

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR

JOSÉ MIGUEL CARRERA

**SISTEMA DE CONTROL DE STOCK, COMPRA
Y VENTA PARA EMPRESA “EL ARCA”**

TRABAJO DE TITULACIÓN
PARA OPTAR AL TÍTULO DE
TÉCNICO UNIVERSITARIO
EN INFORMÁTICA

Alumnos:

Owen Araus Araus

Sebastián Zanetta Ibacache

Profesor guía:

Ricardo Cahe Cabach

DEDICATORIAS

Dedicado a mi familia, a mi madre por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, gracias por todo tu amor incondicional, a mi pareja que vivió todo mi proceso universitario, que estuvo conmigo en las buenas y en las malas, apoyándome en todo los casos, a mis abuelos que sin su comprensión no estaría donde estoy, y por ultimo usted Elia Rossi Batto (QEPD), usted fue la que me insto a superarme académicamente, usted fue la primera en confiar en mí y en mis habilidades, este trabajo se la dedico a ustedes.

- Owen Araus Araus

Agradezco primero que todo a Dios, por la vida y por esta oportunidad; a mi madre y abuela, por ser mi sustento principal y apoyo incondicional durante todo este tiempo; a cada profesor de la enseñanza básica, media y de la UTFSM por haber compartido su conocimiento, a cada uno de mis amigas y amigos que incondicionalmente estuvieron apoyándome en este proceso y a cada persona que estuvo ahí para ayudarme en este proceso.

Dedicado especialmente a la memoria de Agustín Ibacache Montiel, Juan Carlos Ibacache Lizama y Alicia Moreno Molina (QEPD), quienes durante su vida fueron parte fundamental de este crecimiento personal.

A todos ustedes les dedico este trabajo.

- Sebastián Zanetta Ibacache

RESUMEN

KEYWORDS: Control de existencia - Compra y venta – Mantenedor - Login - Paquetería El Arca

El sistema se realizó para la señora Gloria Vega Lizama, empresaria dedicada a la venta de paquetería de la V región, Villa Alemana.

El “Sistema De Control De Stock, Compra Y Venta para Empresa ‘El Arca’ ” fue desarrollado durante todo un periodo académico (10 meses), mediante la herramienta de programación IDE NetBeans y el gestor de base de datos AppServ, para poder trabajar con MySQL.

El trabajo será desarrollado bajo los siguientes temas: índice, introducción, capítulos, conclusión, bibliografía, y anexos.

A continuación se describirá el contenido de cada uno de los capítulos del presente trabajo de titulación:

CAPÍTULO 1: Describe los aspectos relevantes del diseño lógico como: La descripción organizacional de la empresa, su situación actual, los problemas detectados, y la descripción del sistema propuesto con sus objetivos, beneficios e información que manejará.

CAPÍTULO 2: Describe los aspectos relevantes al diseño físico, como la descripción de la configuración del sistema y del software que se utilizarán en el sistema, y la descripción detallada de todos los archivos.

CAPÍTULO 3: Describe los programas que comprenderán el sistema, presentando un diagrama de modular y un diagrama de menú, y la descripción detallada de todos los programas desarrollados.

Y por último, se presentan las conclusiones del trabajo, bibliografía y un anexo con los códigos fuente más relevantes del sistema.

ÍNDICES

INTRODUCCIÓN.....	1
<u>CAPÍTULO 1: ASPECTOS RELEVANTES DEL ANÁLISIS.....</u>	2
1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN.....	3
1.2. DESCRIPCIÓN DE LA SITUACIÓN ACTUAL.....	4
1.2.1. Cuaderno de inventario.....	5
1.2.2. Libro de compra-venta.....	5
1.3. PROBLEMAS DETECTADOS.....	6
1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO.....	7
1.4.1. Objetivo.....	7
1.4.1.1. Objetivo general del sistema.....	7
1.4.1.2. Objetivo específicos del sistema.....	8
1.4.2. Beneficios del sistema.....	8
1.4.3. Descripción general del sistema.....	9
1.4.4. Diagrama de flujo administrativo.....	10
1.4.5. Funcionalidades del sistema.....	11
1.4.6. Entradas.....	13
1.4.7. Salidas	13
1.4.8. Sistema de codificación.....	15
1.4.9. Entidades.....	15
1.4.10. Condicionantes de diseño	17
1.4.11. Modelo relacional de datos.....	17
<u>CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS.....</u>	19
2.1. DESCRIPCIÓN DEL HARDWARE Y SOFTWARE A UTILIZAR EN LA EMPRESA.....	20
2.1.1. Configuración del sistema.....	20
2.1.2. Software a utilizar.....	20
2.1.2.1. Sistema operativo.....	21
2.1.2.2. Software utilitario.....	21

2.1.2.3.	Software de desarrollo.....	21
2.2.	DESCRIPCIÓN DE TABLAS	21
2.2.1.	Tipos de datos	21
2.2.2.	Descripción de archivos.....	22
2.2.2.1.	Venta	22
2.2.2.2.	Boleta.....	23
2.2.2.3.	Detalle_boleta.....	23
2.2.2.4.	Usuario.....	24
2.2.2.5.	Producto.....	25
2.2.2.6.	Medida	26
2.2.2.7.	Proveedor	26
2.2.2.8.	Ajuste_inventario.....	27
2.2.2.9.	Cliente.....	28
2.2.2.10.	Factura.....	29
2.2.2.11.	Detalle_factura.....	30
2.2.2.12.	Catálogo.....	30
2.2.2.13.	Factura_compra.....	31
2.2.2.14.	Detalle_factura_compra.....	32
<u>CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS.....</u>		33
3.1.	DIAGRAMA MODULAR.....	34
3.2.	DIAGRAMA DE MENÚS.....	36
3.3.	TOTALIDAD DE PROGRAMAS.....	37
3.4.	DESCRIPCIÓN DETALLA DE PROGRAMAS	38
3.4.1.	Inicio de sesión.....	38
3.4.2.	Gestión de venta.....	39
3.4.3.	Gestión de compra.....	40
3.4.4.	Mantenedor de producto.....	42
3.4.5.	Mantenedor de proveedor.....	43

3.4.6.	Mantenedor de cliente.....	44
3.4.7.	Mantenedor de usuario.....	46
CONCLUSIONES.....		47
BIBLIOGRAFÍA.....		48
LINKOGRAFÍA.....		49
ANEXOS.....		50
ANEXOS 1: CÓDIGO FUENTE DE LA CLASE CONECTAR.....		51
ANEXOS 2: CÓDIGO FUENTE DE PROGRAMA VENTA		52
ANEXOS 3: CÓDIGO FUENTE DE MANTENEDOR DE PRODUCTO.....		70

ÍNDICES DE FIGURAS

Figura 1-1:	Organigrama Tienda “El Arca”	4
Figura 1-2:	DFA Proceso de Venta.....	10
Figura 1-3:	Modelo Relacional del Sistema Propuesto.....	18
Figura 3-1:	Diagrama Modular General.....	34
Figura 3-2:	Diagrama Modular de Listados.....	34
Figura 3-3:	Diagrama Modular de Mantenedores.....	35
Figura 3-4:	Diagrama Estructura de Menú del Sistema.....	36
Figura 3-5:	Diagrama de Bloques Inicio de Sesión.....	38
Figura 3-6:	Formulario Inicio de Sesión.....	39
Figura 3-7:	Diagrama de Bloques Gestión de Venta.....	39
Figura 3-8:	Formulario Gestión de Venta.....	40
Figura 3-9:	Diagrama de Bloques Gestión de Compra.....	41
Figura 3-10:	Formulario Gestión de Compra.....	42
Figura 3-11:	Diagrama de Bloques Mantenedor de Producto.....	42
Figura 3-12:	Formulario Mantenedor de Producto.....	43
Figura 3-13:	Diagrama de Bloques Mantenedor de Proveedor.....	43
Figura 3-14:	Formulario Mantenedor de Proveedor.....	44
Figura 3-15:	Diagrama de Bloques Mantenedor de Cliente.....	45

Figura 3-16:	Formulario Mantenedor de Cliente.....	45
Figura 3-17:	Diagrama de Bloques Mantenedor de Usuario.....	46
Figura 3-18:	Formulario Mantenedor de Usuario.....	46

ÍNDICES DE TABLAS

Tabla 2-1:	Tipos de Datos Numéricos.....	21
Tabla 2-2:	Tipos de Datos Caracteres o Cadenas.....	22
Tabla 2-3:	Tipos de Datos Fecha y Tiempo.....	22
Tabla 2-4:	Venta.....	22
Tabla 2-5:	Boleta.....	23
Tabla 2-6:	Detalle_boleta.....	24
Tabla 2-7:	Usuario.....	24
Tabla 2-8:	Producto.....	25
Tabla 2-9:	Medida.....	26
Tabla 2-10:	Proveedor.....	26
Tabla 2-11:	Ajuste_inventario.....	27
Tabla 2-12:	Cliente.....	28
Tabla 2-13:	Factura.....	29
Tabla 2-14:	Detalle_factura.....	30
Tabla 2-15:	Catalogo.....	31
Tabla 2-16:	Factura_compra.....	31
Tabla 2-17:	Detalle_factura_compra.....	32
Tabla 3-1:	Listado de Archivos con Descripción.....	37

INTRODUCCIÓN

El presente trabajo detallará todo el proceso de construcción de un sistema de información automatizado el cual se le denomina “SISTEMA DE CONTROL DE STOCK, COMPRA Y VENTA” para ser implementado en la tienda en “El Arca”.

El proyecto nace de la solicitud de la dueña del negocio quien requiere un sistema que controle y de orden a los productos de la empresa por medio de un sistema automatizado de control de stock, el cual será el centro del trabajo a realizar.

La microempresa donde se realizará el proyecto, “El Arca” es una local que se dedica a la venta de paquetería entre otros productos variados, ubicado en calle Santiago #780, local 11, en pleno centro de Villa Alemana a cargo de su administradora la señora Gloria Vega Lizama.

La finalidad del proyecto presentado es pretender dar solución a las necesidades de la empresa y así disminuir los tiempos en la gestión de compra y venta del local, generando así mayor productividad, además de generar informes para la toma de decisiones.

El sistema será construido con la herramienta de programación IDE NetBeans para trabajar con lenguaje Java y un motor de Base de Datos MySql administrado por el programa AppServ.

CAPÍTULO 1: ASPECTOS RELEVANTES DEL ANÁLISIS

1. ASPECTOS RELEVANTES DEL ANÁLISIS

1.1. Descripción de la organización

La paquetería y tienda de artesanía “El Arca” es una distribuidora de artesanía, paquetería y regalos, fundada y administrada desde sus inicios por la señora Gloria Vega Lizama, este consta de un único punto de venta y una bodega central, ambas ubicadas en calle Santiago #780, local 11, en pleno centro de Villa Alemana.

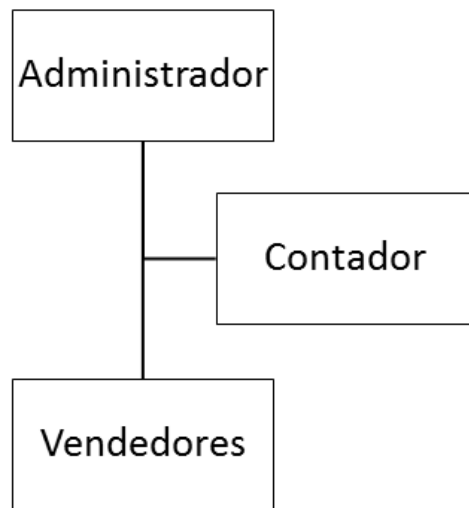
La empresa se encuentra funcionando desde 1998, esta nació debido a la necesidad de las fundadoras de conseguir un ingreso económico y la idea de la artesanía nació debido a 2 factores: el primero consistía en la ausencia de negocios de este tipo en la zona y el segundo fueron las posibilidades que le ofrecían los proveedores de obtener buena mercadería de ese tipo a un precio adecuado al presupuesto. Durante los primeros años se dedicaron a la venta de artículos de mimbre y manualidades, y todo ello ubicado en un galpón que se encontraba en el inmueble que estaba al lado de su dirección actual. Años después adquirieron el local actual, en donde se amplió la variedad y cantidad de productos para la venta.

El local mencionado se mantuvo igual hasta el 2016, cuando la infraestructura sufrió remodelaciones, ya que éste y locales aledaños fueron demolidos para construir una galería. Actualmente, el negocio se encuentra funcionando en uno de los locales de la galería mencionada.

El personal está conformado por su administradora y dos vendedoras, cuyos turnos los reparten por acuerdo de palabra. Cabe destacar que todos los miembros de la empresa con excepción del contador son familiares.

Las ventas se registran en un cuaderno de ventas, para luego ser traspasadas al libro de compra-venta, anualmente el contador realiza al menos un balance general y la administradora y vendedoras realizan un inventario de todas las existencias.

A continuación se presenta el organigrama de la empresa:



Fuente: Propia

Figura 1-1: Organigrama Tienda "El Arca"

1.2. Descripción de la situación actual

Esta empresa se ha mantenido estable con el paso de los años, pero la falta de un sistema informático ha provocado fallas en sus funciones principales de compra, venta, control de existencia, de las cuales se ha podido evidenciar la falta de precisión al momento de inventariar productos, de la cual cabe resaltar que todo material es registrado de forma manual en cuaderno, generando así inconsistencias y redundancias, esto debido a la gran variedad de productos que vende la empresa, además de no depender de un buen sistema de control de existencias. Por otro lado, la ausencia de un sistema de venta, ha generado una disminución en la agilidad, al momento de existir una alta demanda, al no tener un control inmediato de existencias.

Entre los procesos que la empresa efectúa se encuentran:

- Compra: El empleador compra productos a los proveedores, el proveedor genera una factura en donde se incluye el detalle de esta transacción.

- Venta: El personal vende el producto adquirido a clientes, éstas se realizan por boleta o por factura.
- Control de existencias: Los trabajadores del local realizan un inventario manual, en donde se registra en un cuaderno el stock de cada producto.

Los documentos utilizados son los siguientes, según el propósito al cual sirven:

1.2.1. Cuaderno de inventario

Este cuaderno es bastante complejo, por la cantidad de información que almacena, lo cual dificulta llevar un orden en su escritura. Básicamente se anota la siguiente información:

- Nombre del producto
- Tamaño
- Tipo
- Cantidad
- Color

El cuaderno carece de un orden reconocible a simple vista, presentando un formato poco claro, por esta razón es difícil mostrar gráficamente un ejemplo, ya que ésta no sería lo suficiente representativa de todos los tipos de anotaciones que se usan en este cuaderno.

1.2.2. Libro de compra-venta

En el libro de compra-venta, se registran en una página todas las facturas de compras recibidas en un mes y en la plana opuesta, todas las ventas mensuales, entre las que se incluyen los totales diarios vendidos con boleta y las facturas de venta, en cada página del libro se registrará el mes y año en que se está trabajando.

Para cada factura de compra se incluyen los siguientes datos:

- Día
- Número de factura
- Nombre del proveedor

- Rut del proveedor
- Detalle o cargo
- Valor neto
- IVA
- Valor bruto

En cada tupla de la página de ventas, se añade el día, la primera y última boleta de venta emitida, el total de boletas emitidas y el valor neto de ventas.

