

ING SHOP



Manual Técnico

**Universidad San Carlos de Guatemala Facultad de ingeniería.
Ingeniería en ciencias y sistemas**

Introducción

El Manual Técnico del Sistema ING Shop documenta la arquitectura, diseño e implementación de una aplicación de gestión integral para tienda en línea desarrollada en Java. Este sistema representa la solución tecnológica para la administración completa de usuarios, productos, inventario, pedidos y reportes, implementando mejores prácticas de ingeniería de software y patrones de diseño establecidos.

Sancarlista Shop es una aplicación desktop desarrollada como proyecto académico que simula el funcionamiento de un sistema de comercio electrónico completo. El sistema maneja tres roles de usuario diferenciados (Administrador, Vendedor, Cliente) con permisos y funcionalidades específicas para cada uno, garantizando una separación clara de responsabilidades.

Requisitos del Sistema

Entorno de Desarrollo

Lenguaje de Programación: Java SE 21

IDE Recomendado: NetBeans IDE 24

Sistema de Control de Versiones: Github

Requisitos del Software

Sistema Operativo: Windows 10/11, macOS Monterey o superior, Linux (Ubuntu 20.04+)

Java Development Kit (JDK): Oracle JDK 21 o OpenJDK 21

Librerías **Adicionales:** itextpdf.jar

Requisitos del Hardware (recomendados)

Procesador: Intel Core i5 o superior / AMD Ryzen 5 o superior

Memoria RAM: 4 GB mínimo (8 GB recomendado)

Almacenamiento: 500 MB de espacio libre en disco para la aplicación

Tarjeta de Video: No requerida (aplicación basada en GUI simple)

ESTRUCTURA

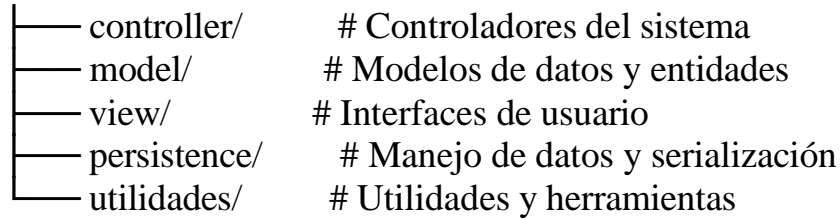
1.1. Patrón Modelo-Vista-Controlador (MVC)

El sistema implementa el patrón MVC para separar responsabilidades:

- Modelo: Clases de dominio y lógica de negocio
- Vista: Interfaces gráficas de usuario
- Controlador: Manejo de eventos y coordinación entre modelo y vista

1.2. Estructura de Paquetes

src/



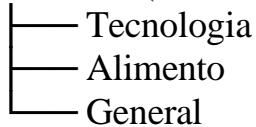
2.1. Jerarquía de Usuarios

Usuario (Abstracta)



2.2. Jerarquía de Productos

Producto (Abstracta)



2.3. Entidades Principales

3. Descripción Detallada de Clases

3.1. Paquete: model

Clase Usuario (Abstracta)

Responsabilidad: Representar la entidad base de usuario del sistema

Atributos:

- codigo: String - Identificador único
- nombre: String - Nombre completo
- contrasena: String - Contraseña encriptada

- genero: char - Género (M/F)

Métodos principales:

- autenticar(String contraseña): boolean
- getTipo(): String (abstracto)

Clase Administrador

Responsabilidad: Gestionar usuarios y productos del sistema

Métodos específicos:

- crearVendedor(Vendedor vendedor)
- eliminarVendedor(String codigo)
- cargarProductosCSV(File archivo)

Clase Vendedor

Responsabilidad: Administrar stock y confirmar pedidos

Atributos adicionales:

- ventasConfirmadas: int

Métodos específicos:

- agregarStock(String codigoProducto, int cantidad)
- confirmarPedido(String codigoPedido)
- crearCliente(Cliente cliente)

