Manual Técnico - Sistema de Gestión de Inventario para Tienda de Ropa

1. Descripción General

El Sistema de Gestión de Inventario para Tienda de Ropa es una aplicación Java de consola que permite administrar productos, registrar ventas, generar reportes y mantener un registro de actividades mediante bitácora.

2. Arquitectura del Sistema

1. Diagrama de Clases

SistemaInventarioTiendaRopa (Main)

▲

|

├── Producto

├── Venta

└── Bitacora

2. Estructura de Archivos

/

├── SistemaInventarioTiendaRopa.java //Clase principal//

├── data/ Directorio de datos

| ├── inventario.txt Datos de productos

| ├── ventas.txt Historial de ventas

| └── bitacora.txt Registro de actividades

└── reportes/ Directorio de reportes

├── [fecha]_Stock.pdf Reportes de stock

└── [fecha]_Venta.pdf Reportes de ventas

3. Especificaciones Técnicas

1. Requisitos del Sistema

- **Lenguaje**: Java 8 o superior

- **Entorno**: Cualquier SO con JVM
- **Almacenamiento**: con capacidad para 100 elementos cada uno

2. Dependencias

```
import java.util.Scanner;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;
```

4. Clases del Sistema

1. Clase Principal: SistemaInventarioTiendaRopa

Atributos:

```
private static Producto[] inventario = new Producto[100];  // Almacén de productos
private static int contadorProductos = 0;                  // Contador de productos
private static Venta[] ventas = new Venta[100];            // Registro de ventas
private static int contadorVentas = 0;                     // Contador de ventas
private static Bitacora[] bitacora = new Bitacora[100];    // Registro de bitácora
private static int contadorBitacora = 0;                   // Contador de bitácora
private static Scanner scanner = new Scanner(System.in);   // Lectura de entrada
```

Métodos Principales:

- main(): Punto de entrada del sistema
- mostrarMenuPrincipal(): Interfaz de usuario principal
- agregarProducto(): Gestión de nuevos productos
- buscarProducto(): Búsqueda en inventario
- eliminarProducto(): Eliminación de productos
- registrarVenta(): Proceso de ventas
- generarReportes(): Generación de reportes
- cargarDatos(): Carga inicial de datos
- guardarDatos(): Persistencia de datos

2. Clase Producto

Atributos:

private String codigo; // Identificador único

private String nombre; // Nombre del producto

private String categoria; // Categoría del producto

private double precio; // Precio unitario

private int cantidad; // Stock disponible

Métodos:

- Constructor con parámetros
- Getters para todos los atributos
- Setter para cantidad
- toString(): Representación en string del objeto

3. Clase Venta

Atributos:

java

private String codigoProducto; // Código del producto vendido

private int cantidad; // Cantidad vendida

private double total; // Total de la venta

private Date fecha; // Fecha de la venta

Métodos:

- Constructor con parámetros
- Getters para todos los atributos

4. Clase Bitacora

Atributos:

private String accion; // Acción realizada

private String detalle; // Detalles de la acción

private Date fecha; // Fecha y hora del registro

Métodos:

- Constructores (con y sin fecha específica)
- Getters para todos los atributos
- toString(): Formato legible del registro

5. Flujo de Datos

1. Inicialización

1. Crear directorios data/ y reportes/
2. Cargar datos desde archivos con cargarDatos()
3. Mostrar menú principal

2. Persistencia de Datos

- **Formato de inventario.txt**: codigo,nombre,categoria,precio,cantidad
- **Formato de ventas.txt**: codigoProducto,cantidad,total,fecha_milisegundos
- **Formato de bitacora.txt**: accion,detalle,fecha_milisegundos

3. Proceso de Ventas

1. Validar existencia del producto
2. Verificar stock suficiente
3. Calcular total: cantidad * precio
4. Actualizar inventario: stock = stock - cantidad
5. Registrar venta en array y archivo