En el caso de las facturas de venta se añade lo siguiente:

- Día
- Número de factura
- Nombre del comprador
- Rut del comprador
- Detalle o cargo
- Valor neto
- IVA
- Valor bruto

1.3. **Problemas detectados**

Tal cual ha sido mencionado con anterioridad, el principio del problema de la empresa incide en el manejo manual de los productos de ésta, la ausencias de un sistema de inventario, genera de forma general inconsistencia y redundancia en los datos del producto, ya que como es controlado de forma manual, no se puede precisar si este producto ha sido ingresado con anterioridad o que además de ser ingresada mantengan los mismo datos, esto genera un desconocimiento de la cantidad real de ésta.

Considerando el punto anterior, el desconocimiento de una cantidad exacta en los productos provoca un problema al momento de clasificarlos, generando así una dificultad al

administrar éste, como también al momento de generar una búsqueda cuando se está efectuando una venta.

Si bien es cierto la empresa ha podido salir adelante en los últimos años, la ausencia de un control de ventas ha impedido llevar de forma normal un total aproximado de la ventas generadas en el día, también se ocasiona dificultad en los periodos de alta demanda, como es mencionado con anterioridad sobre la inconsistencia en las cantidades del producto, originando así un retraso en la atención del cliente al no saber la cantidad exacta de éste.

La empresa, al momento de generar sus propias manualidades, utiliza sus propios productos, pero éstos no se registran como un descuento del inventario, ocasionando inconsistencia, también se han producido pérdidas de tiempo al momento atender a un cliente habitual, por el hecho de llenar las facturas de venta de forma manual.

Todos los problemas detectados han provocado así una disminución en el crecimiento de la empresa.

1.4. **Descripción del sistema propuesto**

El sistema a presentar consistirá en un sistema de control y ajuste de inventario, el cual al iniciarse desplegará una ventana de inicio de sesión, y una vez aceptado el acceso al sistema se desplegará en pantalla una nueva ventana en donde se verán diversas funcionalidades dependiendo del acceso. Este será utilizado tanto por su administradora como por los vendedores.

1.4.1. **Objetivos**

1.4.1.1. **Objetivo general del sistema**

Las decisiones tomadas por el empresario, ya sean buenas o malas, afectan directamente a la gestión y desarrollo de la organización, independiente de su tamaño, es por ello que se hace necesario contar con información relevante, de tal modo que se puedan tomar

las decisiones correspondientes, por eso se proveerá a la empresa de un software que pueda facilitar las labores de compra, venta y manejo de inventario de productos de la empresa.

1.4.1.2. Objetivos específicos del sistema

Los objetivos a considerar para el sistema son:

- Facilitar el ingreso y egreso de productos al stock, y el acceso a la información de éste y sus actualizaciones, para así evitar pérdidas e inconsistencias.
- Clasificar cada uno de los productos con un código de barra, con el fin de que cada uno de ellos pueda ser ingresado al sistema por medio del escaneo de éste. Esto facilitará también la distinción de éstos al momento de ser inventariados.
- Manejar las ventas, en donde cada producto se ingresará al sistema por medio del lector de código de barras. Una vez escaneado su código se ingresarán los detalles de la venta (cantidad, medida, número de productos a ser vendidos).
- Manejar las compras por factura, que provienen de proveedores, en donde cada producto se ingresará al sistema por medio del lector de código de barras. Una vez escaneado su código se ingresarán los detalles de la compra (medida, número de productos a ser comprados, nombre de proveedor, nombre del producto, tipo de producto, precio de compra).

1.4.2. Beneficios del sistema

Los beneficios que se podrán obtener del sistema son:

- El inventario de productos quedará más consistente y ordenado, además de reducir las redundancias al momento de ingresar productos.
- Se obtendrá como resultado un control de mermas y de ajustes de mercadería en caso de pérdidas y devoluciones.

- Las ventas en períodos de alta demanda se agilizarán y se evitará el descuento manual al stock, ya que el sistema lo descontará de manera automática.
- El registro de todos los productos al sistema, incluyendo ingresos, egresos, mermas y ajustes varios, ofrece una solución al “problema de las manualidades” mencionado anteriormente.
- Agilizar los tiempos de ingreso de los productos comprados a proveedores al sistema de stock, evitando así el registro manual del ingreso de mercadería.

1.4.3. Descripción general del sistema

Éste consistirá en un sistema de administración de inventario, con funcionalidades adicionales de compra y venta de productos. Éste desplegará una ventana de inicio de sesión al ejecutarse, en donde todo usuario deberá ingresar su usuario y contraseña para acceder al menú principal.

El sistema contará con tres niveles de acceso:

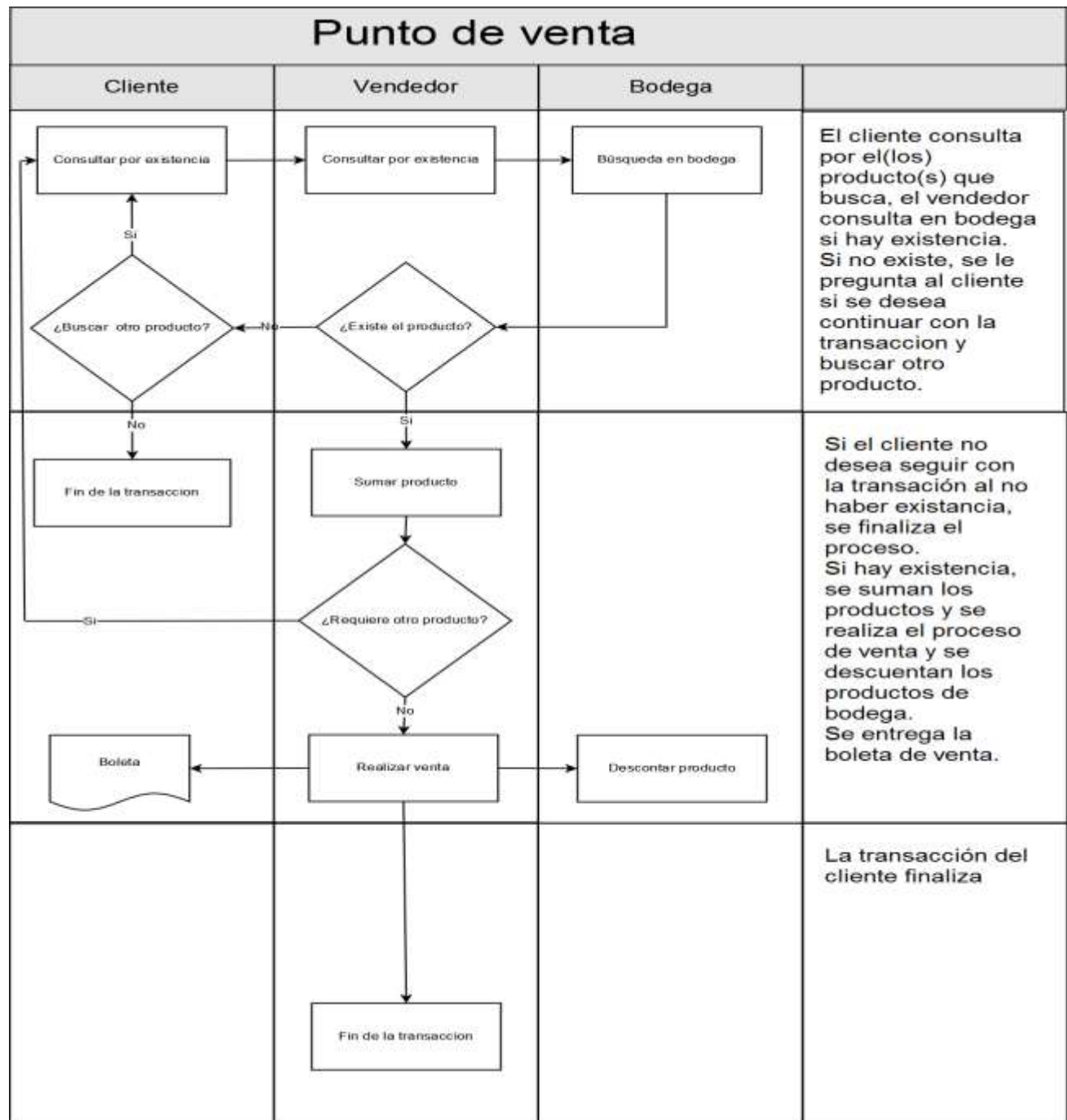
- Nivel de mantención: Este nivel tendrá privilegio de superusuario del sistema, es decir, acceso total a éste y facultades para modificar cualquier dato que estime conveniente. Existirá un único usuario de este tipo y no se podrá modificar ni eliminar del sistema.
- Nivel de administración: A diferencia del anterior éste sólo tendrá acceso al menú de generación de ventas, listados y a todos los mantenedores del sistema, excepto al de usuario.
- Nivel del vendedor: Es el nivel más bajo del sistema y sólo tendrá acceso al sistema de venta y consulta de precios y stock, además de los listados de productos.

A nivel de hardware, el sistema necesitará como periférico adicional de los esenciales, un lector de código de barras.

A nivel de software, el sistema estará conectado a una base de datos guardada en el mismo computador en el cual se trabajará.

1.4.4. Diagrama de flujo administrativo

El diagrama de flujo administrativo que se presenta a continuación muestra el ciclo de trabajo del negocio en el proceso de ventas. Éste se verá desarrollado en el siguiente sistema:



Fuente: Propia

Figura 1-2: DFA Proceso de Venta

1.4.5. Funcionalidades del sistema

El sistema contará con las siguientes funcionalidades:

- Ingreso al sistema (login): El usuario que utilizará el sistema deberá ingresar una clave para poder iniciar sesión con su respectivo nivel de acceso, dependiendo de su nivel de acceso el usuario tendrá entrada a distintas funcionalidades.
- Venta: En esta función se registrarán las transacciones generadas por el vendedor, para que además de registrar la venta en el sistema, se modificará de manera simultánea el stock del inventario.
- Compra: Se registrarán las compras realizadas a cada proveedor, todo ello mediante el registro de las facturas efectuadas por la empresa al sistema.
- Consultar producto: El sistema solicitará el código del producto, una vez ingresado y validado se desplegará en pantalla los detalles de éste, el cual incluirá: nombre del producto, descripción y cantidad en stock.
- Mantenedor de inventario: Función en donde el usuario podrá efectuar acciones básicas respecto a los productos. Entre las funciones que se podrán desarrollar están: agregar, modificar y eliminar.
- Mantenedor de usuario: Función en donde los encargados del sistema podrá efectuar acciones básicas respecto a los usuarios (agregar, modificar, eliminar y consultar).
- Mantenedor de proveedor: Función en donde el usuario podrá efectuar acciones básicas respecto a los proveedores (agregar, modificar, eliminar y consultar).

- Mantenedor de factura: Función en donde el usuario podrá efectuar acciones básicas respecto a la factura a emitir (agregar, modificar, eliminar y consultar).
- Mantenedor de cliente: Función en donde el usuario podrá efectuar acciones básicas respecto a los clientes (agregar, modificar, eliminar y consultar).
- Mantenedor de ajuste: Función en donde el usuario podrá efectuar acciones básicas respecto a los ajustes de inventario (agregar, modificar, eliminar y consultar).
- Mantenedor de catálogo: Función en donde el usuario podrá efectuar acciones básicas respecto a los tipos de productos (agregar, modificar, eliminar y consultar).
- Mantenedor de medida: Función en donde el usuario podrá efectuar acciones básicas respecto a los tipos de medidas (agregar, modificar, eliminar y consultar).
- Mantenedor de boleta: Función en donde el usuario podrá efectuar acciones básicas respecto a las boletas (agregar, modificar, eliminar y consultar).
- Consulta de Stock crítico: Una vez iniciada la sesión se mostrará un mensaje por pantalla preguntando si se desea mostrar el stock crítico, éste desplegará por pantalla los detalles del producto, los cuales incluirá: nombre del producto, cantidad en stock y stock crítico.
- Listado de proveedores ordenado por nombre: El sistema genera un informe, que será mostrado por pantalla, este desplegará a los proveedores ordenado por nombre.
- Listado de productos ordenados por proveedor: El sistema genera un informe, que será mostrado por pantalla, este desplegará a los productos ordenados por proveedor.

- Listado de productos clasificados por tipo: El sistema genera un informe, que será mostrado por pantalla, éste desplegará a los productos clasificados por tipo.
- Listado de los ajustes de mercadería, ordenados por código de producto: El sistema genera un informe, que será mostrado por pantalla, éste desplegará a los ajustes de mercadería, ordenados por código de producto.

1.4.6. Entradas

- Datos de facturas de compra, contiene número de pedido, nombre del proveedor, dirección del proveedor, fono del proveedor, fecha de emisión y detalle de compra.
- Detalle del producto, contiene código del producto, nombre del producto, descripción, stock, precio, tipo de producto, tipo de medida.
- Detalle de la compra, contiene código del producto, cantidad y número de usuarios.
- Datos de venta, contiene número de las ventas, fecha de venta, código de usuario, precio total de la venta y el tipo de documento.
- Datos de proveedores, contiene código del proveedor, nombre del proveedor, dirección del proveedor y fono del proveedor.

1.4.7. Salidas

- Listado de proveedores ordenado por nombre:
 - Rut del proveedor
 - Nombre del proveedor
 - Dirección del proveedor
 - Fono del proveedor
- Listado de productos ordenados por proveedor:
 - Código del producto
 - Nombre de producto

- Stock del producto
 - Precio del producto
 - Nombre del proveedor
- Listado de productos clasificados por catálogo:
 - Código del producto
 - Nombre de producto
 - Stock del producto
 - Precio del producto
 - Nombre del catálogo
- Listado de los ajustes de mercadería, ordenados por código de producto:
 - Fecha del ajuste
 - Código del producto
 - Nombre del producto
 - Tipo de ajuste (merma, ajuste de error, descuento por manualidad)
 - Cantidad ajustada de productos
- Consulta de Stock crítico del producto:
 - Nombre del producto
 - Stock actual del producto
 - Stock mínimo requerido: consiste en el stock que debe tener un producto para realizar un pedido. Si el stock actual es menor que el stock mínimo requerido, el encargado deberá realizar un pedido del producto al respectivo proveedor.
- Inventario de la empresa: será mostrado por pantalla, desplegado mediante método de búsqueda por código de producto:
 - Código del producto
 - Nombre del producto
 - Stock
 - Tipo de producto
 - Medida: se mostrará si se venderá por unidad, por medida de longitud o por medida de peso.

1.4.8. Sistema de codificación

codigo_producto: El código será definido en base a su código de barra. Para los productos que ya poseen uno y éste sea compatible con el mismo, se conservará el original. Si el producto no posee código o éste no es recibido por el lector de código, se confeccionará un código del tipo Code 128 para este producto.

1.4.9. Entidades

La información será manejada en una base de datos relacional que contendrá toda la información de la empresa. El modelo desarrollado se centrará en la funcionalidad de venta, generando así las siguientes tablas:

- Venta: Esta entidad se encargará de registrar la transacción de ventas hechas por el vendedor. Está conformado por: número de las ventas, fecha de venta, código de usuario, precio total de la venta y el tipo de documento.
- Boleta: En esta entidad se encuentran los datos de las boletas emitidas. Está conformado por: código de la boleta y código de venta.
- Detalle_boleta: Esta entidad se encargará de almacenar de manera reiterativa los productos generados en la venta por boleta. Está conformado por: código de la boleta, código del producto, cantidad de productos, precio del producto y total precio por producto.
- Usuario: Esta entidad almacenará los datos de todos los usuarios relacionados con el sistema de venta. Está conformado por: código de usuario, nombre de usuario, nivel de acceso, fono de usuario y clave de acceso.