Clase Cliente

Responsabilidad: Realizar pedidos y gestionar carrito

Atributos adicionales:

- fechaCumpleanos: Date
- carrito: Carrito

Métodos específicos:

- agregarAlCarrito(Producto producto, int cantidad)
- realizarPedido()

Clase Producto (Abstracta)

Responsabilidad: Representar productos del catálogo

Atributos:

- codigo: String
- nombre: String
- categoria: String
- stock: int
- precio: double

Métodos abstractos:

- getAtributoEspecifico(): String

Clases Específicas de Producto

Tecnologia:

- mesesGarantia: int

Alimento:

- fechaCaducidad: Date

General:

- material: String

Clase Pedido

Responsabilidad: Representar órdenes de compra

Atributos:

- codigo: String
- fechaGeneracion: Date

- cliente: Cliente
- estado: String
- lineasPedido: List<LineaPedido>
- total: double

Clase Carrito

Responsabilidad: Gestionar productos seleccionados temporalmente

Atributos:

- items: Map<Producto, Integer>
- subtotal: double

3.2. Paquete: controller

AuthController

Responsabilidad: Manejar autenticación de usuarios

Métodos principales:

- iniciarSesion(String codigo, String contrasena): boolean
- cerrarSesion()
- validarCredenciales(String codigo, String contrasena): boolean

AdminController

Responsabilidad: Gestionar operaciones de administrador

Métodos principales:

- crearVendedor(String codigo, String nombre, char genero, String contrasena)
- actualizarVendedor(String codigo, String nombre, String contrasena)
- eliminarVendedor(String codigo)

- cargarVendedoresCSV(File archivo)
- crearProducto(Producto producto)
- actualizarProducto(String codigo, String nombre, String atributo)

VendedorController

Responsabilidad: Gestionar operaciones de vendedor

Métodos principales:

- agregarStock(String codigoProducto, int cantidad)
- cargarStockCSV(File archivo)
- crearCliente(String codigo, String nombre, char genero, Date cumpleaños, String contraseña)
- confirmarPedido(String codigoPedido)
- generarReporteIngresos(): File

ClienteController

Responsabilidad: Gestionar operaciones de cliente

Métodos principales:

- agregarAlCarrito(String codigoProducto, int cantidad)
- eliminarDelCarrito(String codigoProducto)
- actualizarCantidadCarrito(String codigoProducto, int cantidad)
- realizarPedido(): boolean
- obtenerHistorialCompras(): List<Pedido>

ReporteController

Responsabilidad: Generar reportes del sistema

Métodos principales:

- generarReporteProductosMasVendidos(): File
- generarReporteProductosMenosVendidos(): File
- generarReporteInventario(): File
- generarReporteVentasVendedor(): File
- generarReporteClientesActivos(): File
- generarReporteFinanciero(): File
- generarReporteProductosPorCaducar(): File

BitacoraController

Responsabilidad: Registrar y consultar bitácora del sistema

Métodos principales:

- registrarEvento(String usuarioTipo, String codigoUsuario, String operacion, String estado, String descripcion)
- filtrarBitacora(Date desde, Date hasta, String usuarioTipo, String operacion): List<Bitacora>
- exportarBitacoraCSV(File archivo)
- exportarBitacoraPDF(File archivo)

3.3. Paquete: persistence

DataManager

Responsabilidad: Gestionar persistencia de datos

Métodos principales:

- guardarUsuarios(List<Usuario> usuarios)
- cargarUsuarios(): List<Usuario>
- guardarProductos(List<Producto> productos)
- cargarProductos(): List<Producto>

- guardarPedidos(List<Pedido> pedidos)
- cargarPedidos(): List<Pedido>

Serializer

Responsabilidad: Serializar y deserializar objetos

Métodos principales:

- serializar(Object obj, String archivo)
- deserializar(String archivo): Object

CSVUtils

Responsabilidad: Utilidades para manejo de archivos CSV

Métodos principales:

- leerCSV(File archivo): List<String[]>
- escribirCSV(File archivo, List<String[]> datos)
- validarFormatoCSV(File archivo, String[] encabezados): boolean

3.4. Paquete: utilidades

PDFGenerator

Responsabilidad: Generar reportes en formato PDF

Métodos principales:

- generarPDF(String titulo, List<String> contenido, String archivoSalida)
- generarTablaPDF(String[] encabezados, List<String[]> datos, String archivoSalida)

Monitor

Responsabilidad: Monitoreo de sesiones activas

Métodos principales:

- `iniciarMonitoreo()`
- `detenerMonitoreo()`
- `actualizarContadores()`

Bitacora

Responsabilidad: Registro centralizado de eventos

Métodos principales:

- `log(String nivel, String mensaje)`
- `obtenerRegistros(Date desde, Date hasta): List<String>`

3.5. Paquete: view

VentanaLogin

Responsabilidad: Interfaz de autenticación

Componentes principales:

- Campos: código y contraseña
- Botón: Iniciar Sesión
- Manejo de errores de autenticación

VentanaPrincipal

Responsabilidad: Ventana principal según rol de usuario

Componentes principales:

- Menú de navegación
- Barra de estado
- Opción de cerrar sesión

Ventanas Específicas por Rol

VentanaAdmin:

- Pestañas: Vendedores, Productos, Reportes

- Operaciones CRUD para vendedores y productos
- Generación de reportes

VentanaVendedor:

- Pestañas: Productos, Clientes, Pedidos
- Gestión de stock
- Confirmación de pedidos

VentanaCliente:

- Pestañas: Productos, Carrito Compra, Historial Compras
- Catálogo de productos
- Gestión de carrito

4. Flujo de Datos

4.1. Autenticación

text

VentanaLogin → AuthController → DataManager → Serializer

4.2. Gestión de Productos

text

VentanaAdmin → AdminController → Producto → DataManager

4.3. Procesamiento de Pedidos

text

VentanaCliente → ClienteController → Carrito → Pedido →
VendedorController

5. Manejo de Archivos

5.1. Serialización

- Usuarios: usuarios.ser

- Productos: productos.ser
- Pedidos: pedidos.ser
- Bitácora: bitacora.ser

5.2. Archivos CSV

- Vendedores: Formato específico con encabezados
- Productos: Formato por categoría
- Stock: Código y cantidad
- Clientes: Datos personales completos

6. Hilos y Procesamiento Concurrente

6.1. Monitor de Sesiones Activas

- Actualización cada 10 segundos
- Contador de usuarios conectados

6.2. Simulador de Pedidos Pendientes

- Actualización cada 8 segundos
- Monitoreo de cola de pedidos

6.3. Generador de Estadísticas

- Actualización cada 15 segundos
- Métricas del sistema en tiempo real

7. Validaciones y Manejo de Errores

7.1. Validaciones de Entrada

- Códigos únicos
- Formatos de fecha
- Stock disponible
- Credenciales de usuario

7.2. Manejo de Excepciones

- ArchivoNoEncontradoException
- FormatoCSVInvalidoException
- UsuarioNoEncontradoException
- StockInsuficienteException

