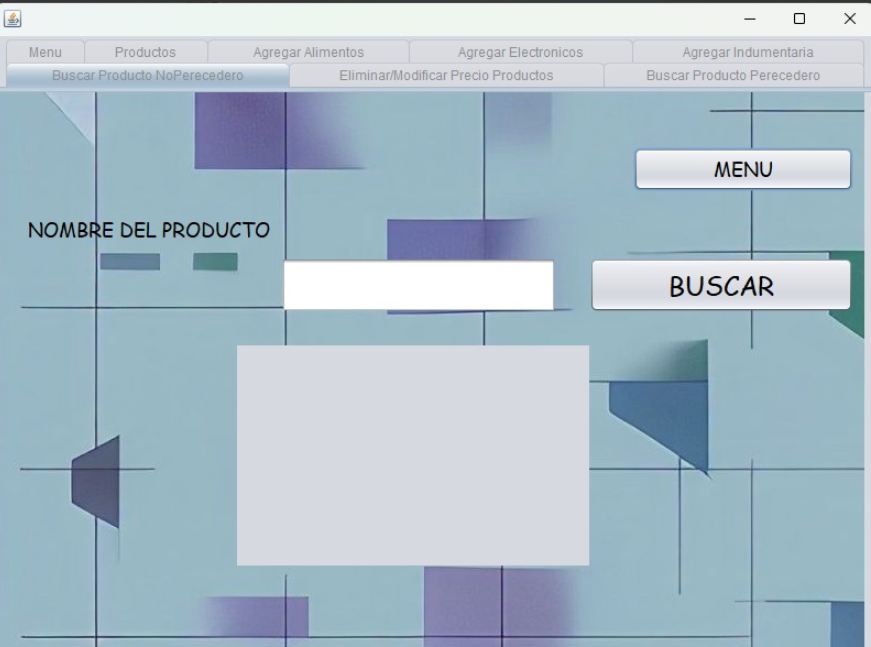
System.out.println("Material: "+material);

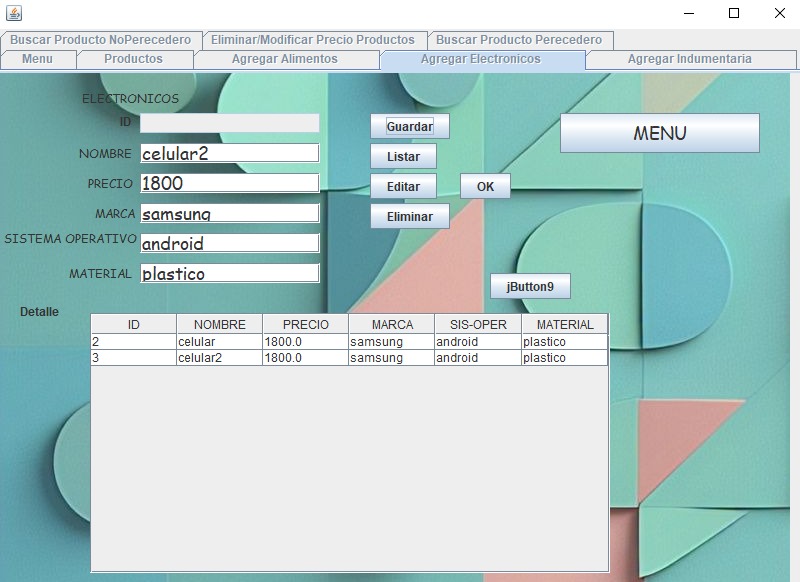
}

}

3.3 Diseño de Interfaces





Y ahí vemos como interactúan la interfaz y la base de datos

**3.5 Manejo de Archivos**

Se implementa la persistencia de datos utilizando conexiones a la base de datos MySQL. Esto asegura la disponibilidad y consistencia de los datos en operaciones concurrentes.

**3.6 Uso de Patrones de Diseño**

Se recomienda el uso de los siguientes patrones de diseño:

DAO (Data Access Object): Simplifica las operaciones CRUD, separando la lógica de acceso a datos de la lógica de negocio.

MVC (Modelo-Vista-Controlador): Garantiza una estructura modular y escalable, facilitando el mantenimiento y futuras extensiones.

**4. Pruebas del Sistema**

Se realizaron pruebas exhaustivas para verificar la funcionalidad del sistema, incluyendo:

Agregar productos de diferentes tipos.

Buscar y eliminar productos.

Modificar el precio de los productos.

Ordenar el inventario por precio y cantidad de lotes.

Verificar la correcta visualización de los productos en la interfaz gráfica.

**5. Conclusión**

El sistema desarrollado permite la gestión eficiente de electrónicos, demostrando principios de POO, modularidad, y patrones de diseño como MVC y DAO. La integración con MySQL asegura la persistencia de los datos y permite escalar el sistema para futuras ampliaciones.

El proyecto desarrollado cubre los aspectos fundamentales de la POO, ofreciendo una solución escalable y modular para la gestión de inventarios. La integración de una interfaz gráfica mejora la usabilidad del sistema, permitiendo a los usuarios realizar operaciones de manera intuitiva. El sistema es flexible y puede extenderse para incluir nuevos tipos de productos o funcionalidades adicionales en el futuro.