- Producto: Esta entidad se encargará de guardar en detalle los productos almacenados en bodega. Está conformado por: código del producto, nombre del producto, código del proveedor, código del tipo de medida (masa, longitud y unidad), descripción del producto, stock del producto y precio del producto.
- Medida: Esta entidad se encargará de almacenar los tipos de medida (Masa, metros y unidad). Está conformado por: código del tipo de medida y nombre del tipo de medida.
- Proveedor: Esta entidad mantendrá los datos de la empresa a las cuales se les compra sus productos a vender. Está conformado por: código del proveedor, nombre del proveedor, dirección del proveedor y fono del proveedor.
- Ajuste_inventario: En esta entidad se registrarán los ajustes (positivos o negativos) del producto. Está conformado por: fecha del ajuste, código del producto, tipo de ajuste y cantidad del producto.
- Cliente: En esta entidad se encuentran los datos de los compradores habituales a los cuales la empresa vende sus productos, además de todos los clientes que realicen compras con, al menos, una factura. Está conformado por: rut del cliente, nombre del cliente, fono del cliente, dirección del cliente y nombre de la empresa.
- Factura: En esta entidad se encuentran los datos de las facturas emitidas. Está conformado por: número de venta, folio de la factura, fecha de la factura, Rut del cliente y el total de la factura.
- Detalle_factura: Esta entidad se encargará de almacenar de manera reiterativa los productos generados en la venta por factura. Está conformado por: folio de la factura, número de venta, Rut de cliente, fecha de factura y total de factura.

- Catálogo: Esta entidad almacenara todos los tipos de productos, con la finalidad de ser usado como un filtro de búsqueda. Está conformado por: código de tipo y nombre del tipo.
- Factura_compra: Esta entidad se encargará de registrar la transacción de compras hechas por la empresa. Está conformado por: folio de factura de compra, rut del proveedor, fecha de compra y total de la compra.
- Detalle_factura_compra: Esta entidad se encargará de almacenar de manera iterativa los productos comprados a un proveedor. Está conformado por: folio de factura de compra, código del producto y cantidad del producto a ser comprada.

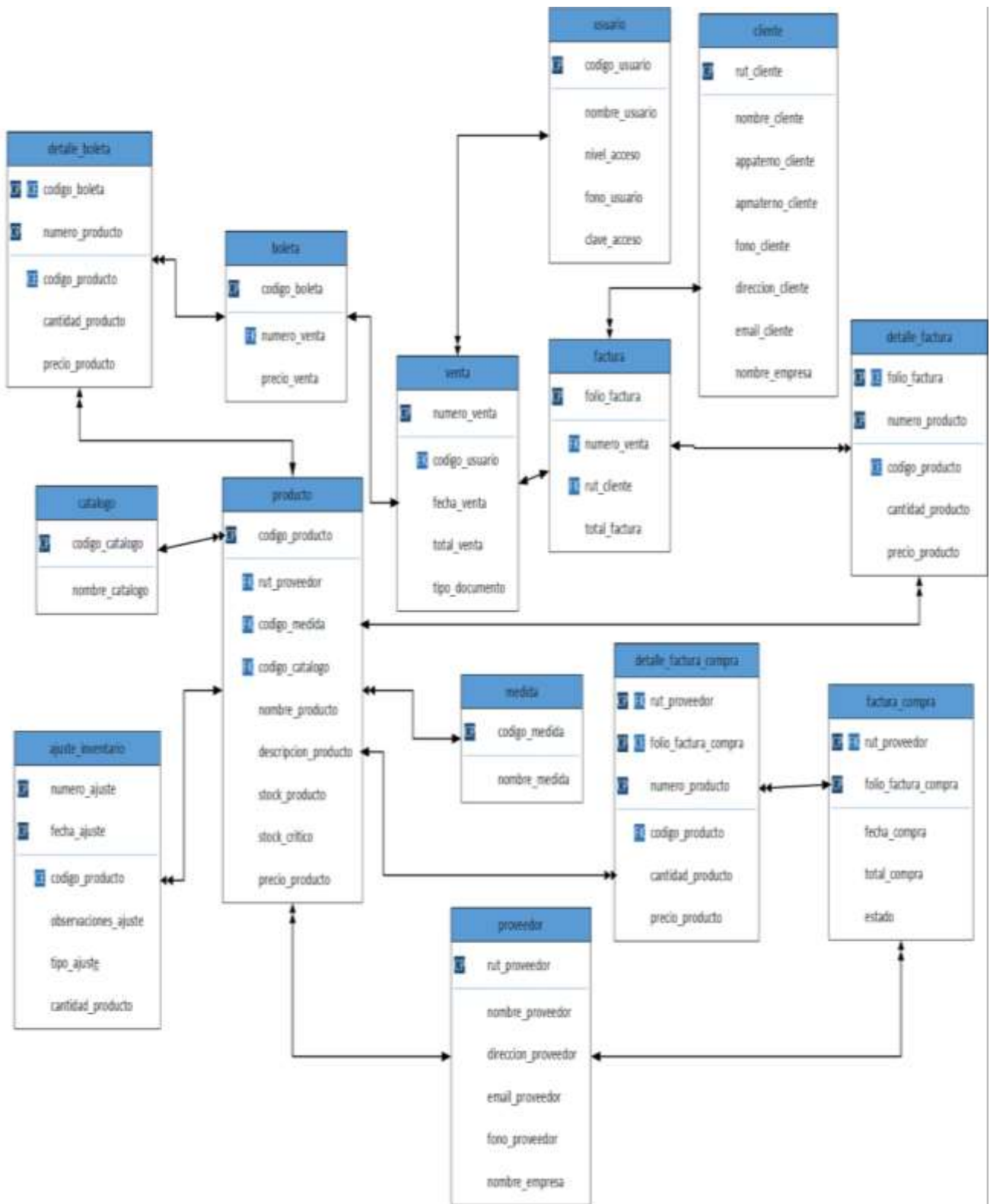
1.4.10. Condicionantes de diseño

Lenguaje de programación: Debido al tipo de sistema que se implementará y que la aplicación es de tipo stand-alone, se utilizará como lenguaje de programación IDE NetBeans, porque permite hacer una interfaz visualmente agradable para el usuario en general, facilitando el uso de ella.

Base de datos: Se trabajará con AppServ, ya que es compatible con IDE NetBeans, este se puede utilizar de manera gratuita ya que es un GNU, además que ésta permitirá almacenar la cantidad de datos que se necesitarán para la empresa.

1.4.11. Modelo relacional de datos

A continuación se presenta el modelo relacional:



Fuente: Propia

Figura 1-2: DFA Proceso de Venta

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2.1. Descripción del hardware y software a utilizar en la empresa

A continuación se detallarán los aspectos de hardware y software que se utilizarán para diseñar y modelar el sistema. Cabe destacar que, como ya se ha mencionado, el proceso actual se desarrolla de manera completamente manual, por lo que los recursos de hardware y software a utilizar nacen del análisis de las herramientas idóneas para el negocio, las cuales buscan principalmente evitar gastos adicionales para el cliente, de tal manera que la puesta en marcha de sistema sea accesible desde el punto de vista económico.

Para la configuración del sistema se detallarán las características más importantes del recurso de hardware a adquirir para la puesta en marcha del sistema. Cabe destacar que tanto la estación de trabajo principal como la base de datos estarán ubicados en el mismo computador, por lo que no será necesaria una conexión a internet para acceder a ella, ésta sólo se utilizará para actualizaciones.

2.1.1. Configuración del sistema

- Procesador: AMD A10-4655M APU, x64
- Disco Duro: Disco duro de 904 Gb
- Memoria: 8 Gb de memoria RAM

a) Periféricos adicionales para la Estación de Trabajo principal

- Lector de código de barras

2.1.2. Software a utilizar

El software a utilizar se ha escogido con el fin de evitar que el cliente incurra en costos adicionales por concepto de licencias.

2.1.2.1. Sistema Operativo

Se ha elegido el sistema operativo Windows 10 en su versión de 64 bits, ya que es más fácil de aprender para el usuario que no tiene mucho conocimiento de informática (caso de los vendedores y administrador del local) y con base en ello también poder recurrir a otras herramientas que el usuario necesite en casos externos del sistema.

2.1.2.2. Software Utilitarios

- AppServ: proveedor de servicios de conexión del cual utilizaremos MySQL.

2.1.2.3. Software de desarrollo

- IDE NetBeans: Para el desarrollo de la aplicación, se utilizará el IDE NetBeans. En este IDE se desarrollará la aplicación Java y se conectará esta con la base de datos de MySQL provista por AppServ.

2.2. Descripción de tablas

2.2.1. Tipos de datos

Tabla 2-1: Tipos de Datos Numéricos

Nombre	Tamaño de almacenamiento	Rango
tinyint	1 byte	-128 a 127
smallint	2 bytes	-32.768 a 32.767
integer	4 bytes	-2.147.483.648 hasta +2.147.483.647

Fuente: Propia

Tabla 2-2: Tipos de Datos Caracteres o Cadenas

Nombre	Tamaño de almacenamiento	Definición
char(n)	4 bytes + n adicionales	Cadena de largo fijo, los valores no ocupados se rellenan con espacios.
varchar(n)	4 bytes + n adicionales	Cadena de largo variable, cuya longitud es n caracteres.

Fuente: Propia

Tabla 2-3: Tipos de Datos Fecha y Tiempo

Nombre	Tamaño de almacenamiento	Rango
Date	4 bytes	4713 A.C. hasta 32767 D.C. Resolución: 1 día.

Fuente: Propia

2.2.2. Descripción de archivos

2.2.2.1. Venta

Nombre de la tabla: Venta

Descripción: Registra datos generales de cada venta realizada.

Clave primaria: numero_venta

Clave foránea: codigo_usuario (Referencia a archivo Usuario)

Longitud del registro: 27

Tabla 2-4: Venta

Nombre	Tipo	Largo	Descripción
numero_venta	integer	7	Número que define la

			transacción
codigo_usuario	tinyint	2	Identifica al usuario
fecha_venta	varchar(10)	10	Fecha en el que se efectúa la venta
total_venta	integer	7	Precio total de la venta
tipo_documento	tinyint	1	Código del tipo de documento. 1: Boleta; 2: Factura

Fuente: Propia

2.2.2.2. Boleta

Nombre de la tabla: Boleta

Descripción: Datos de la boleta de venta

Clave primaria: codigo_boleta

Clave foránea: numero_venta (Referencia a archivo Venta)

Longitud del registro: 25

Tabla 2-5: Boleta

Nombre	Tipo	Largo	Descripción
codigo_boleta	varchar(10)	10	Código de la boleta de venta
numero_venta	integer	7	Código de venta a la cual está vinculada la boleta
precio_venta	integer	8	Precio total de la venta

Fuente: Propia

2.2.2.3. Detalle boleta

Nombre de la tabla: Detalle_boleta

Descripción: Detalle de una boleta de venta.

Clave primaria: codigo_boleta+codigo_producto+numero_producto

Clave foránea: codigo_boleta (Referencia a archivo Boleta)

codigo_producto (Referencia a archivo Producto)

Longitud del registro: 35

Tabla 2-6: Detalle_boleta

Nombre	Tipo	Largo	Descripción
codigo_boleta	varchar(10)	10	Código de la boleta de venta
numero_producto	tinyint	2	Número que representa una transacción de un producto en específico
codigo_producto	varchar(15)	15	Identifica a cada producto
cantidad_producto	smallint	3	Cantidad de productos a vender
precio_producto	integer	5	Precio unitario del producto

Fuente: Propia

2.2.2.4. Usuario

Nombre de la tabla: Usuario

Descripción: Datos del usuario que utilizará el sistema.

Clave primaria: codigo_usuario

Longitud del registro: 139

Tabla 2-7: Usuario

Nombre	Tipo	Largo	Descripción
codigo_usuario	tinyint	2	Código de usuario
nombre_usuario	varchar(20)	20	Nombre del usuario

nivel_acceso	tinyint	1	Nivel de acceso del usuario 0: Mantención 1: Administrador 2: Vendedor
fono_usuario	varchar(16)	16	Teléfono del usuario
clave_acceso	varchar(100)	100	Clave encriptada de acceso del usuario

Fuente: Propia

2.2.2.5. Producto

Nombre de la tabla: Producto

Descripción: Datos del producto

Clave primaria: codigo_producto

Clave foránea: codigo_proveedor (Referencia a archivo Proveedor)

codigo_medida (Referencia a archivo Medida)

codigo_catalogo (Referencia a archivo Catálogo)

Longitud del registro: 178

Tabla 2-8: Producto

Nombre	Tipo	Largo	Descripción
codigo_producto	varchar(15)	15	Código del producto
rut_proveedor	varchar(10)	10	Rut del proveedor
codigo_medida	tinyint	1	Código del tipo de medida del producto
codigo_catalogo	tinyint	2	Indica la categoría en la que se clasificó el producto
nombre_producto	varchar(20)	20	Nombre del producto

descripcion_producto	varchar(120)	120	Descripción del producto
stock_producto	smallint	3	Stock actual del producto
stock_critico	smallint	2	Stock crítico del producto
precio_producto	integer	5	Precio actual del producto

Fuente: Propia

2.2.2.6. Medida

Nombre de la tabla: Medida

Descripción: Datos de las medidas que se utilizarán para vender el producto.