8. Consideraciones Técnicas

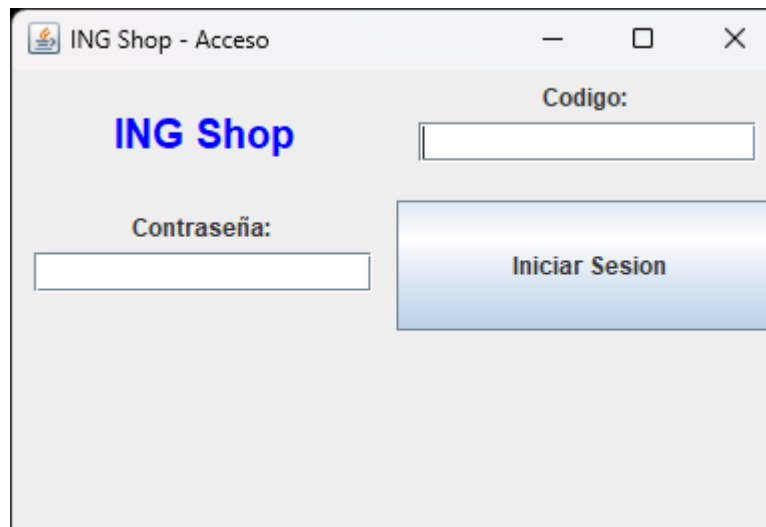
8.1. Restricciones Implementadas

- No uso de ArrayList, HashMap, List (usando arrays nativos)
- Implementación manual de estructuras de datos
- Validación de código original

8.2. Patrones de Diseño

- MVC para separación de responsabilidades
- Factory Method para creación de productos
- Observer para monitoreo en tiempo real
- Singleton para gestión de datos

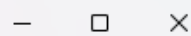
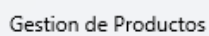
Interfaz



The screenshot shows a window titled "ING Shop - Acceso". On the left, the text "ING Shop" is displayed in blue. To its right is a text input field labeled "Codigo:". Below the "ING Shop" text is a text input field labeled "Contraseña:". To the right of the password field is a blue button with the text "Iniciar Sesion".



The screenshot shows a window titled "Modulo Administrador - ING Shop". The main heading is "Modulo Administrador" in green. Below this heading are four blue buttons stacked vertically: "Gestion de Vendedores", "Gestion de Productos", "Generar Reportes", and "Cerrar Sesion".



Crear Producto

--

--

Tecnologia

--

Cargar CSV

--Productos Registrados--

No hay Productos registrados. Use el Formulario para Ingresar Productos

Eliminar

[Ver Details](#)



Generar Reportes



SELECCIONE TIPO DE REPORTE

Productos Mas Vendidos

Productos Menos Vendidos

Reporte Inventario

Ventas por Vendedor

Clientes Activos

Reporte Financiero

Productos por Caducar

Código

The screenshot shows the IntelliJ IDEA IDE with the 'ControladorAdmin.java' file open. The left sidebar displays the project structure for 'Sancarlita_Shop', including folders like 'idea', 'datos', 'Librerias', 'out', 'reports', 'src', 'controlador', 'interfaces', 'OperacionesCRUD', 'main', 'META-INF', 'modelo', 'utilidades', and 'vista'. The 'Controlador' folder is expanded, showing 'ControladorAdmin', 'ControladorCliente', 'ControladorLogin', 'ControladorReportes', and 'ControladorVendedor'. The 'ControladorAdmin.java' file is selected, and its code is displayed in the main editor. The code defines the 'ControladorAdmin' class, which manages the application's state and interactions. It includes private attributes for 'vista', 'productos', 'totalProductos', 'vistaAdmin', 'vistaVendedores', 'vistaActualizarVendedor', 'vistaBitacora', 'usuarios', and 'totalUsuarios'. The 'ControladorAdmin' class has a constructor that initializes these attributes and calls 'cargarDatos()' and 'configurarListenerAdmin()'. It also has a 'configurarListenerAdmin()' method that sets up listeners for the 'vistaAdmin' and 'vistaVendedores' objects. The code is written in Java and uses standard IDE conventions for formatting and commenting.

```
1 package controlador;
2
3 > import ...
4
5
6
7
8
9
10
11
12
13 public class ControladorAdmin { 2 usages
14
15     private vista.VistaGestionProductos vistaProductos; 15 usages
16     private modelo.Producto[] productos; 20 usages
17     private int totalProductos; 19 usages
18
19     private VistaAdmin vistaAdmin; 14 usages
20     private VistaGestionVendedores vistaVendedores; 13 usages
21     private VistaActualizarVendedor vistaActualizarVendedor; 14 usages
22     private vista.VistaBitacora vistaBitacora; 12 usages
23     private Usuario[] usuarios; 17 usages
24     private int totalUsuarios; 16 usages
25
26     public ControladorAdmin(vista.VistaAdmin vistaAdmin, modelo.Usuario[] usuarios, int totalUsuarios) { 1 usage
27         this.vistaAdmin = vistaAdmin;
28         this.usuarios = usuarios;
29         this.totalUsuarios = totalUsuarios;
30         configurarListenerAdmin();
31
32         this.productos = new modelo.Producto[100];
33         this.totalProductos = 0;
34
35         cargarDatos();
36         configurarListenerAdmin();
37     }
38
39     public void configurarListenerAdmin() { 2 usages
40         vistaAdmin.setGestionVendedoresL(new ActionListener() {
41             @Override
42             public void actionPerformed(ActionEvent e) { abrirGestionVendedores(); }
43         });
44
45         vistaAdmin.setCerrarSesionL(new ActionListener() {
46             @Override
47             public void actionPerformed(ActionEvent e) { cerrarSesion(); }
48         });
49
50         vistaAdmin.setGestionProductosL(new ActionListener() {
```