6. Validaciones y Manejo de Errores

1. Validaciones de Entrada

```
// Validación de números//
try {
   cantidad = Integer.parseInt(scanner.nextLine());
   if (cantidad < 0) {
      System.out.println("Error: La cantidad debe ser positiva.");
      continue;
   }
   break;
} catch (NumberFormatException e) {
   System.out.println("Error: Ingrese un valor numérico válido.");
}
```

```
// Validación de código único
for(int i = 0; i < contadorProductos; i++) {
   if(inventario[i].getCodigo().equals(codigo)) {
      System.out.println("Error: Ya existe un producto con este código.");
      return;
   }
}
```

7. Algoritmos Implementados

1. Búsqueda Lineal

Utilizado en búsqueda de productos por código, nombre o categoría:

```
for(int i = 0; i < contadorProductos; i++) {
   if(inventario[i].getCodigo().equals(busqueda)) {
      // Producto encontrado
   }
}
```

2. Eliminación de Elementos

Desplazamiento de elementos en array:

java

```
for(int i = indice; i < contadorProductos - 1; i++) {
   inventario[i] = inventario[i + 1];
}
contadorProductos--;
inventario[contadorProductos] = null;
```

8. Formatos de Salida

1. Reportes de Stock

```
=== REPORTE DE STOCK ===
Fecha: Sat Sep 20 11:09:52 CDT 2025
================================================
C▯digo  Nombre  Categor▯a   Precio  Stock
================================================
ROPA001 camiseta basica camiseta    $15.99  16
camisa  camisa  camisa  $3.0    50

```

2. Reportes de Ventas

```
=== REPORTE DE VENTAS ===
Fecha: Sat Sep 20 11:11:18 CDT 2025
======================================================
C▯digo  Cantidad    Total   Fecha
======================================================
ROPA001 34  $543.66 13/09/2025 07:41
camisa  30  $90.0   20/09/2025 11:11
======================================================
TOTAL DE VENTAS: $633.66

```

3. Bitácora

02/01/2022 10:30:45 - Registrar Venta - Éxito: Venta de 2 unidades de ROPA001

9. Limitaciones Conocidas

1. **Capacidad fija**: limitados a 100 elementos
2. **Búsqueda secuencial**: No optimizada para grandes volúmenes de datos
3. **Persistencia simple**: Archivos de texto sin encriptación
4. **Interfaz de consola**: Sin interfaz gráfica de usuario

10. Posibles Mejoras

1. Implementar ArrayList para capacidad dinámica
2. Agregar funciones de edición de productos
3. Implementar búsqueda binaria para mejor rendimiento
4. Añadir autenticación de usuarios
5. Desarrollar interfaz gráfica
6. Implementar base de datos real

11. Métricas Técnicas

- **Líneas de código**: ~720 líneas
- **Clases**: 4
- **Métodos**: 25

12. Pruebas Unitarias Recomendadas

1. Creación y validación de productos
2. Registro y cálculo de ventas
3. Persistencia y carga de datos
4. Generación de reportes
5. Registro en bitácora

13. Mantenimiento

.1. Backup de Datos

Realizar copia periódica de la carpeta data/ para prevenir pérdida de información.

2. Monitoreo

Verificar regularmente el espacio en disco para los archivos de reportes.

3. Actualizaciones

Extender funcionalidades según necesidades específicas de la tienda.