Clave primaria: codigo_medida

Longitud del registro: 16

Tabla 2-9: Medida

Nombre	Tipo	Largo	Descripción
codigo_medida	tinyint	1	Código de la medida
nombre_medida	varchar(15)	15	Nombre de la medida

Fuente: Propia

2.2.2.7. Proveedor

Nombre de la tabla: Proveedor

Descripción: Datos del proveedor

Clave primaria: rut_proveedor

Longitud del registro: 296

Tabla 2-10: Proveedor

Nombre	Tipo	Largo	Descripción
rut_proveedor	varchar(10)	10	Rut del proveedor

nombre_proveedor	varchar(30)	30	Nombre del proveedor
direccion_proveedor	varchar(100)	100	Dirección del proveedor
email_proveedor	varchar(100)	100	Correo electrónico del proveedor
fono_proveedor	varchar(16)	16	Teléfono del proveedor
nombre_empresa	varchar(40)	40	Nombre de la empresa que representa el proveedor

Fuente: Propia

2.2.2.8. Ajuste inventario

Nombre de la tabla: Ajuste_inventario

Descripción: Ajustes realizados al inventario

Clave primaria: fecha_ajuste+codigo_producto+numero_ajuste

Clave foránea: codigo_producto (Referencia a archivo Producto)

Longitud del registro: 154

Tabla 2-11: Ajuste_inventario

Nombre	Tipo	Largo	Descripción
numero_ajuste	smallint	5	Número que representa al ajuste
fecha_ajuste	varchar(10)	10	Fecha en la que se genera el ajuste
codigo_producto	varchar(15)	15	Código que define al producto
observacion_ajuste	varchar(120)	120	Descripción del ajuste,

			este puede ser para devolver o agregar productos
tipo_ajuste	tinyint	1	Define si el ajuste se abona o se carga al stock. 1: Positivo; 2: Negativo.
cantidad_producto	smallint	3	Cantidad del producto a devolver o agregar

Fuente: Propia

2.2.2.9. Cliente

Nombre de la tabla: Cliente

Descripción: Cliente que realiza la compra de un producto

Clave primaria: rut_cliente

Longitud del registro: 336

Tabla 2-12: Cliente

Nombre	Tipo	Largo	Descripción
rut_cliente	varchar(10)	10	Rut del cliente
nombre_cliente	varchar(40)	40	Nombres del cliente
appaterno_cliente	varchar(15)	15	Apellido paterno del cliente
apmaterno_cliente	varchar(15)	15	Apellido materno del cliente
fono_cliente	varchar(16)	16	Teléfono del cliente

direccion_cliente	varchar(100)	100	Dirección personal del cliente
email_cliente	varchar(100)	100	Dirección de correo electrónico del cliente
nombre_empresa	varchar(40)	40	Nombre de la empresa que representa el cliente

Fuente: Propia

2.2.2.10. Factura

Nombre de la tabla: Factura

Descripción: Datos de factura de venta

Clave primaria: folio_factura

Clave foránea: numero_venta (Referencia a archivo Venta)

rut_cliente (Referencia a archivo Cliente)

Longitud del registro: 39

Tabla 2-13: Factura

Nombre	Tipo	Largo	Descripción
folio_factura	varchar(15)	15	Folio de la factura
numero_venta	integer	7	Número de la venta asociada a la factura
rut_cliente	varchar(10)	10	Rut del cliente que recibe la factura
total_factura	integer	7	Total de la factura

Fuente: Propia

2.2.2.11. Detalle factura

Nombre de la tabla: Detalle_factura

Descripción: Detalle de la factura de venta

Clave primaria: folio_factura+numero_producto

Clave foránea: folio_factura (Referencia a archivo Factura)

codigo_producto (Referencia a archivo Producto)

Longitud del registro: 40

Tabla 2-14: Detalle_factura

Nombre	Tipo	Largo	Descripción
folio_factura	varchar(15)	15	Folio de la factura
numero_producto	tinyint	2	Número que representa al producto
codigo_producto	varchar(15)	15	Código del producto a ser vendido
cantidad_producto	smallint	3	Cantidad del producto vendida
precio_producto	integer	5	Precio unitario del producto

Fuente: Propia

2.2.2.12. Catalogo

Nombre de la tabla: Catalogo

Descripción: Datos del catálogo, en donde los productos se clasifican según su tipo

Clave primaria: codigo_catalogo

Longitud del registro: 22

Tabla 2-15: Catalogo

Nombre	Tipo	Largo	Descripción
codigo_catalogo	tinyint	2	Código que identifica el tipo de producto
nombre_catalogo	varchar(20)	20	Nombre del tipo de producto

Fuente: Propia

2.2.2.13. Factura compra

Nombre de la tabla: Factura_compra

Descripción: Datos de una factura de compra

Clave primaria: rut_proveedor+folio_factura_compra

Clave foránea: rut_proveedor (Referencia a archivo Proveedor)

Longitud del registro: 42

Tabla 2-16: Factura_compra

Nombre	Tipo	Largo	Descripción
rut_proveedor	varchar(10)	10	Rut del proveedor
folio_factura_compra	varchar(15)	15	Folio de la factura de compra
fecha_compra	varchar(10)	10	Fecha de emisión de la factura de compra
total_compra	integer	6	Total de la compra realizada por la factura
estado	tinyint	1	Estado de pago de la factura de compra: 1:Pagado

			2:Impaga
--	--	--	----------

Fuente: Propia

2.2.2.14. Detalle factura compra

Nombre de la tabla: Detalle_factura_compra

Descripción: Detalle de una factura de compra

Clave primaria: rut_proveedor+folio_factura_compra+numero_producto

Claves foráneas: rut_proveedor (Referencia a archivo Proveedor)

rut_proveedor+folio_factura_compra

(Referencia a archivo Factura_compra)

codigo_producto (Referencia a archivo Producto)

Longitud del registro: 48

Tabla 2-17: Detalle_factura_compra

Nombre	Tipo	Largo	Descripción
rut_proveedor	varchar(10)	10	Rut del proveedor
folio_factura_compra	varchar(15)	15	Número que identifica la factura de compra
numero_producto	tinyint	2	Número que representa al producto
codigo_producto	varchar(15)	15	Código que identifica al producto
cantidad_producto	smallint	2	Cantidad del producto a comprar
precio_producto	integer	4	Precio de compra del producto

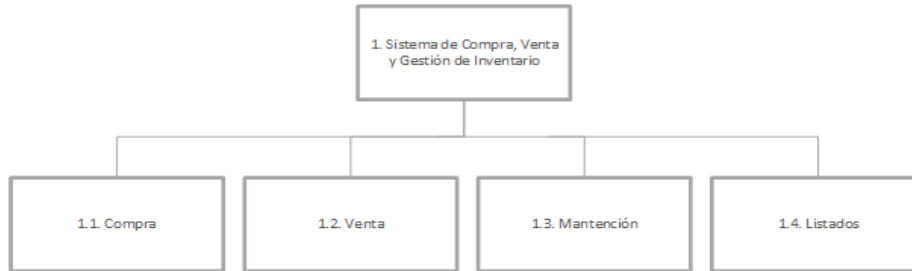
Fuente: Propia

CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCIÓN DE PROGRAMAS

3.1. Diagrama modular

A continuación se muestran imágenes con la estructura modular del sistema:



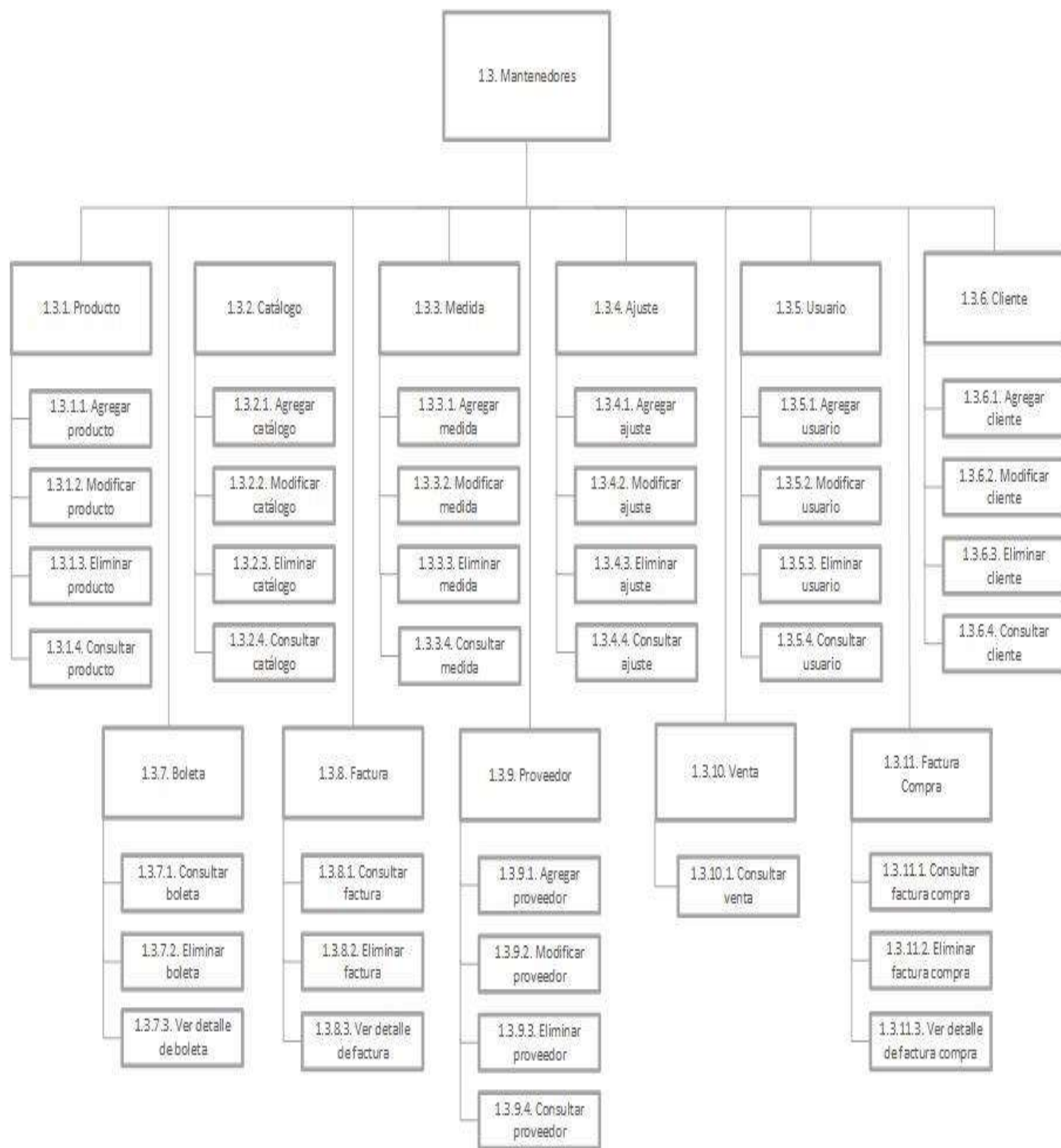
Fuente: Propia

Figura 3-1: Diagrama Modular General



Fuente: Propia

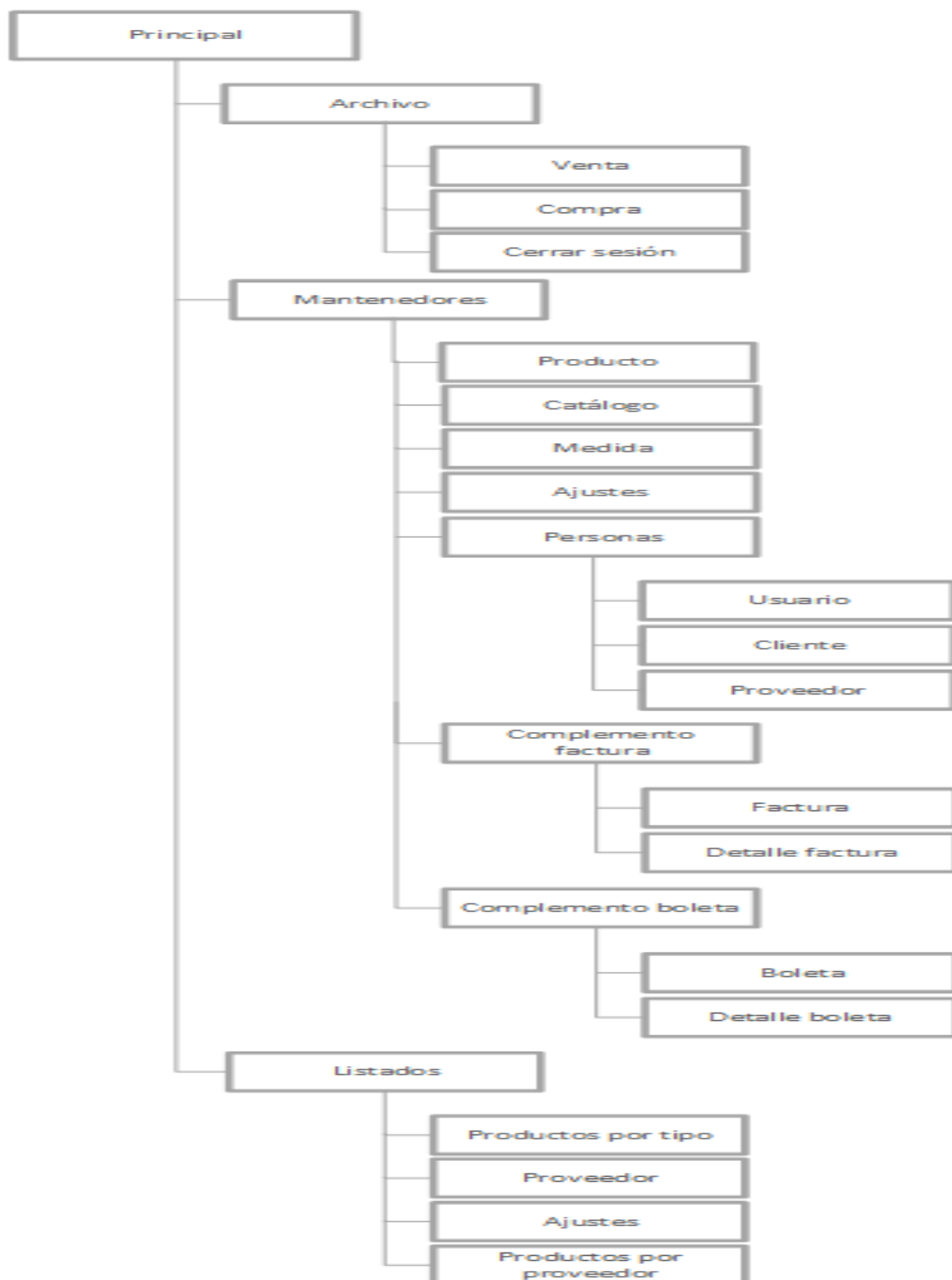
Figura 3-2: Diagrama Modular de Listados



Fuente: Propia

Figura 3-3: Diagrama Modular de Mantenedores

3.2. Diagrama de menús



Fuente: Propia

Figura 3-4: Diagrama Estructura de Menú del Sistema

3.3. Totalidad de programas

Tabla 3-1: Listado de Archivos con Descripción

Nombre Lógico	Descripción
Principal	Pantalla principal del sistema, se carga al iniciar el programa. Contiene el menú con las funciones del sistema.
Login*	Inicio de sesión, valida el ingreso de un usuario al sistema.
Compra*	Registra compra a proveedor.
Venta*	Registra ventas de productos y tipo de venta.
List_ajusprod	Lista de los ajuste de merma ordenados por producto, generada para la toma de decisiones.
List_prodprov	Lista de productos ordenados por proveedor, generada para la toma de decisiones.
List_prodtipo	Lista de productos ordenados por catálogo, generada para la toma de decisiones.
List_prov	Lista de proveedores ordenada por nombre, generada para la toma de decisiones.
Man_ajuste	Registra los ajustes en el stock de los productos, y especifica el detalle de la cantidad ajustada.
Man_boleta	Mantiene la entidad Boleta en la base de datos
Man_detalleboleta	Visualiza los datos del detalle de la Boleta
Man_catalogo	Mantiene la entidad Catalogo en la base de datos
Man_cliente*	Mantiene la entidad Cliente en la base de datos
Man_factura	Mantiene la entidad Factura en la base de datos
Man_detallefactura	Visualiza los datos del detalle de la Factura
Man_facturacompra	Mantiene la entidad Factura de Compra en la base de datos
Man_detallefacturacompra	Visualiza los datos del detalle de la Factura de Compra
Man_medida	Mantiene la entidad Medida en la base de datos

Man_producto*	Mantiene la entidad Producto en la base de datos
Man_proveedor*	Mantiene la entidad Proveedor en la base de datos
Man_usuario*	Mantiene la entidad Usuario en la base de datos
Man_venta	Mantiene la entidad Venta en la base de datos

Fuente: Propia

Los programas indicados con '*' son los que se describirán en detalle a continuación.

