

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

**SISTEMA INFORMÁTICO PARA LA GESTIÓN DE VENTAS E INVENTARIO
DE LA EMPRESA KARISHMAS**

Trabajo de Titulación para optar al título
de Técnico Universitario en
INFORMÁTICA

Alumnos:

Javier Andrés Echeverría Farías

Jorge Felipe González Orellana

Profesor Guía:

Catherine Vanessa Gómez Barrera

2017

RESUMEN

KEYWORDS: VENTA, SISTEMA INFORMÁTICO, JAVA SE, INVENTARIO, AUTOMATIZACIÓN, TRABAJO DE TÍTULO.

El presente trabajo de título, Sistema informático para gestión de venta e inventario fue desarrollado para la empresa KARISHMAS IMPORTACIONES Ltda., cuya dependencia se sitúa en Salvador Sanfuentes 2541, local N°6, Santiago.

El Sistema fue propuesto por uno de los integrantes del proyecto al dueño de la tienda para poder optimizar y automatizar las tareas, las cuales eran realizadas de forma manual. Por lo tanto el objetivo general del sistema es automatizar y gestionar correctamente las tareas de venta, control de stock y registros de clientes, proveedores y productos, beneficiando la gestión y tiempo de respuesta para cualquier acción fundamental para el negocio.

Para la construcción del sistema se utilizó como herramienta de desarrollo la edición Standard de Java Netbeans, especializada en herramientas de escritorio, y para la gestión de manejo de datos se utiliza SQL en la plataforma PhpMyAdmin de Xampp, El sistema cuenta con la gestión de 4 pilares fundamentales de funcionamiento que son: Gestión de ventas, Control de productos o inventario, Control de clientes y Control de proveedores.

La estructura de la presente memoria se presenta de la siguiente manera:

- ❖ Capítulo I: Se plantea la situación actual del funcionamiento de la tienda, dando como propuesta en solución un sistema informático que automatizará la mayor parte de las actividades que actualmente se realizan de forma manual, como por ejemplo: ventas, inventario, contactos, reservas, etc. A demás se definen flujos de salida y entrada de información en el sistema.
- ❖ Capítulo II: Se define el recurso computacional a utilizar, de tal forma que el software pueda ser explotado sin inconveniente alguno. Así mismo se definen los archivos que gestionarán el funcionamiento y representarán las entidades que componen el sistema presentado.
- ❖ Capítulo III: Se muestran los componentes desarrollados para este sistema, tales como: menú, programas, y diseños de pantalla que podrá ver el usuario al usar el sistema.

Seguido a esto se presentan una conclusión, una bibliografía y un anexo que consiste en el código fuente del software desarrollado.

ÍNDICE

RESUMEN

INTRODUCCIÓN

CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LOGICO

- 1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN
- 1.2. DESCRIPCIÓN DE SITUACIÓN ACTUAL
- 1.3. PROBLEMAS DETECTADOS
- 1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO
 - 1.4.1. Objetivo general del sistema propuesto
 - 1.4.2. Objetivos específicos del sistema propuesto
 - 1.4.3. Beneficios del sistema propuesto
 - 1.4.4. Descripción general de solución propuesta
 - 1.4.5. Diagramas de Flujo administrativo
 - 1.4.6. Estructura funcional del sistema
 - 1.4.7. Descripción de fórmulas y cálculos empleados
 - 1.4.8. Información que se manejará
 - 1.4.9. Estructura de códigos
 - 1.4.10. Condicionantes de diseño

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

- 2.1. DESCRIPCIÓN DEL RECURSO COMPUTACIONAL
 - 2.1.1. Configuración del sistema
 - 2.1.2. Software utilizado
- 2.2. DESCRIPCIÓN DE ARCHIVOS
 - 2.2.1. Boleta
 - 2.2.2. Detalle boleta
 - 2.2.3. Producto
 - 2.2.4. Ajuste de stock
 - 2.2.5. Categoría
 - 2.2.6. Cliente
 - 2.2.7. Mail Cliente
 - 2.2.8. Fono Cliente
 - 2.2.9. Región