The screenshot shows the IntelliJ IDEA IDE with the 'Main.java' file open. The left sidebar displays the project structure for 'Sancarlita_Shop', including folders like 'idea', 'datos', 'Librerias', 'out', 'reports', 'src', 'controlador', 'interfaces', 'OperacionesCRUD', 'main', 'META-INF', 'modelo', 'utilidades', and 'vista'. The 'Main' folder is expanded, showing 'Main', 'Carrito', 'Cliente', 'ItemCarrito', 'Pedido', 'Producto', 'ProductoAlimento', 'ProductoGeneral', 'ProductoTecnologia', 'Reporte', 'Usuario', and 'Vendedor'. The 'Main.java' file is selected, and its code is displayed in the main editor. The code defines the 'Main' class, which is the entry point of the application. It includes a 'main' method that calls 'iniciarAplicacion()' and a 'run' method that calls 'iniciarAplicacion()'. The 'iniciarAplicacion()' method initializes the application, including creating the 'Login' object, the 'ControladorLogin' object, and the 'Controlador' object. It also prints out the status of the application. The code is written in Java and uses standard IDE conventions for formatting and commenting.

```
1 package main;
2
3 > import ...
4
5
6
7
8 public class Main { 1 usage
9     public static void main(String[] args) {
10         System.out.println("Iniciando sistema...");
11
12         SwingUtilities.invokeLater(new Runnable() {
13             @Override
14             public void run() { iniciarAplicacion(); }
15         });
16     }
17
18     private static void iniciarAplicacion() { 1 usage
19         System.out.println("Iniciando interfaz...");
20
21         HilosSistema.iniciarHilos();
22
23         try {
24             Login vistaLogin = new Login();
25             ControladorLogin controlador = new ControladorLogin(vistaLogin);
26             controlador.iniciar();
27
28             System.out.println("Interfaz iniciada");
29         } catch (Exception e) {
30             System.err.println("Error: " + e.getMessage());
31             e.printStackTrace();
32         }
33     }
34 }
35
36 }
```

Project ▾

- SanCarlista_Shop C:\Users\Emanuel\Desktop\Proyecto2_San_Ca
 - idea
 - datos
 - Librerias
 - out
 - reportes
 - src
 - controlador
 - datos
 - interfaces
 - main
 - Main
 - META-INF
 - modelo
 - utilidades
 - ArchivosCSV
 - Bitacora
 - GeneradorPDF
 - HilosSistema
 - Serializador
 - Validaciones
 - vista
 - .gitignore
 - SanCarlista_Shop.iml
- External Libraries
- Scratches and Consoles