3.4. Descripción detallada de programas

3.4.1. Inicio de sesión

- Nombre: Login.
- Objetivo: Validar si el nombre y la contraseña ingresados corresponden a algún usuario registrado en el sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-5 Diagrama de Bloques Inicio de Sesión

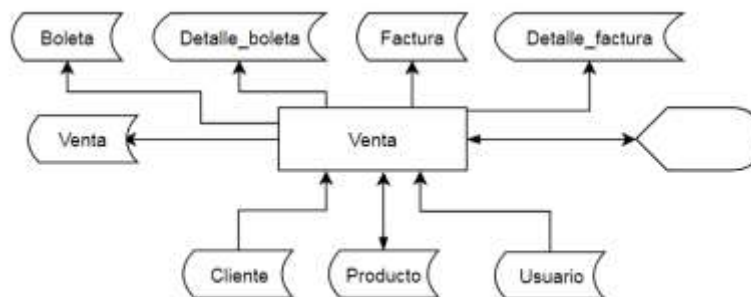
- Reglas de proceso: Se presenta un formulario que solicita el nombre de usuario y contraseña. El usuario ingresa los datos solicitados y al presionar el botón Ingresar, el sistema validará que el usuario ingresado exista en la tabla de usuarios y además que la contraseña corresponda a la de éste. Si los datos son correctos, entonces el sistema despliega el formulario principal. Si los datos no son correctos, entonces el sistema despliega una ventana emergente con un mensaje de error que detalla el problema que ocurrió al ejecutar la acción.
- Pantalla:

Fuente: Propia

Figura 3-6: Formulario Inicio de Sesión

3.4.2. Gestión de venta

- Nombre: Venta
- Objetivo: Registrar una venta en la base de datos del sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-7: Diagrama de Bloques Gestión de Venta

- Reglas de proceso: El usuario correctamente autenticado accede desde el formulario principal al presionar en el menú la pestaña de venta. Se comunica con las tablas Venta, Boleta, Detalle_boleta, Factura, Detalle_factura, Cliente y Producto. El usuario selecciona un vendedor (obtenido desde la tabla Usuario) y una vez seleccionado aparecerá al lado del cuadro de lista de vendedor el nombre de éste. Luego, se selecciona el tipo de documento (boleta o factura); si selecciona una factura, solicitará

además nombre del cliente, sino, no solicitará este dato. El número de documento y la fecha aparecerán de manera automática. Luego, se buscarán los productos con las herramientas que están dentro del frame “Producto”. Se colocará el código o parte de éste en el campo de texto “Buscar por código:”, lo mismo aplica para el campo “Buscar por nombre”, entonces los resultados aparecerán automáticamente en la caja de listado “Búsqueda”. De esa lista, se selecciona el producto deseado haciéndole clic en ésta y definiendo la cantidad en el frame de producto. Una vez definidos los datos, se presiona el botón verde con el visto para incluirlo en el listado inferior. Una vez agregados, si los datos solicitados en pantalla están seleccionados, entonces se podrá finalizar la venta haciendo clic en el botón “Finalizar”. Si se desea limpiar la lista inferior, se debe hacer clic en el botón “Limpiar”. Finalmente, para anular la venta, se presiona el botón “Volver”.

- Pantalla:

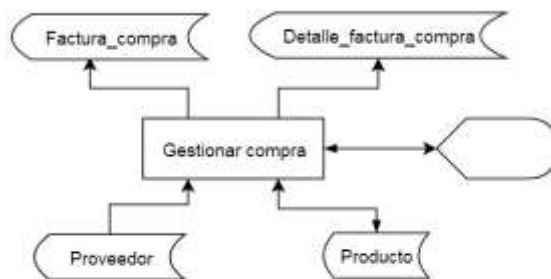
Fuente: Propia

Figura 3-8: Formulario Gestión de Venta

3.4.3. Gestión de compra

- Nombre: Compra
- Objetivo: Añadir datos de una compra a la base de datos del sistema.

- Diagrama de bloques:



Fuente: Propia

Figura 3-9: Diagrama de Bloques Gestión de Compra

- Reglas de proceso: El usuario correctamente autenticado accede desde el formulario principal al presionar en el menú la pestaña de compra. Se comunica con las tablas Factura_compra, Detalle_factura_compra, Proveedor y Producto. El usuario ingresa el número de folio, selecciona un proveedor (obtenido de la tabla Proveedor), una fecha (que por defecto se asignará la fecha actual) y el estado de la factura (que puede ser “Pagada” o “Pendiente”). Luego, el usuario del sistema buscará los productos con las herramientas que están dentro del marco “Producto”. Se colocará el código o parte de éste en el campo de texto “Buscar por código”, lo mismo aplica para el campo “Buscar por nombre”, entonces los resultados aparecerán automáticamente en la caja de listado “Búsqueda”. De esa lista, se selecciona el producto deseado al hacer clic en ésta y definiendo la cantidad y precio unitario en el marco de producto. Una vez agregados, se presiona el botón verde con el visto para agregarlo al listado inferior. Una vez añadidos todos los productos necesarios para registrar la compra, se realiza el registro en la base de datos haciendo clic en el botón “Finalizar”. Si se desea limpiar la tabla inferior, se debe hacer clic en el botón “Limpiar”. Finalmente, para anular todo lo realizado y volver a la pantalla principal, se debe hacer clic en el botón “Volver”.

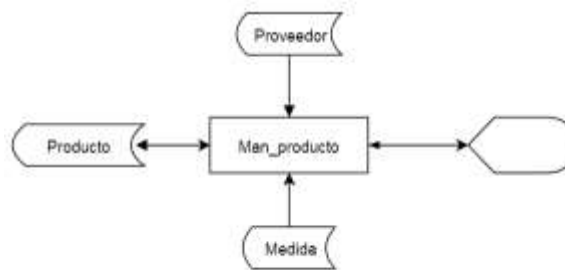
- Pantalla:

Fuente: Propia

Figura 3-10: Formulario Gestión de Compra

3.4.4. Mantenedor de Producto

- Nombre: Man_producto
- Objetivo: Actualizar datos de la tabla Producto en la base de datos del sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-11: Diagrama de Bloques Mantenedor de Producto

- Reglas de proceso: Para agregar un producto se ingresa el código, el nombre del producto, una descripción y se selecciona el proveedor, el catálogo, la medida, el precio, el stock y el stock crítico. Al presionar el botón de Agregar, el sistema validará que se hayan ingresado y seleccionado todos los datos y que éstos sean consistentes. Si se cumplen estos requisitos, el producto será agregado a la base de datos. Para modificar o

eliminar, se selecciona una tupla de la tabla y se presiona el botón con la acción deseada para modificar o eliminar un objeto de la tabla. El sistema validará que todos los datos estén ingresados y seleccionados en los campos de definición de parámetros, y sólo se realizará la acción si pasa la validación.

- Pantalla:

Fuente: Propia

Figura 3-12: Formulario Mantenedor de Producto

3.4.5. Mantenedor de Proveedor

- Nombre: Man_proveedor
- Objetivo: Actualizar datos de la tabla Proveedor en la base de datos del sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-13: Diagrama de Bloques Mantenedor de Proveedor

- Reglas de proceso: Para agregar un proveedor se ingresa el RUT (número y dígito verificador por separado), el nombre del producto, el teléfono, correo electrónico, dirección y el nombre de la empresa. Al presionar el botón de Agregar, el sistema

validará que se hayan ingresado todos los datos obligatorios (sólo el correo electrónico y el nombre de la empresa son optativos), que el dígito verificador del RUT sea consistente con la parte numérica (validado por el módulo 11) y que el correo electrónico incluya un arroba (@) y un punto, en caso de ser ingresado. Si se cumplen estos requisitos, el proveedor será agregado a la base de datos. Para modificar o eliminar, se selecciona una tupla de la tabla o se ingresan los datos manualmente, luego se presiona el botón con la acción deseada para modificar o eliminar un objeto de la tabla. El sistema validará que todos los datos estén ingresados y seleccionados en los campos de definición de parámetros, y sólo se realizará la acción si pasa la validación.

- Pantalla:

PROVEEDOR

Rut:

-

Nombre:

Teléfono:

Email:

Dirección:

Nombre Empresa:

Código	Nombre	Dirección	Teléfono	Email	Nombre Empresa
10469018-1	Jessica				
1048108-8	Gertrudis	Calera	11111111		
13115168-3	Francisco				
13732718-K	Mario	Quilpué	12341234		
15653148-0	Ignacio				
15655064-7	Marta				
16461877-3	Luis	Quilpué			
16820314-4	Sebastian				
17101449-2	Paulino				
19176974-0	Owen				
9376474-9	Armando				

Mostrar Búsqueda:

Código	Nombre	Dirección	Teléfono	Email	Nombre Empresa

Agregar

Modificar

Eliminar

Consultar

Cancelar

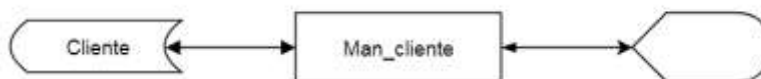
Volver

Fuente: Propia

Figura 3-14: Formulario Mantenedor de Proveedor

3.4.6. Mantenedor de Cliente

- Nombre: Man_cliente
- Objetivo: Actualizar datos de la tabla Cliente en la base de datos del sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-15: Diagrama de Bloques Mantenedor de Cliente

- Reglas de proceso: Para agregar un cliente se ingresa el RUT (número y dígito verificador por separado), el nombre del cliente, apellido paterno, apellido materno, la empresa que representa, el teléfono personal, dirección y correo electrónico. Al presionar el botón de Agregar, el sistema validará que se hayan ingresado todos los datos obligatorios (sólo el correo electrónico es optativo), que el dígito verificador del RUT sea consistente con la parte numérica (validado por el módulo 11) y que el correo electrónico incluya un arroba (@) y un punto, en caso de ser ingresado. Si se cumplen estos requisitos, el cliente será agregado a la base de datos. Para modificar o eliminar, se selecciona una tupla de la tabla o se ingresan los datos manualmente, luego se presiona el botón con la acción deseada para modificar o eliminar un objeto de la tabla. El sistema validará que todos los datos estén ingresados, seleccionados y validados.
- Pantalla:

La interfaz de usuario para el mantenimiento de clientes se divide en varias secciones:

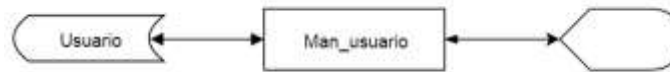
- Encabezado:** Un título "CLIENTE" en la parte superior.
- Formulario de Datos:** Campos de entrada para "Rut*", "Nombre*", "Apellidos*", "Empresa*", "Fono*", "Dirección*" y "Correo electrónico".
- Tabla de Datos:** Una tabla con columnas: Rut, Nombre, Ap. Paterno, Ap. Materno, Fono, Dirección, Correo, Empresa.
- Sección de Búsqueda:** Un botón "Mostrar Búsqueda" y una segunda tabla con las mismas columnas que la anterior.
- Botones de Acción:** "Agregar", "Modificar", "Eliminar", "Consultar", "Cancelar" y "Volver".

Fuente: Propia

Figura 3-16: Formulario Mantenedor de Cliente

3.4.7. Mantenedor de Usuario

- Nombre: Man_usuario
- Objetivo: Actualizar datos de la tabla Usuario en la base de datos del sistema.
- Diagrama de bloque:



Fuente: Propia

Figura 3-17: Diagrama de Bloques Mantenedor de Usuario

- Reglas de proceso: Para agregar un usuario se ingresa el código del usuario, el nombre del usuario, la clave que utilizará para ingresar al sistema, su teléfono y su nivel. Al presionar el botón de Agregar, el sistema validará que se hayan ingresado todos los datos y que el usuario no exista previamente. Si se cumplen estos requisitos, el usuario será agregado a la base de datos con la clave encriptada en el sistema. Para modificar o eliminar, se selecciona una tupla de la tabla o se ingresan los datos manualmente, luego se presiona el botón con la acción deseada para modificar o eliminar un objeto de la tabla. El sistema validará que todos los datos estén ingresados y seleccionados en los campos de definición de parámetros, y sólo se realizará la acción si pasa la validación.
- Pantalla:

El formulario 'USUARIO' tiene los siguientes campos de entrada:

- Código:
- Clave:
- Nivel:
- Nombre:
- Fono:

Debajo de los campos, hay una tabla con los datos de los usuarios:

Código	Nombre	Nivel	Teléfono	Clave
1	admin	Alto	11112222	4144534954

Debajo de la tabla, hay un botón 'Mostrar Búsqueda' y una tabla para los resultados de la búsqueda:

Código	Nombre	Nivel	Teléfono	Clave
--------	--------	-------	----------	-------

En la parte inferior del formulario, hay una barra de botones con las siguientes opciones: Agregar, Modificar, Eliminar, Consultar, Cancelar y Volver.

Fuente: Propia

Figura 3-18: Formulario Mantenedor de Usuario

CONCLUSIONES

Antes del análisis, se debería definir bien los requisitos del sistema. En el análisis el personal a cargo del sistema descubre que no es fácil enfrentarse a los requerimientos del cliente, y éstos deben ser revisados constantemente para obtener un sistema que sea funcional y a la vez cumpla con los requisitos más alcanzables para el cliente.

Concebir un sistema desde el punto de vista analítico es una tarea difícil de visualizar al comienzo, pero alcanzable a medida que se trabaja en ello.

Una vez finalizado el análisis comienza el diseño del sistema. En el diseño de éste, es frecuente que ocurran inconvenientes como errores en el análisis o cambios de requerimientos, entonces esto significa retroceder a la etapa de análisis. Una vez solucionado esto, en el diseño se define la concepción de la base de datos y los archivos de ésta, los cuales serán cruciales para almacenar la información del programa de manera ordenada.

En cuanto a la etapa de la construcción, acá se crean las interfaces de usuario, las cuales serán la parte más amigable del programa, y además el código de éstas para que cumplan con las funciones solicitadas por el cliente. Esta última parte es mucho más difícil que la primera.

Finalmente, con base en lo mencionado en esta conclusión, se puede concluir que no es fácil construir un sistema de información, ya que antes de programar se requiere cumplir unas agotadoras etapas de definición de requerimientos, análisis y diseño de éste, las cuales, si no se realizan no se podría crear un sistema consistente con los requerimientos solicitados por el usuario con tanta facilidad.

Además, sin las herramientas como los frameworks, provee una forma estándar para construir y desplegar aplicaciones, esta tarea sería más complicada aún.