Diagrama de flujo técnico

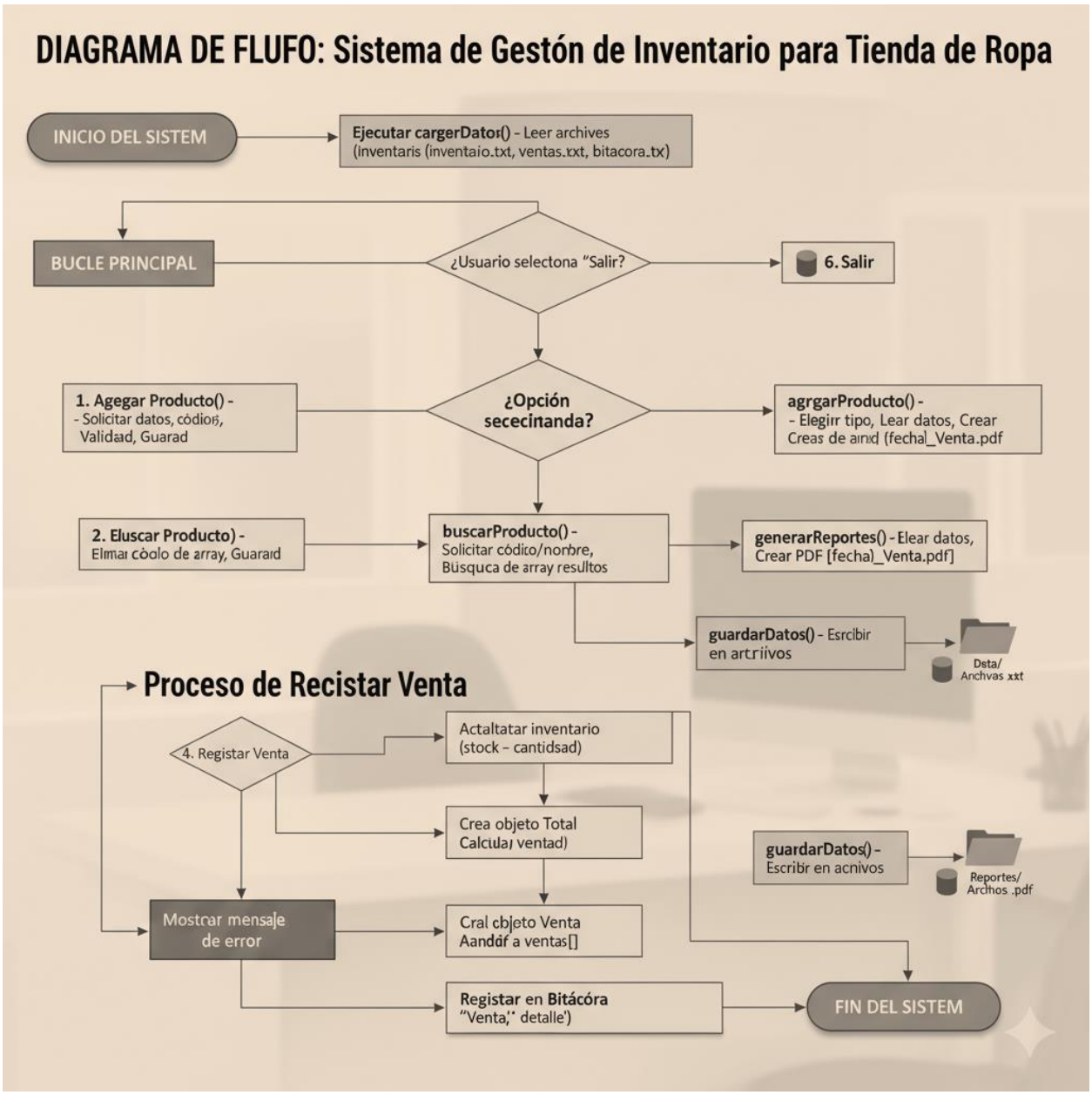
DIAGRAMA DE FLUFO: Sistema de Gestón de Inventario para Tienda de Ropa
INICIO DEL SISTEM
Ejecutar cargerDator() - Leer archives (Inventaris (inventaio.txt, ventas.txt, bitacora.tx)
BUCLE PRINCIPAL
¿Usuario selectona "Salir?
6. Salir
1. Agegar Producto() - - Solicitar datos, códios, Validaad, Guarad
¿Opción sececinanda?
agrgarProducto() - - Elegirr tipo, Lear datos, Crear Creas de aind (fechal_Venta.pdf
2. Eluscar Producto) - Elmar cóolo de array, Guarard
buscarProducto() - Solicitar códico/nonbre, Büsquca de array resultos
generarReportes() - Elear datos, Crear PDF [fecha]_Venta.pdf]
guardarDatos() - Esrcibir en artriivos
Dsta/ Anchvas xxt
Proceso de Recistar Venta
4. Registar Venta
Actaltatar inventario (stock – cantidsad)
Crea objeto Total Calcuiaj ventad)
guardarDatos() - Escribir en acnivos
Reportes/ Archos .pdf
Mostrar mensaje de error
Cral cbjeto Venta Aandáf a ventas[]
Registar en Bitácóra "Venta," detalle')
FIN DEL SISTEM