ASIGNATURA:

PROGRAMACIÓN ORIENTADA A OBJETOS

TRABAJO PRACTICO N.1

Sistema de Gestión de Inventario con MVC.

REALIZADO POR:

Pablo Guber Camacho Bravo

Estalyn Daniel Licuy Mecías

Nayeli Scarleth Loachamin Tipan

Deisy Abigail Quillupangui Tupe

DOCENTE:

Luis Enrique Jaramillo Montaño

SANGOLQUI - ECUADOR

Sistema de Gestión de Inventario con MVC.

1. Introducción

La implementación de un sistema de gestión de inventario es esencial para administrar los productos de una tienda de manera eficiente. En este informe, se describe el diseño de un sistema utilizando el patrón de diseño Modelo Vista Controlador (MVC). Este patrón facilita la separación de las responsabilidades en una aplicación, mejorando la escalabilidad y el mantenimiento del sistema.

2. Objetivos

Objetivo General

Desarrollar un sistema de gestión de inventario para una tienda utilizando el patrón de diseño Modelo Vista Controlador (MVC), que permita administrar eficientemente los productos, actualizar el inventario y calcular el valor total del mismo, mejorando así la gestión de los recursos.

Objetivos Específicos

- Implementar la capa del Modelo, que gestione las operaciones de registro, actualización y consulta de productos, así como el cálculo del valor total del inventario.
- Diseñar y desarrollar la Vista que permita interactuar con el usuario a través de una interfaz sencilla (como consola o GUI), mostrando la lista de productos y permitiendo la entrada de datos para registrar y actualizar el inventario.

3. Marco teórico

3.1 Análisis del proceso

El proceso de gestión de inventario en una tienda implica varias actividades clave:

- **Registro de productos:** Se ingresan nuevos productos con sus respectivas características.
- **Actualización de inventario:** Se modifican las cantidades disponibles en función de las compras y ventas.
- **Consulta de productos:** Se obtiene información actualizada de los productos en stock.
- **Cálculo del valor total:** Se determina el valor del inventario sumando el costo de cada producto multiplicado por su cantidad.

El uso del patrón MVC facilita la implementación de estas funciones al separar la lógica del negocio, la presentación de datos y la interacción del usuario.

3.2 Análisis de requisitos

Para desarrollar el sistema de gestión de inventario, se identificaron los siguientes requisitos:

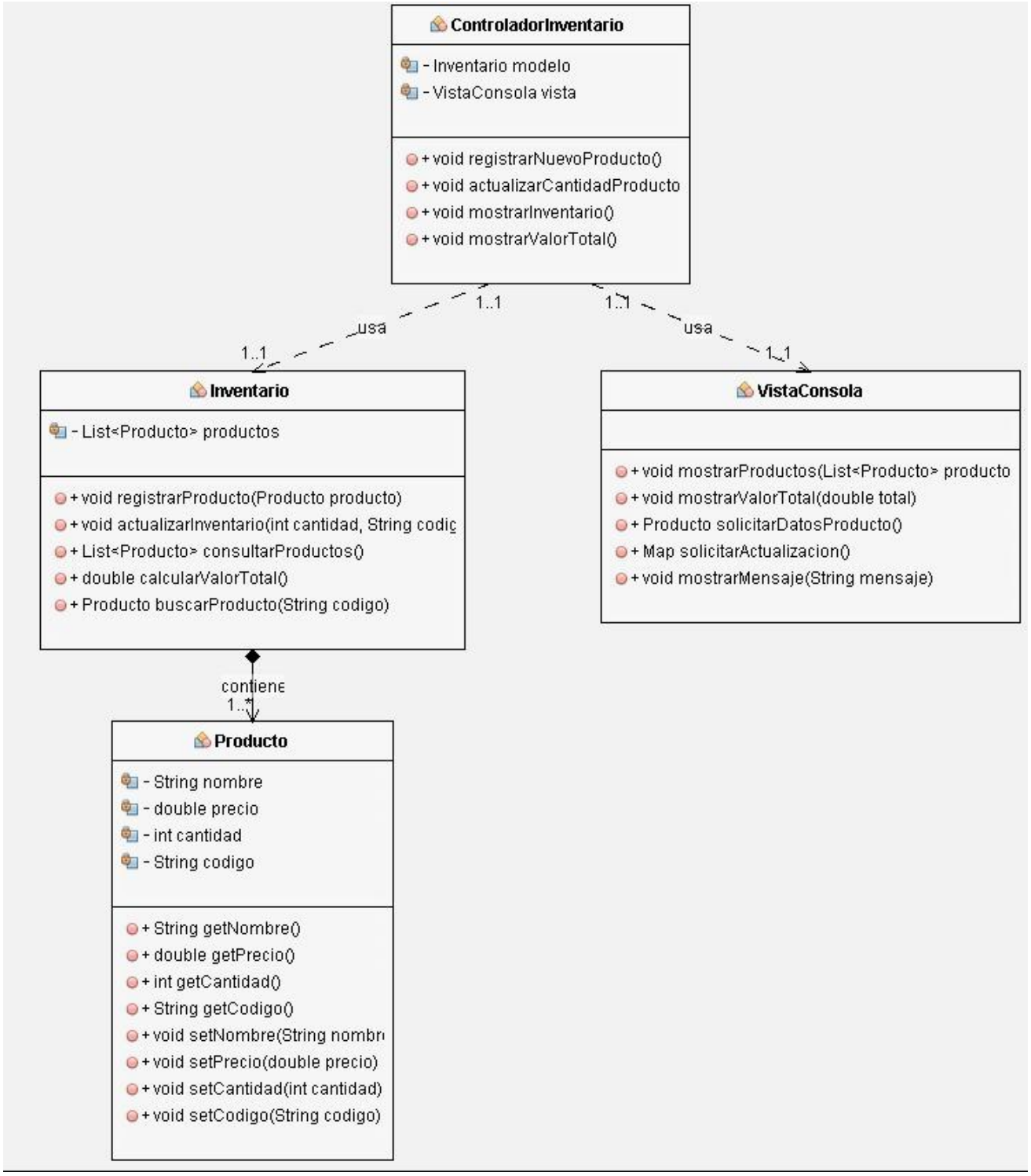
Requisitos Funcionales:

1. Permitir el registro de nuevos productos con código, nombre, precio y cantidad.
2. Permitir la actualización del inventario al modificar la cantidad de productos existentes.
3. Proporcionar una opción para consultar la lista de productos almacenados.
4. Calcular el valor total del inventario.
5. Implementar una interfaz de usuario basada en consola para interactuar con el sistema.

Requisitos No Funcionales:

1. Utilizar el patrón MVC para modularidad y facilidad de mantenimiento.
2. Implementar el sistema en Python para mayor flexibilidad.
3. Garantizar una interfaz de usuario sencilla e intuitiva.
4. Posibilidad de expansión a una interfaz gráfica en el futuro.

3.3 Diagrama UML


ControladorInventario
- Inventario modelo
- VistaConsola vista
+ void registrarNuevoProducto()
+ void actualizarCantidadProducto
+ void mostrarInventario()
+ void mostrarValorTotal()
1..1
usa
1..1
1..1
usa
1..1
Inventario
- List<Producto> productos
+ void registrarProducto(Producto producto)
+ void actualizarInventario(int cantidad, String codig
+ List<Producto> consultarProductos()
+ double calcularValorTotal()
+ Producto buscarProducto(String codigo)
VistaConsola
+ void mostrarProductos(List<Producto> producto
+ void mostrarValorTotal(double total)
+ Producto solicitarDatosProducto()
+ Map solicitarActualizacion()
+ void mostrarMensaje(String mensaje)
contiene
1..*
Producto
- String nombre
- double precio
- int cantidad
- String codigo
+ String getNombre()
+ double getPrecio()
+ int getCantidad()
+ String getCodigo()
+ void setNombre(String nombr
+ void setPrecio(double precio)
+ void setCantidad(int cantidad)
+ void setCodigo(String codigo)

3.4 Implementación del código

La implementación del sistema incluye tres componentes principales: el modelo, la vista y el controlador.