BIBLIOGRAFÍA

Lenguaje de programación JAVA:

<https://docs.google.com/file/d/0Byy7aUI9u4fBRnJwc1U5VkdnaIk/edit>

Lenguaje JAVA Avanzado:

<http://www.jtech.ua.es/j2ee/publico/lja-2012-13/wholesite.pdf>

LINKOGRAFÍA

Sitio de descarga de NetBeans:

<https://netbeans.org/downloads/>

Sitio de descarga de biblioteca de jCalendar:

<http://www.java2s.com/Code/Jar/j/Downloadjcalendar14jar.htm>

Sitio de descarga de AppServ:

<https://www.appserv.org/download/>

Tutorial de jCalendar:

<http://datojava.blogspot.cl/2015/11/jcalendarJavaSwingEjemploTutorial.html>

Tutorial de MySql:

<https://www.tutorialspoint.com/mysql/>

ANEXOS

ANEXO 1: CÓDIGO FUENTE DE LA CLASE CONECTAR.

```
//-----
// Definición de las librerías propias de Java
//-----

package Clases;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

//-----
// Establecer los valores para generar la conexión
//-----

public class Conectar {
    Connection conecta;
    String BD = "arca";
    String usuario = "root";

    // Contraseña del equipo a ocupar
    String password = "*****";

//-----
// Atributos para establecer las conexión con la base de datos
//-----

    public Conectar() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {}

        try {
            conecta = (Connection) DriverManager.getConnection("jdbc:mysql://" + "localhost/" + BD,
                usuario, password);
        } catch (SQLException ex) {
            System.out.println(""+ex);
        }
    }

    public Connection getconnect(){
```

```

        return conecta;
    }

//-----
// Código para cerrar las conexión con la base de datos
//-----

    public void cerrar(){
        if (conecta != null){
            try {
                conecta.close();
            } catch (SQLException ex) {
                System.out.println("Ya está cerrado"+ex);
            }
        }
    }
}

```

ANEXO 2: CÓDIGO FUENTE DE PROGRAMA VENTA.

```

//-----
// Definición de las librerías propias y librerías propias de Java
//-----

package Proyecto.Archivo;
import Clases.Conectar;
import Clases.Sql;
import Proyecto.Mantenedores.Man_ajuste;
import Proyecto.Mantenedores.Man_cliente;
import Proyecto.Mantenedores.Man_producto;
import java.awt.Image;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;

```

```

import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

//-----
// Generación de instancias para establecer las conexión con la base de datos e inicio
// de componentes del programa
//-----

public class Venta extends javax.swing.JDialog {

    //Conectarme a la base de datos.
    Conectar connection = new Conectar();

    //Tener un elemento cn con el cual nos permite hacer la sentencias.
    Connection cn = connection.getconnect();

    public Venta(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();

        int ancho = 1133;
        int alto = 629;
        this.setBounds(0, 0, ancho, alto);
        this.setTitle("EL ARCA - Venta");
        this.setLocationRelativeTo(null);
        this.DC_fecha.setDate(new Date());
        this.DC_fecha.setSelectableDateRange(this.DC_fecha.getDate(), this.DC_fecha.getDate());

        ImageIcon imagen = new ImageIcon("src/Imagenes/fondo_cv.jpg");
        Icon icono = new ImageIcon(imagen.getImage().getScaledInstance(ancho, alto,
            Image.SCALE_DEFAULT));
        Fondo_venta.setIcon(icono);
        this.repaint();

        TBL_normal.getTableHeader().setReorderingAllowed(false);
        TBL_normal.setRowSelectionAllowed(true);
        TBL_normal.setColumnSelectionAllowed(false);

        TBL_busqueda.getTableHeader().setReorderingAllowed(false);
        TBL_busqueda.setRowSelectionAllowed(true);
        TBL_busqueda.setColumnSelectionAllowed(false);
    }
}

```

```

    llenarComboUser();
    llenarComboCliente();
}

```

```

//-----
// Definición de eventos y validaciones del programa
//-----

```

```

private void BTN_volverActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón Volver para cerrar la ventana modal.

```

```

    this.setVisible(false);
}

```

```

private void BTN_cancelarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón Cancelar representado con una X, para formatear los campos
    //en la sección de producto.

```

```

    limpiarTabla("B");
    SP_cantidad.setValue(1);
    TF_busCod.setText("");
    TF_busNom.setText("");
}

```

```

private void RB_boletaActionPerformed(java.awt.event.ActionEvent evt) {
    //Al seleccionar la opción boleta del radio bottom despliega los componentes
    //correspondientes a la boleta.

```

```

    LBL_tipocodigo.setText("Número de boleta: ");
    CB_codigoCli.hide();
    BTN_actualizar.hide();
    jLabel4.hide();
    TF_documento.setText("");
}

```

```

private void RB_facturaActionPerformed(java.awt.event.ActionEvent evt) {
    //Al seleccionar la opción factura del radio bottom despliega los componentes
    //correspondientes a la factura.

```

```

    LBL_tipocodigo.setText("Folio de factura: ");
    CB_codigoCli.show();
    BTN_actualizar.show();
    jLabel4.show();

```

```

    TF_documento.setText("");
}

private void BTN_limpiarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón limpiar para resetear de manera general los
    //valores de los componentes.

    CB_codigoVen.setSelectedIndex(0);
    CB_codigoCli.setSelectedIndex(0);
    SP_cantidad.setValue(1);
    TF_busCod.setText("");
    TF_busNom.setText("");
    TF_documento.setText("");
    DC_fecha.setDate(new Date());
    LBL_total.setText("$0");
    limpiarTabla("N");
    limpiarTabla("B");
}

private void BTN_productoActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón producto, para activar el programa modal
    //"Mantenderor de Producto".

    Man_producto frame = new Man_producto(this, true);
    frame.setVisible(true);
}

private void BTN_ajusteActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón ajuste, para activar el programa modal
    //"Mantenedor de Ajuste".

    Man_ajuste frame = new Man_ajuste(this, true);
    frame.setVisible(true);
}

private void CB_codigoVenActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el combobox del vendedor, al seleccionar una de las opciones
    //trae a través de la base de datos, el nombre del vendedor que gestiona
    //la venta.

    int codigo = CB_codigoVen.getSelectedIndex();
    String nom;
    String query = "select * from usuario WHERE "

```

```

        + "codigo_usuario = '" + codigo + "'";

if (codigo != 0) {
    try {

        Statement st = cn.createStatement();
        ResultSet lista = st.executeQuery(query);

        if (lista.next() == true) {

            nom = lista.getString("nombre_usuario");

            LBL_nombreVent.setText(nom);

        }

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error al encontrar Vendedor", "Error",
            JOptionPane.ERROR_MESSAGE);
    }
} else {
    LBL_nombreVent.setText("");
}
}

private void BTN_aceptarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón Aceptar representado con un símbolo de aprobación,
    //para efectuar los cálculos de producto por cantidad y así ser ingresados
    //a la tabla de venta, usados en la sección de producto.

    TF_busCod.requestFocus();

    int selec = TBL_busqueda.getSelectedRow();

    if (selec == -1) {
        JOptionPane.showMessageDialog(null, "Seleccione un producto", "Error",
            JOptionPane.ERROR_MESSAGE);
    } else {
        DefaultTableModel tm = (DefaultTableModel) TBL_busqueda.getModel();

        String cod = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 0));
        String nom = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 1));
        String pre = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 2));
    }
}

```

```

        int precio = Integer.parseInt(pre);
        int cantidad = Integer.valueOf(SP_cantidad.getValue().toString());
        int total = precio * cantidad;
        actualizarTotal(total);
        DefaultTableModel modelo = (DefaultTableModel) TBL_normal.getModel();
        modelo.addRow(new Object[]{cantidad, cod, nom, precio, total});
    }
}

private void TBL_normalMouseClicked(java.awt.event.MouseEvent evt) {
    //Al generar un doble click en la tabla venta, se procede a eliminar el registro
    //señalado por el doble click.

    if (evt.getClickCount() == 2) {
        DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();
        String pre = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 4));
        int precio = Integer.parseInt(pre);

        String cifra = LBL_total.getText().substring(1, LBL_total.getText().length());
        int auxCifra = Integer.valueOf(cifra);
        int aux = auxCifra - precio;

        String STtotal = String.valueOf(aux);
        String format = "$".concat(STtotal);
        this.LBL_total.setText(format);
        tm.removeRow(TBL_normal.getSelectedRow());
    }
}

private void BTN_finalizarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón finalizar, para efectuar la venta, y guardar los datos de la misma,
    //en sus respectivas tablas en la base de datos.

    String vacioDocu = TF_documento.getText();

    //Datos para la venta
    int vend = CB_codigoVen.getSelectedIndex();

    int anno = DC_fecha.getCalendar().get(Calendar.YEAR);
    int mes = DC_fecha.getCalendar().get(Calendar.MONTH) + 1;
    int dia = DC_fecha.getCalendar().get(Calendar.DAY_OF_MONTH);
    String fecha = dia + "-" + mes + "-" + anno;

```

```

String cifra = LBL_total.getText().substring(1, LBL_total.getText().length());
int total = Integer.valueOf(cifra);

int tipo = 0;
if (RB_boleta.isSelected()) {
    tipo = 1;
} else if (RB_factura.isSelected()) {
    tipo = 2;
}

int selec = TBL_normal.getRowCount();

if (CB_codigoVen.getSelectedIndex() == 0) {
    JOptionPane.showMessageDialog(null, "Seleccione a un vendedor", "Error",
        JOptionPane.ERROR_MESSAGE);
} else if ((RB_factura.isSelected() == false) && (RB_boleta.isSelected() == false)) {
    JOptionPane.showMessageDialog(null, "Seleccione a una opcion de documento", "Error",
        JOptionPane.ERROR_MESSAGE);
} else if ((CB_codigoCli.getSelectedIndex() == 0) && (RB_factura.isSelected() == true)) {
    JOptionPane.showMessageDialog(null, "Seleccione a un cliente", "Error",
        JOptionPane.ERROR_MESSAGE);
} else if (vacioDocu.equals("")) {
    JOptionPane.showMessageDialog(null, "Ingrese un numero de boleta o un folio de
        factura", "Error", JOptionPane.ERROR_MESSAGE);
} else if (selec == 0) {
    JOptionPane.showMessageDialog(null, "La lista de venta esta vacia", "Error",
        JOptionPane.ERROR_MESSAGE);
} else {
    try {
        Statement st = (Statement) cn.createStatement();

        //Se ingresa la venta
        int numero = numCorVen();

        st.executeUpdate("INSERT INTO venta
(número_venta,código_usuario,fecha_venta,total_venta,tipo_documento) VALUES("
            + numero + ","
            + vend + ","
            + fecha + ","
            + total + ","
            + tipo
            + ")");
    }
}

```

```

if (tipo == 1) {
    //Datos para la boleta
    String numBoleta = TF_documento.getText();

    //Se ingresa la boleta
    st.executeUpdate("INSERT INTO boleta (codigo_boleta,numero_venta,precio_venta)
        VALUES('"
            + numBoleta + ","
            + numero + ","
            + total
            + "')");

    //Contador de filas de la tabla
    int cant = TBL_normal.getRowCount();

    //Actualizar producto
    int cont = 0;
    int invalido = 0;
    while ((cont < cant) || (invalido == 0)) {
        //Datos detalle boleta
        DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();

        String cantidad = String.valueOf(tm.getValueAt(cont, 0));
        int auxCant = Integer.parseInt(cantidad);

        String codigo = String.valueOf(tm.getValueAt(cont, 1));

        invalido = actualizarProducto(codigo, auxCant);

        cont = cont + 1;
    }

    //Insertar en el detalle de la boleta
    if (invalido == 1) {
        int cont1 = 0;
        while (cont1 < cant) {
            //Datos detalle boleta
            DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();
            String cantidad = String.valueOf(tm.getValueAt(cont1, 0));
            int auxCant = Integer.parseInt(cantidad);

            String codigo = String.valueOf(tm.getValueAt(cont1, 1));

```

```

String pre = String.valueOf(tm.getValueAt(cont1, 3));
int auxPre = Integer.parseInt(pre);

int cor_boleta = numCorBol(numBoleta);

//Se ingresa el detalle boleta
st.executeUpdate("INSERT INTO detalle_boleta
(numero_producto,codigo_boleta,codigo_producto,cantidad_producto,"
+ "precio_producto) VALUES("
+ cor_boleta + ","
+ numBoleta + ","
+ codigo + ","
+ auxCant + ","
+ auxPre
+ ")");
cont1 = cont1 + 1;
}
}
} else if (tipo == 2) {

//Datos para la factura
String numFactura = TF_documento.getText();

ResultSet buscarRutProv = st.executeQuery("SELECT rut_cliente FROM cliente WHERE
nombre_cliente = '" + CB_codigoCli.getSelectedItem() + "'");
buscarRutProv.next();
String cliente = buscarRutProv.getString("rut_cliente");

//Se ingresa la factura
st.executeUpdate("INSERT INTO factura
(folio_factura,numero_venta,rut_cliente,total_factura) VALUES("
+ numFactura + ","
+ numero + ","
+ cliente + ","
+ total
+ ")");

//Contador de filas de la tabla
int cant = TBL_normal.getRowCount();

//Actualizar producto
int cont = 0;
int invalido = 0;

```

```

while ((cont < cant) || (invalido == 0)) {
    //Datos detalle boleta
    DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();

    String cantidad = String.valueOf(tm.getValueAt(cont, 0));
    int auxCant = Integer.parseInt(cantidad);

    String codigo = String.valueOf(tm.getValueAt(cont, 1));

    invalido = actualizarProducto(codigo, auxCant);

    cont = cont + 1;
}

//Insertar en el detalle de la factura
if (invalido == 1) {
    int cont1 = 0;
    while (cont1 < cant) {
        //Datos detalle boleta
        DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();
        String cantidad = String.valueOf(tm.getValueAt(cont1, 0));
        int auxCant = Integer.parseInt(cantidad);

        String codigo = String.valueOf(tm.getValueAt(cont1, 1));

        String pre = String.valueOf(tm.getValueAt(cont1, 3));
        int auxPre = Integer.parseInt(pre);

        int cor_factura = numCorFact(numFactura);

        //Se ingresa el detalle boleta
        st.executeUpdate("INSERT INTO
detalle_factura(numero_producto,folio_factura,"
            + "codigo_producto,cantidad_producto,precio_producto) VALUES("
            + cor_factura + ","
            + numFactura + ","
            + codigo + ","
            + auxCant + ","
            + auxPre
            + ")");
        cont1 = cont1 + 1;
    }
}

```

```

    }

    JOptionPane.showMessageDialog(null, "Venta realizada con exito", "Aviso",
        JOptionPane.INFORMATION_MESSAGE);

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error al realizar la venta", "Aviso",
            JOptionPane.INFORMATION_MESSAGE);
    }
}

private void TBL_busquedaMouseClicked(java.awt.event.MouseEvent evt) {
    //Usado para darle foco al spinner de cantidad, una vez echo click
    //a un producto en la tabla de busqueda.

    if (TBL_busqueda.getSelectedRow() >= 0) {
        this.SP_cantidad.requestFocus();
    }
}

private void BTN_actualizarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón actualizar, para cargar el combo box de cliente,
    //en el caso de crear un cliente nuevo desde el programa de venta.

    CB_codigoCli.removeAllItems();
    CB_codigoCli.addItem("Seleccione...");
    llenarComboCliente();
}

private void TF_busCodKeyReleased(java.awt.event.KeyEvent evt) {
    //Usado en el input de código, para efectuar la búsqueda en la base de datos,
    //por el criterio indicado antes, al momento de digitar un número.

    // aquí buscará cuando escriba un código
    String auxCod = TF_busCod.getText();
    String auxNom = TF_busNom.getText();
    String CodProd, NomProd, PrecProd, StockProd, StockCrProd;
    try {
        Statement st = cn.createStatement();
        limpiarTabla("B");
    }
}