Main.java ControladorAdmin.java Carrito.java ArchivosCSV.java

```
1 package utilidades;
2
3 > import ...
4
11 public class ArchivosCSV { 4 usages
12
13 @
14     public static Vendedor[] cargarVendedorDesdeCSV(String rutaArchivo) { 1 usage
15         Vendedor[] vendedores = new Vendedor[50];
16         int totalCargados = 0;
17
18         try {
19             BufferedReader lector = new BufferedReader(new FileReader(rutaArchivo));
20             String linea;
21             int numeroLinea = 0;
22
23             while ((linea = lector.readLine()) != null && totalCargados < vendedores.length) {
24                 numeroLinea++;
25
26                 if (numeroLinea == 1) continue;
27                 if (linea.trim().isEmpty()) continue;
28
29                 String[] datos = linea.split(regex: "\\s");
30                 if (datos.length != 4) continue;
31
32                 String codigo = datos[0].trim();
33                 String nombre = datos[1].trim();
34                 String genero = datos[2].trim().toUpperCase();
35                 String contrasenia = datos[3].trim();
36
37                 if (codigo.isEmpty() || nombre.isEmpty() || (!genero.equals("M") && !genero.equals("F")) || contrasenia.length() < 4) {
38                     continue;
39                 }
40
41                 vendedores[totalCargados] = new Vendedor(codigo, nombre, genero, contrasenia);
42                 totalCargados++;
43             }
44             lector.close();
45
46             Vendedor[] resultado = new Vendedor[totalCargados];
47             for (int i = 0; i < totalCargados; i++) {
48                 resultado[i] = vendedores[i];
49             }
50         }
51     }
52 }
```

SanCarlista_Shop > src > utilidades > ArchivosCSV 11:14 CRLF UTF-8 4 spaces

Project ▾

- SanCarlista_Shop C:\Users\Emanuel\Desktop\Proyecto2_San_Ca
 - idea
 - datos
 - Librerias
 - out
 - reportes
 - src
 - controlador
 - datos
 - interfaces
 - main
 - Main
 - META-INF
 - modelo
 - utilidades
 - ArchivosCSV
 - Bitacora
 - GeneradorPDF
 - HilosSistema
 - Serializador
 - Validaciones
 - vista
 - .gitignore
 - SanCarlista_Shop.iml
- External Libraries
- Scratches and Consoles

Main.java ControladorAdmin.java Carrito.java ArchivosCSV.java GeneradorPDF.java

```
1 package utilidades;
2
3 > import ...
4
11 public class GeneradorPDF { 2 usages
12
13 @
14     public static boolean generarPDF(Reporte reporte) { 1 usage
15         String nombreArchivo = "reportes/" + reporte.getNombreArchivo();
16
17         try {
18             PdfWriter writer = new PdfWriter(nombreArchivo);
19             PdfDocument pdf = new PdfDocument(writer);
20             Document document = new Document(pdf);
21
22             document.add(new Paragraph(reporte.getTitulo()).setFontSize(10));
23             document.add(new Paragraph(text: "Fecha: " + reporte.getFechaGeneracion()));
24             document.add(new Paragraph(text: " "));
25
26             String contenido = reporte.generarContenido();
27             document.add(new Paragraph(contenido));
28
29             document.close();
30
31             Bitacora.registrarOperacion(tipoUsuario: "SISTEMA", codigoUsuario: "ADMIN", operacion: "GENERAR_REPORTES", estado: "EXITOSO", description: "Reporte: " + nombreArchivo);
32             return true;
33         } catch (IOException e) {
34             Bitacora.registrarOperacion(tipoUsuario: "SISTEMA", codigoUsuario: "ADMIN", operacion: "GENERAR_REPORTES", estado: "FALLIDO", description: "Error: " + e.getMessage());
35             return false;
36         }
37     }
38 }
39 }
```

SanCarlista_Shop > src > utilidades > GeneradorPDF 12:14 CRLF UTF-8 4 spaces