Clase producto

La clase Producto almacena la información de cada producto.

```
public class Producto {
    private int id;
    private String nombre;
    private double precio;
    private int cantidad;


    public Producto(int id, String nombre, double precio, int cantidad) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.cantidad = cantidad;
    }


    public int getId() { return id; }
    public void setId(int id) { this.id = id; }


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


    public double getPrecio() { return precio; }
    public void setPrecio(double precio) { this.precio = precio; }


    public int getCantidad() { return cantidad; }
    public void setCantidad(int cantidad) { this.cantidad = cantidad; }


    @Override
    public String toString() {
        return "ID: " + id + " | Nombre: " + nombre + " | Precio: $" + precio
+ " | Cantidad: " + cantidad;
    }
}
```

Clase inventario

La clase Inventario gestiona una lista de productos y proporciona los métodos para su manipulación.

```java
public class Inventario {
    private List<Producto> productos;


    public Inventario() {
        this.productos = new ArrayList<>();
    }


    public void agregarProducto(Producto producto) {
        productos.add(producto);
    }


    public void actualizarCantidad(int id, int nuevaCantidad) {
        for (Producto producto : productos) {
            if (producto.getId() == id) {
                producto.setCantidad(nuevaCantidad);
                break;
            }
        }
    }


    public List<Producto> obtenerProductos() {
        return new ArrayList<>(productos);
    }


    public double calcularValorTotal() {
        double total = 0;
        for (Producto producto : productos) {
            total += producto.getPrecio() * producto.getCantidad();
        }
        return total;
    }
```

```
    public Producto buscarProducto(int id) {
        for (Producto producto : productos) {
            if (producto.getId() == id) {
                return producto;
            }
        }
        return null;
    }
}
```

3.5 Funcionalidad del sistema

El sistema de inventario permite las siguientes funcionalidades:

- Registrar productos con sus respectivos códigos, nombres, precios y cantidades.
- Actualizar el inventario sumando cantidades a productos existentes.
- Consultar la lista de productos registrados.
- Calcular el valor total del inventario multiplicando la cantidad por el precio de cada producto.

3.6 Vista (VistaConsola)

La vista se encarga de interactuar con el usuario, mostrando información y recibiendo entradas.

```
public class VistaConsola {
    private Scanner scanner;


    public VistaConsola() {
        this.scanner = new Scanner(System.in);
    }


    public void mostrarMenu() {
        System.out.println("\n=== SISTEMA DE GESTIÓN DE INVENTARIO ===");
        System.out.println("1. Agregar producto");
        System.out.println("2. Actualizar cantidad");
        System.out.println("3. Mostrar inventario");
        System.out.println("4. Mostrar valor total");
        System.out.println("5. Salir");
        System.out.print("Seleccione una opción: ");
    }


    public int obtenerOpcion() {
        return scanner.nextInt();
```

```java
    }

    public Producto obtenerDatosProducto() {
        System.out.println("\n--- Agregar Nuevo Producto ---");
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Nombre: ");
        String nombre = scanner.nextLine();
        System.out.print("Precio: ");
        double precio = scanner.nextDouble();
        System.out.print("Cantidad: ");
        int cantidad = scanner.nextInt();

        return new Producto(id, nombre, precio, cantidad);
    }

    public int[] obtenerDatosActualizacion() {
        System.out.println("\n--- Actualizar Cantidad ---");
        System.out.print("ID del producto: ");
        int id = scanner.nextInt();
        System.out.print("Nueva cantidad: ");
        int cantidad = scanner.nextInt();

        return new int[]{id, cantidad};
    }

    public void mostrarInventario(List<Producto> productos) {
        System.out.println("\n--- Inventario Actual ---");
        for (Producto producto : productos) {
            System.out.println(producto);
        }
    }

    public void mostrarValorTotal(double total) {
        System.out.printf("\nValor total del inventario: $%.2f%n", total);
    }

    public void mostrarMensaje(String mensaje) {
        System.out.println(mensaje);
    }
}
```