```

```

ResultSet lista = st.executeQuery("SELECT * FROM producto WHERE "
    + "codigo_producto LIKE '%" + auxCod + "%' AND "
    + "nombre_producto LIKE '%" + auxNom + "%'");
while (lista.next()) {
    CodProd = lista.getString("codigo_producto");
    NomProd = lista.getString("nombre_producto");
    PrecProd = lista.getString("precio_producto");
    StockProd = lista.getString("stock_producto");
    StockCrProd = lista.getString("stock_critico");
    DefaultTableModel modelo = (DefaultTableModel) TBL_busqueda.getModel();
    modelo.addRow(new Object[]{CodProd, NomProd, PrecProd, StockProd, StockCrProd});
}

} catch (SQLException ex) {
    Logger.getLogger(Venta.class.getName()).log(Level.SEVERE, null, ex);
}

}

private void TF_busNomKeyReleased(java.awt.event.KeyEvent evt) {
    //Usado en el input de nombre, para efectuar la búsqueda en la base de datos,
    //por el criterio indicado antes, al momento de digitar una letra.

    // aquí buscará cuando escriba un código
    String auxNom = TF_busNom.getText();
    String auxCod = TF_busCod.getText();

    String CodProd, NomProd, PrecProd, StockProd, StockCrProd;
    try {
        Statement st = cn.createStatement();
        limpiarTabla("B");
        ResultSet lista = st.executeQuery("SELECT * FROM producto WHERE "
            + "codigo_producto LIKE '%" + auxCod + "%' AND "
            + "nombre_producto LIKE '%" + auxNom + "%'");
        while (lista.next()) {
            CodProd = lista.getString("codigo_producto");
            NomProd = lista.getString("nombre_producto");
            PrecProd = lista.getString("precio_producto");
            StockProd = lista.getString("stock_producto");
            StockCrProd = lista.getString("stock_critico");
            DefaultTableModel modelo = (DefaultTableModel) TBL_busqueda.getModel();
            modelo.addRow(new Object[]{CodProd, NomProd, PrecProd, StockProd, StockCrProd});
        }
    }
}

```

```

    }

    } catch (SQLException ex) {
        Logger.getLogger(Venta.class.getName()).log(Level.SEVERE, null, ex);
    }

}

private void TF_documentoKeyTyped(java.awt.event.KeyEvent evt) {
    //Validador del campo documento, segun la eleccion, se define el largo del documento
    //y limitara su contenido, solo se podran ingresar numeros.
    char car = evt.getKeyChar();

    if (RB_boleta.isSelected()) {
        if (TF_documento.getText().length() >= 10) {
            evt.consume();
        }
        if (car < '0' || car > '9') {
            evt.consume();
        }
    } else if (RB_factura.isSelected()) {
        if (TF_documento.getText().length() >= 15) {
            evt.consume();
        }
        if (car < '0' || car > '9') {
            evt.consume();
        }
    }
}

private void TF_busCodKeyTyped(java.awt.event.KeyEvent evt) {
    //Validador del campo busqueda por codigo, se define el largo del codigo.

    if (TF_busCod.getText().length() >= 15) {
        evt.consume();
    }
}

private void BTN_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón cliente, para activar el programa modal
    //"Mantenedor de Cliente".

```

```

    Man_cliente frame = new Man_cliente(this, true);
    frame.setVisible(true);

}

private void TF_busNomKeyTyped(java.awt.event.KeyEvent evt) {
    //Validador del campo busqueda por codigo, se define el largo del codigo.

    if (TF_busCod.getText().length() >= 20) {
        evt.consume();
    }

}

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            Venta dialog = new Venta(new javax.swing.JFrame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                @Override
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });
}

//-----
// Definición de las funciones y procedimientos propias del programa
//-----

public void limpiarTabla(String tipo) {
    //Procedimiento usado para vaciar ambas tablas en el programa.
    try {
        if (tipo.equals("B")) {
            DefaultTableModel modelo = (DefaultTableModel) TBL_busqueda.getModel();
            int filas = TBL_busqueda.getRowCount();
            for (int i = 0; filas > i; i++) {

```

```

        modelo.removeRow(0);
    }
} else if (tipo.equals("N")) {
    DefaultTableModel modelo = (DefaultTableModel) TBL_normal.getModel();
    int filas = TBL_normal.getRowCount();
    for (int i = 0; filas > i; i++) {
        modelo.removeRow(0);
    }
}
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
}
}

public void llenarComboUser() {
    //Procedimiento usado para llenar el combo box de vendedores.

    int cod_user;
    String aux;

    try {
        Statement st = cn.createStatement();
        ResultSet combo = st.executeQuery("SELECT * FROM usuario");
        while (combo.next()) {
            cod_user = combo.getInt("codigo_usuario");
            aux = String.valueOf(cod_user);
            if (cod_user != 0) {
                CB_codigoVen.addItem(aux);
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "No se ha podido llenar", "Advertencia",
            JOptionPane.WARNING_MESSAGE);
    }
    TBL_normal.setDefaultEditor(Object.class, null);
}

public void llenarComboCliente() {
    //Procedimiento usado para llenar el combo box de clientes.

    String nom_cliente;

```

```

try {
    Statement st = cn.createStatement();
    ResultSet combo = st.executeQuery("SELECT * FROM cliente ORDER BY nombre_cliente");
    while (combo.next()) {
        nom_cliente = combo.getString("nombre_cliente");
        CB_codigoCli.addItem(nom_cliente);
    }
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "No se ha podido llenar", "Advertencia",
        JOptionPane.WARNING_MESSAGE);
}
TBL_normal.setDefaultEditor(Object.class, null);
}

```

```

public int numCorVen() {
    int num = 0;
    try {
        Statement st = cn.createStatement();
        ResultSet lista = st.executeQuery("SELECT MAX(numero_venta)maximo FROM venta ");
        if (lista.next()) {
            num = lista.getInt("maximo");
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error en base de datos", "Error",
            JOptionPane.ERROR_MESSAGE);
    }
    return num + 1;
}

```

```

public int numCorBol(String boleta) {
    //Funcion usada para crear el numero correlativo en la boleta.

    int num = 0;
    try {
        Statement st = cn.createStatement();
        ResultSet lista = st.executeQuery("SELECT MAX(numero_producto)maximo "
            + "FROM detalle_boleta "
            + "WHERE codigo_boleta = '" + boleta + "' ");
        if (lista.next()) {
            num = lista.getInt("maximo");
        }
    }
}

```

```

    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Error en base de datos", "Error",
        JOptionPane.ERROR_MESSAGE);
}
return num + 1;
}

public int numCorFact(String factura) {
    //Funcion usada para crear el numero correlativo en la factura.

    int num = 0;
    try {
        Statement st = cn.createStatement();
        ResultSet lista = st.executeQuery("SELECT MAX(numero_producto)maximo "
            + "FROM detalle_factura "
            + "WHERE folio_factura = '" + factura + "' ");
        if (lista.next()) {
            num = lista.getInt("maximo");
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error en base de datos", "Error",
            JOptionPane.ERROR_MESSAGE);
    }
    return num + 1;
}

public int actualizarProducto(String codProd, int cantProd) {
    //Funcion usada para validar que el stock a vender sea mayor que la cantidad solicitada.

    int ajuste = 0;
    int Stock = 0;
    String Nom = "";

    int invalido = 0;

    try {

        Statement st = (Statement) cn.createStatement();
        ResultSet lista = st.executeQuery("SELECT * from producto WHERE codigo_producto='" +
            codProd + "'");
        if (lista.next()) {

```

```

    Stock = lista.getInt("stock_producto");

    ajuste = Stock - cantProd;

    if (ajuste > 0) {
        st.executeUpdate("UPDATE producto "
            + "SET stock_producto = '" + ajuste + "' "
            + "WHERE codigo_producto ='" + codProd + "'");

        invalido = 1;
    } else {
        invalido = 0;
        JOptionPane.showMessageDialog(null, "La cantidad a vender supera a la cantidad del
            producto " + Nom + ".", "Aviso", JOptionPane.INFORMATION_MESSAGE);
    }
}
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Error en base de datos", "Error",
        JOptionPane.ERROR_MESSAGE);
}
return invalido;
}

private void actualizarTotal(int total) {
    //Funcion usada para mostrar a traves de un label la suma total de la venta.

    if (LBL_total.getText().equals("$0")) {
        String STtotal = String.valueOf(total);
        String format = "$".concat(STtotal);
        this.LBL_total.setText(format);
    } else {
        String cifra = LBL_total.getText().substring(1, LBL_total.getText().length());
        int auxCifra = Integer.valueOf(cifra);
        int aux = auxCifra + total;
        String STtotal = String.valueOf(aux);
        String format = "$".concat(STtotal);
        this.LBL_total.setText(format);
    }
}
}
}

```

ANEXO 3: CÓDIGO FUENTE DE MANTENEDOR DE PRODUCTO.

```
//-----
// Definición de las librerías propias y librerías propias de Java
//-----

package Proyecto.Mantenedores;
import Clases.Conectar;
import Clases.Sql;
import java.awt.Color;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
import java.sql.Connection;
import javax.swing.table.DefaultTableModel;

//-----
// Generación de instancias para establecer las conexión con la base de datos e inicio
// de componentes del programa
//-----

public final class Man_producto extends javax.swing.JDialog {

    //Conectarme a la base de datos.
    Conectar connection = new Conectar();

    //Tener un elemento cn con el cual nos permite hacer la sentencias.
    Connection cn = connection.getconnect();

    // Crea una instancia de la clase Sql.
    Sql sql = new Sql();

    // Se inician los componentes cuando es abierto en modo Frame.
    public Man_producto(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        this.setTitle("EL ARCA - Mantenedor de producto");
        this.setLocationRelativeTo(null);

        llenarComboMedida();
    }
}
```

```

llenarComboProveedor();
llenarComboCatalogo();
llenarTabla();

TBL_normal.getTableHeader().setReorderingAllowed(false);
TBL_normal.setRowSelectionAllowed(true);
TBL_normal.setColumnSelectionAllowed(false);

TBL_busqueda.getTableHeader().setReorderingAllowed(false);
TBL_busqueda.setRowSelectionAllowed(true);
TBL_busqueda.setColumnSelectionAllowed(false);

}

// Se inician los componentes cuando es abierto en modo Dialog.
public Man_producto(java.awt.Dialog parent, boolean modal) {
    super(parent, modal);
    initComponents();
    this.setTitle("EL ARCA - Mantenedor de producto");
    this.setLocationRelativeTo(null);

    llenarComboMedida();
    llenarComboProveedor();
    llenarComboCatalogo();
    llenarTabla();

    TBL_normal.getTableHeader().setReorderingAllowed(false);
    TBL_normal.setRowSelectionAllowed(true);
    TBL_normal.setColumnSelectionAllowed(false);

    TBL_busqueda.getTableHeader().setReorderingAllowed(false);
    TBL_busqueda.setRowSelectionAllowed(true);
    TBL_busqueda.setColumnSelectionAllowed(false);

}

//-----
// Definición de eventos y validaciones del programa
//-----

private void BTN_volverActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón Volver para cerrar la ventana modal.

```

```

    this.setVisible(false);
}

```

```

private void BTN_eliminarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usada por el botón Eliminar, para suprimir un producto según el código el mismo.

```

```

    String CodigoE = TF_codigo.getText();
    String NombreE = TF_nombre.getText();

```

```

    String query = "";
    ResultSet existe;

```

```

    try {
        Statement st = (Statement) cn.createStatement();
        query = "SELECT * FROM producto WHERE codigo_producto='" + CodigoE + "'";
        existe = st.executeQuery(query);
        if (existe.next() == true) {
            int j = JOptionPane.showConfirmDialog(null, "¿Está seguro que quiere eliminar " +
                NombreE + "?", "Confirmar Salida", JOptionPane.YES_NO_OPTION, 1);
            if (j == 0) {
                st.executeUpdate("DELETE FROM producto WHERE codigo_producto='" + CodigoE +
                    "'");
                JOptionPane.showMessageDialog(null, "Datos eliminados exitosamente", "Aviso",
                    JOptionPane.INFORMATION_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(null, "El producto no se puede eliminar",
                "Advertencia", JOptionPane.WARNING_MESSAGE);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "No existe el producto, no se puede eliminar",
            "Advertencia", JOptionPane.WARNING_MESSAGE);
    }
}

```

```

    TF_codigo.setText("");
    TF_nombre.setText("");
    TA_descripcion.setText("");
    CB_proveedor.setSelectedIndex(0);
    CB_catalogo.setSelectedIndex(0);
    CB_medida.setSelectedIndex(0);
    SP_precio.setValue(0);
    SP_stock.setValue(0);
    SP_stock_critico.setValue(0);

```

```

try {
    DefaultTableModel modelo1 = (DefaultTableModel) TBL_busqueda.getModel();
    int filas = TBL_busqueda.getRowCount();
    for (int i = 0; filas > i; i++) {
        modelo1.removeRow(0);
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error al limpiar la tabla", "Error",
        JOptionPane.ERROR_MESSAGE);
}
limpiarTabla();
llenarTabla();
}

private void BTN_agregarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usada por el boton Agregar, para efectuar una adicion en la base de datos.

    String codigoA = TF_codigo.getText();
    String nombreA = TF_nombre.getText();
    int medidaA;
    int catalogoA;
    String proveedorA;
    int precioA = Integer.valueOf(SP_precio.getValue().toString());
    int stockA = Integer.valueOf(SP_stock.getValue().toString());
    int stockCA = Integer.valueOf(SP_stock_critico.getValue().toString());
    String despcA = TA_descripcion.getText();

    try {
        if (codigoA.equals("")) {
            JOptionPane.showMessageDialog(null, "Ingrese un codigo", "Advertencia",
                JOptionPane.WARNING_MESSAGE);
        } else if (nombreA.equals("")) {
            JOptionPane.showMessageDialog(null, "Ingrese un nombre", "Advertencia",
                JOptionPane.WARNING_MESSAGE);
        } else if (CB_proveedor.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(null, "Seleccione un proveedor", "Advertencia",
                JOptionPane.WARNING_MESSAGE);
        } else if (CB_catalogo.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(null, "Seleccione un catálogo", "Advertencia",
                JOptionPane.WARNING_MESSAGE);
        } else if (CB_medida.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(null, "Seleccione una medida", "Advertencia",
                JOptionPane.WARNING_MESSAGE);
        }
    }
}

```

```

} else if (stockA <= stockCA) {
    JOptionPane.showMessageDialog(null, "El stock critico no puede ser mayor a stock
        ingresado", "Advertencia", JOptionPane.WARNING_MESSAGE);
} else {
    Statement st = (Statement) cn.createStatement();
    /* las siguientes 3 líneas son para que encuentre el rut del proveedor*/

    ResultSet buscarDato = st.executeQuery("SELECT rut_proveedor FROM proveedor
        WHERE nombre_proveedor = '" + CB_proveedor.getSelectedItemAt() + "'");
    buscarDato.next();
    proveedorA = buscarDato.getString("rut_proveedor");

    buscarDato = st.executeQuery("SELECT codigo_catalogo FROM catalogo WHERE
        nombre_catalogo = '" + CB_catalogo.getSelectedItemAt() + "'");
    buscarDato.next();
    catalogoA = buscarDato.getInt("codigo_catalogo");

    buscarDato = st.executeQuery("SELECT codigo_medida FROM medida WHERE
        nombre_medida = '" + CB_medida.getSelectedItemAt() + "'");
    buscarDato.next();
    medidaA = buscarDato.getInt("codigo_medida");

    st.executeUpdate("INSERT INTO producto
        (codigo_producto,rut_proveedor,codigo_medida,codigo_catalogo,"
        +"nombre_producto,descripcion_producto,stock_producto,stock_critico,precio_
        producto) VALUES("
        + codigoA + "," // varchar
        + proveedorA + "," // varchar
        + medidaA + "," //tinyint
        + catalogoA + "," //tinyint
        + nombreA + "," // varchar
        + despcA + "," // varchar
        + stockA + "," // int
        + stockCA + "," // int
        + precioA // int
        + ")");
    JOptionPane.showMessageDialog(null, "Datos guardados con exito", "Aviso",
        JOptionPane.INFORMATION_MESSAGE);
}
} catch (SQLException e) {
    String nl = System.getProperty("line.separator");
    JOptionPane.showMessageDialog(null,
        "No se ha podido ingresar datos" + nl

```

```

        + nombreA + " ya existe" + nl
        + "Verifique sus datos", "Error", JOptionPane.ERROR_MESSAGE);

    }

    limpiarTabla();
    llenarTabla();

}

private void BTN_modificarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usada por el boton Modificar, para efectuar una actualización en un producto.

    this.TF_nombre.setEnabled(true);
    this.TA_descripcion.setEnabled(true);
    this.CB_proveedor.setEnabled(true);
    this.CB_catalogo.setEnabled(true);
    this.CB_medida.setEnabled(true);
    this.SP_precio.setEnabled(true);
    this.SP_stock.setEnabled(true);
    this.SP_stock_critico.setEnabled(true);

    String codigoM = TF_codigo.getText();
    String nombreM = TF_nombre.getText();
    int medidaM;
    int catalogoM;
    String proveedorM;
    String precioM = SP_precio.getValue().toString();
    String stockM = SP_stock.getValue().toString();
    String stockCM = SP_stock_critico.getValue().toString();
    String descM = TA_descripcion.getText();
    ResultSet existe = null;

    try {
        if (codigoM.equals("")) {
            JOptionPane.showMessageDialog(null, "Ingrese un codigo", "Aviso",
                JOptionPane.WARNING_MESSAGE);
        } else if (nombreM.equals("")) {
            JOptionPane.showMessageDialog(null, "Ingrese un nombre", "Aviso",
                JOptionPane.WARNING_MESSAGE);
        } else if (CB_proveedor.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog(null, "Seleccione un proveedor", "Aviso",
                JOptionPane.WARNING_MESSAGE);
        }
    }
}

```



```

        + "" WHERE codigo_producto ="" + codigoM + """);

        JOptionPane.showMessageDialog(null, "Datos Modificados con exito", "Aviso",
        JOptionPane.INFORMATION_MESSAGE);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Datos no actualizados", "Alerta",
        JOptionPane.INFORMATION_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(null, "No existe persona, no se puede modificar",
        "Alerta", JOptionPane.INFORMATION_MESSAGE);
    }
    }
    }
    } catch (SQLException e) {
    }
    limpiarTabla();
    llenarTabla();
}

```

```

private void BTN_buscarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón Consultar para generar una lista de coincidencias
    //según el codigo ingresado, y desplegar esos productos en la lista de busqueda.

```

```

    this.TF_codigo.setEnabled(true);
    this.TF_nombre.setEnabled(true);
    this.TA_descripcion.setEnabled(true);
    this.CB_proveedor.setEnabled(true);
    this.CB_catalogo.setEnabled(true);
    this.CB_medida.setEnabled(true);
    this.SP_precio.setEnabled(true);
    this.SP_stock.setEnabled(true);
    this.SP_stock_critico.setEnabled(true);

```

```

    //Busqueda por codigo del producto
    String busCod = TF_codigo.getText();
    String busNom = TF_nombre.getText();

```

```

    String Cod = "";
    String Nom = "";
    String Pro = "";
    String Nom_Med = "";
    String Nom_Cat = "";

```

```

String Nom_Pro = "";
String Des = "";
int Cat = 0;
int Med = 0;
int Pre = 0;
int Stock = 0;
int Stock_c = 0;

int filas;
String query = "select * from producto WHERE "
    + "codigo_producto LIKE '%" + busCod + "%' AND "
    + "nombre_producto LIKE '%" + busNom + "%' ";

try {
    if (busCod.equals("") && busNom.equals("")) {
        JOptionPane.showMessageDialog(null, "Ingrese Codigo y/o Nombre para consultar",
            "Aviso", JOptionPane.WARNING_MESSAGE);
    } else {
        Statement st = cn.createStatement();
        ResultSet lista = st.executeQuery(query);

        DefaultTableModel modelo = (DefaultTableModel) TBL_busqueda.getModel();

        filas = TBL_busqueda.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo.removeRow(0);
        }
        while (lista.next()) {
            Cod = lista.getString("codigo_producto");
            Nom = lista.getString("nombre_producto");
            Pro = lista.getString("rut_proveedor");
            Cat = lista.getInt("codigo_catalogo");
            Med = lista.getInt("codigo_medida");
            Des = lista.getString("descripcion_producto");
            Pre = lista.getInt("precio_producto");
            Stock = lista.getInt("stock_producto");
            Stock_c = lista.getInt("stock_critico");

            Nom_Med = sql.TablaNomMed(Med);
            Nom_Pro = sql.TablaNomPro(Pro);
            Nom_Cat = sql.TablaNomCat(Cat);

            modelo.addRow(new Object[]{Cod, Nom, Des, Nom_Pro, Nom_Cat, Nom_Med, Pre,

```

```

        Stock, Stock_c});

    }
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "Error al solicitar consulta", "Error",
        JOptionPane.ERROR_MESSAGE);
}

}

private void BTN_cancelarActionPerformed(java.awt.event.ActionEvent evt) {
    //Usado en el botón cancelar para resetear de manera general los
    //valores de los componentes.

    this.BTN_agregar.setEnabled(true);
    this.BTN_modificar.setEnabled(true);
    this.BTN_eliminar.setEnabled(true);
    this.BTN_volver.setEnabled(true);
    this.BTN_buscar.setEnabled(true);

    this.TF_codigo.setEnabled(true);
    this.TF_nombre.setEnabled(true);
    this.TA_descripcion.setEnabled(true);
    this.CB_proveedor.setEnabled(true);
    this.CB_catalogo.setEnabled(true);
    this.CB_medida.setEnabled(true);
    this.SP_precio.setEnabled(true);
    this.SP_stock.setEnabled(true);
    this.SP_stock_critico.setEnabled(true);

    TF_codigo.setText("");
    TF_nombre.setText("");
    TA_descripcion.setText("");
    CB_proveedor.setSelectedIndex(0);
    CB_catalogo.setSelectedIndex(0);
    CB_medida.setSelectedIndex(0);
    SP_precio.setValue(1);
    SP_stock.setValue(1);
    SP_stock_critico.setValue(1);

    try {
        DefaultTableModel modelo1 = (DefaultTableModel) TBL_busqueda.getModel();

```

```

        int filas = TBL_busqueda.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo1.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}

```

```

private void TBL_normalMouseClicked(java.awt.event.MouseEvent evt) {
    //Al momento de hacer un click en la tabla de productos o tabla normal
    //se procede a tomar el valor de cada columna e ingresarla en su campo respectivo
    //en el programa.

```

```

    DefaultTableModel tm = (DefaultTableModel) TBL_normal.getModel();

```

```

    String cod = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 0));
    this.TF_codigo.setText(cod);

```

```

    String nom = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 1));
    this.TF_nombre.setText(nom);

```

```

    String des = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 2));
    this.TA_descripcion.setText(des);

```

```

    String pro = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 3));
    this.CB_proveedor.setSelectedItem(pro);

```

```

    String cat = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 4));
    this.CB_catalogo.setSelectedItem(cat);

```

```

    String med = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 5));
    this.CB_medida.setSelectedItem(med);

```

```

    String pre = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 6));
    int preaux = Integer.parseInt(pre);
    this.SP_precio.setValue(preaux);

```

```

    String stock = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 7));
    int stockaux = Integer.parseInt(stock);
    this.SP_stock.setValue(stockaux);

```

```

String stock_c = String.valueOf(tm.getValueAt(TBL_normal.getSelectedRow(), 8));
int stock_caux = Integer.parseInt(stock_c);
this.SP_stock_critico.setValue(stock_caux);

}

private void TBL_busquedaMouseClicked(java.awt.event.MouseEvent evt) {
    //Al momento de hacer un click en la tabla de busqueda se procede a
    //tomar el valor de cada columna e ingresarla en su campo respectivo en el programa.

    DefaultTableModel tm = (DefaultTableModel) TBL_busqueda.getModel();

    String cod = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 0));
    this.TF_codigo.setText(cod);

    String nom = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 1));
    this.TF_nombre.setText(nom);

    String des = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 2));
    this.TA_descripcion.setText(des);

    String pro = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 3));
    this.CB_proveedor.setSelectedItem(pro);

    String cat = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 4));
    this.CB_catalogo.setSelectedItem(cat);

    String med = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 5));
    this.CB_medida.setSelectedItem(med);

    String pre = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 6));
    int preaux = Integer.parseInt(pre);
    this.SP_precio.setValue(preaux);

    String stock = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 7));
    int stockaux = Integer.parseInt(stock);
    this.SP_stock.setValue(stockaux);

    String stock_c = String.valueOf(tm.getValueAt(TBL_busqueda.getSelectedRow(), 8));
    int stock_caux = Integer.parseInt(stock_c);
    this.SP_stock_critico.setValue(stock_caux);

}

```

```

private void TA_descripcionKeyReleased(java.awt.event.KeyEvent evt) {
    //Utilizado en el text area, para limitar la cantidad de carecteres a 120,
    //y el cuadro contados (LB_contador) cambia de color al faltar solo 10 caracteres.

    String text = TA_descripcion.getText();
    int largo = 120 - text.length();
    String largoText = String.valueOf(largo);

    if (largo > -1) {
        if (largo < 11) {
            LB_contador.setForeground(Color.red);
            LB_contador.setText(largoText);
        } else {
            LB_contador.setText(largoText);
            LB_contador.setForeground(Color.orange);
        }
    } else {
        String textaux = text.substring(0, 120);
        TA_descripcion.setText(textaux);
    }
}

private void TF_codigoKeyTyped(java.awt.event.KeyEvent evt) {
    //Usado para validar el largo de los caracteres ingresados en el codigo.

    if (TF_codigo.getText().length() >= 15) {
        evt.consume();
    }
}

private void TF_nombreKeyTyped(java.awt.event.KeyEvent evt) {
    //Usado para validar el largo de los caracteres ingresados en el nombre.

    if (TF_nombre.getText().length() >= 20) {
        evt.consume();
    }
}

private void TF_codigoKeyReleased(java.awt.event.KeyEvent evt) {
    //Usado para convertir a mayuscula los caracteres ingresados

```

```

        //por teclado en el campo codigo.

String auxCod = TF_codigo.getText().toUpperCase();
TF_codigo.setText(auxCod);

}

private void TF_nombreKeyReleased(java.awt.event.KeyEvent evt) {
    //Usado para convertir a mayuscula los caracteres ingresados
    //por teclado en el campo nombre.

String aux = TF_nombre.getText().toUpperCase();
TF_nombre.setText(aux);

}

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            Man_producto dialog = new Man_producto(new javax.swing.JFrame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                @Override
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });

//-----
// Definición de las funciones y procedimientos propias del programa
//-----

public void llenarTabla() {
    //Funcion utilizada para mostrar todo los productos en la tabla normal o de tabla de
    //producto una vez iniciado el mantenedor
String Cod = "";
String Nom = "";
String Pro = "";
int Cat = 0;

```

```

int Med = 0;
String Nom_Med = "";
String Nom_Cat = "";
String Nom_Pro = "";
String Des = "";
int Pre = 0;
int Stock = 0;
int Stock_c = 0;

String query = "SELECT * FROM producto";

try {
    Statement st = cn.createStatement();
    ResultSet lista = st.executeQuery(query);
    while (lista.next()) {
        Cod = lista.getString("codigo_producto");
        Nom = lista.getString("nombre_producto");
        Pro = lista.getString("rut_proveedor");
        Cat = lista.getInt("codigo_catalogo");
        Med = lista.getInt("codigo_medida");
        Des = lista.getString("descripcion_producto");
        Pre = lista.getInt("precio_producto");
        Stock = lista.getInt("stock_producto");
        Stock_c = lista.getInt("stock_critico");

        Nom_Med = sql.TablaNomMed(Med);
        Nom_Pro = sql.TablaNomPro(Pro);
        Nom_Cat = sql.TablaNomCat(Cat);

        if (!Nom.equals("system")) {
            DefaultTableModel modelo = (DefaultTableModel) TBL_normal.getModel();
            modelo.addRow(new Object[]{Cod, Nom, Des, Nom_Pro, Nom_Cat, Nom_Med, Pre,
                Stock, Stock_c});
        }
    }
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "Error en mostrar los datos", "Advertencia",
        JOptionPane.WARNING_MESSAGE);
}
TBL_busqueda.setDefaultEditor(Object.class, null);
}

```

```

public void llenarComboMedida() {
    //Funcion utilizada para llenar todas la medidas de la tabla medida en la
    //base de datos y desplerarlos en el combo box.

    String nombre_med;
    try {
        Statement st = cn.createStatement();
        ResultSet combo = st.executeQuery("SELECT * FROM medida ORDER BY nombre_medida
            asc");
        while (combo.next()) {
            nombre_med = combo.getString("nombre_medida");
            CB_medida.addItem(nombre_med);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "No se ha podido llenar", "Advertencia",
            JOptionPane.WARNING_MESSAGE);
    }
    TBL_busqueda.setDefaultEditor(Object.class, null);
}

```

```

public void llenarComboProveedor() {
    //Funcion utilizada para llenar todas los proveedores de la tabla proveedor en la
    //base de datos y desplerarlos en el combo box.

    String nombre_pro;
    try {
        Statement st = cn.createStatement();
        ResultSet combo = st.executeQuery("SELECT * FROM proveedor ORDER BY
            nombre_proveedor asc");
        while (combo.next()) {
            nombre_pro = combo.getString("nombre_proveedor");
            CB_proveedor.addItem(nombre_pro);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "No se ha podido llenar combo proveedor",
            "Advertencia", JOptionPane.WARNING_MESSAGE);
    }
    TBL_busqueda.setDefaultEditor(Object.class, null);
}

```

```

public void llenarComboCatalogo() {
    //Funcion utilizada para llenar todas los tipos de productos de la tabla catalogo en la

```

//base de datos y desplazarlos en el combo box.

```
String nombre_cat;
try {
    Statement st = cn.createStatement();
    ResultSet combo = st.executeQuery("SELECT * FROM catalogo ORDER BY nombre_catalogo
        asc");
    while (combo.next()) {
        nombre_cat = combo.getString("nombre_catalogo");
        CB_catalogo.addItem(nombre_cat);
    }
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "No se ha podido llenar combo catalogo",
        "Advertencia", JOptionPane.WARNING_MESSAGE);
}
TBL_busqueda.setDefaultEditor(Object.class, null);
}

public void limpiarTabla() {
    //Funcion utilizada para vaciar todos los datos en ambas tablas de formulario.

    try {
        DefaultTableModel modelo1 = (DefaultTableModel) TBL_busqueda.getModel();
        int filas = TBL_busqueda.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo1.removeRow(0);
        }
        DefaultTableModel modelo2 = (DefaultTableModel) TBL_normal.getModel();
        filas = TBL_normal.getRowCount();
        for (int i = 0; filas > i; i++) {
            modelo2.removeRow(0);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al limpiar la tabla.");
    }
}
}
```