3.7 Controlador (ControladorInventario)

El controlador coordina las interacciones entre el modelo y la vista.

```
public ControladorInventario(Inventario modelo, VistaConsola vista) {
        this.modelo = modelo;
        this.vista = vista;
    }

    public void ejecutar() {
        boolean ejecutando = true;

        while (ejecutando) {
            vista.mostrarMenu();
            int opcion = vista.obtenerOpcion();

            switch (opcion) {
                case 1:
                    Producto nuevoProducto = vista.obtenerDatosProducto();
                    modelo.agregarProducto(nuevoProducto);
                    vista.mostrarMensaje("Producto agregado exitosamente.");
                    break;

                case 2:
                    int[] datos = vista.obtenerDatosActualizacion();
                    modelo.actualizarCantidad(datos[0], datos[1]);
                    vista.mostrarMensaje("Cantidad actualizada exitosamente.")
                    break;

                case 3:
                    vista.mostrarInventario(modelo.obtenerProductos());
                    break;

                case 4:
                    vista.mostrarValorTotal(modelo.calcularValorTotal());
                    break;

                case 5:
                    ejecutando = false;
                    vista.mostrarMensaje("¡Gracias por usar el sistema!");
```

```
                    break;


                default:
                    vista.mostrarMensaje("Opción no válida. Por favor intente
de nuevo.");
            }
        }
    }
}
```

4. **Resultados**

La implementación del sistema ha demostrado ser funcional y eficiente en la gestión del inventario de una tienda. Los resultados obtenidos incluyen:

- Un sistema modular y escalable basado en el patrón MVC.
- Correcto manejo de los productos y su inventario.
- Interacción fluida entre el usuario y el sistema mediante la interfaz de consola.
- Cálculo preciso del valor total del inventario.

5. **Conclusión**

En este proyecto, aprendimos que un sistema de gestión de inventario desarrollado con el patrón Modelo Vista Controlador (MVC) ofrece una solución eficiente para la administración de productos en una tienda, simplificando tareas esenciales como el registro, la actualización y la consulta del inventario, además del cálculo del valor total del inventario. La aplicación del patrón MVC ha facilitado una separación clara de responsabilidades entre la lógica de negocio, la interacción con el usuario y el control de datos, lo cual contribuye a mejorar la escalabilidad, el mantenimiento y la flexibilidad del sistema. Además, este enfoque modular permite expandir el sistema de manera sencilla, integrando nuevas funcionalidades sin comprometer su estructura básica.

6. **Evidencias**

Copilacion del archivo

```
=== SISTEMA DE GESTIÓN DE INVENTARIO ===
1. Agregar producto
2. Actualizar cantidad
3. Mostrar inventario
4. Mostrar valor total
5. Salir
Seleccione una opción: 2

--- Actualizar Cantidad ---
ID del producto: 1
Nueva cantidad: 24
Cantidad actualizada exitosamente.
=== SISTEMA DE GESTIÓN DE INVENTARIO ===
1. Agregar producto
2. Actualizar cantidad
3. Mostrar inventario
4. Mostrar valor total
5. Salir
Seleccione una opción: 3

--- Inventario Actual ---
ID: 1 | Nombre: Chocolate | Precio: $1.0 | Cantidad: 24

=== SISTEMA DE GESTIÓN DE INVENTARIO ===
1. Agregar producto
2. Actualizar cantidad
3. Mostrar inventario
4. Mostrar valor total
5. Salir
Seleccione una opción: 4

Valor total del inventario: $24.00
=== SISTEMA DE GESTIÓN DE INVENTARIO ===
1. Agregar producto
2. Actualizar cantidad
3. Mostrar inventario
4. Mostrar valor total
5. Salir
Seleccione una opción: 5
¡Gracias por usar el sistema!
PS C:\Users\essta\Downloads\AC 1>
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Debug: Main

```
PS C:\Users\essta\Downloads\AC 1>  & 'D:\JAVA\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:56886' '-cp' 'C:\Users\essta\AppData\Roaming\Code\User\workspaceStorage\1ec48118c1ce9936f3922dbd4ca91818\redhat.java\jdt_ws\AC 1_cebcb9f3\bin' 'Main'

=== SISTEMA DE GESTIÓN DE INVENTARIO ===
1. Agregar producto
2. Actualizar cantidad
3. Mostrar inventario
4. Mostrar valor total
5. Salir
Seleccione una opción: 1

--- Agregar Nuevo Producto ---
ID: 1
Nombre: Chocolate
Precio: 1
Cantidad: 12
Producto agregado exitosamente.
```