- 2.2.10. Comuna
- 2.2.11. Factura
- 2.2.12. Detalle de factura
- 2.2.13. Reserva
- 2.2.14. Detalle de reserva
- 2.2.15. Usuario
- 2.2.16. Proveedor
- 2.2.17. Mails de proveedor
- 2.2.18. Teléfonos de proveedor
- 2.2.19. Producto de proveedor
- 2.2.20. Tarjetas
- 2.2.21. Cuotas

CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS

- 3.1. DIAGRAMA MODULAR
- 3.2. DIAGRAMA DE MENÚS
- 3.3. LISTA DE PROGRAMAS
- 3.4. DESCRIPCIÓN DE PROGRAMAS SEÑALADOS
 - 3.4.1. Venta con boleta
 - 3.4.2. Ingresar producto
 - 3.4.3. Consultar Producto
 - 3.4.4. Modificar Producto
 - 3.4.5. Eliminar producto
 - 3.4.6. Ingresar Cliente
 - 3.4.7. Consultar Cliente
 - 3.4.8. Modificar cliente
 - 3.4.9. Eliminar Cliente
 - 3.4.10. Ingresar Cuotas
 - 3.4.11. Ingresar Reserva

CONCLUSIONES Y RECOMENDACIONES

BIBLIOGRAFIA

ANEXO A: CÓDIGO FUENTE DEL SISTEMA DE VENTAS E INVENTARIO DESARROLLADO

ÍNDICE DE FIGURAS

Figura 3-1.	Vista de venta con boleta
Figura 3-2.	Vista de venta con factura
Figura 3-3.	Vista de ingresar Producto
Figura 3-4.	Vista Consultar Producto
Figura 3-5.	Vista modificar producto
Figura 3-6.	Vista eliminar producto
Figura 3-7.	Vista ingresar cliente
Figura 3-8.	Vista de consultar cliente
Figura 3-9.	Vista modificar cliente
Figura 3-10.	Vista eliminar Cliente
Figura 3-11.	Vista Ingresar Cuotas
Figura 3-12.	Vista Ingresar Reserva

ÍNDICE DE TABLAS

Tabla 2-1.	Estructura archivo Boleta
Tabla 2-2.	Estructura archivo detalle boleta
Tabla 2-3.	Estructura archivo producto
Tabla 2-4.	Estructura archivo Ajuste de stock
Tabla 2-5.	Estructura archivo Categoría
Tabla 2-6.	Estructura archivo Cliente
Tabla 2-7.	Estructura archivo mail cliente
Tabla 2-8.	Estructura archivo teléfono cliente
Tabla 2-9.	Estructura archivo Región
Tabla 2-10.	Estructura archivo comuna
Tabla 2-11.	Estructura archivo Factura
Tabla 2-12.	Estructura archivo detalle factura
Tabla 2-13.	Estructura archivo Reserva
Tabla 2-14.	Estructura archivo detalle reserva
Tabla 2-15.	Estructura archivo Usuario
Tabla 2-16.	Estructura archivo Proveedor
Tabla 2-17.	Estructura archivo Mail del Proveedor

Tabla 2-18.	Estructura archivo teléfono de proveedor
Tabla 2-19.	Estructura archivo producto de proveedor
Tabla 2-20.	Estructura archivo Tarjetas
Tabla 2-21.	Estructura archivo cuotas
Tabla 3-1.	Lista de programas del sistema propuesto

ÍNDICE DE DIAGRAMAS

Diagrama 1-1.	Proceso de venta
Diagrama 1-2.	Proceso de reserva de producto
Diagrama 1-3.	Proceso de venta por teléfono
Diagrama 1-4.	Diagrama modular del sistema
Diagrama 3-1.	Diagrama modular del sistema propuesto
Diagrama 3-2.	Despliegue de menú para el nivel usuario
Diagrama 3-3.	Despliegue de menú para el nivel Súper usuario
Diagrama 3-4.	Diagrama de bloques venta con boleta
Diagrama 3-5.	Diagrama de bloques venta con factura
Diagrama 3-6.	Diagrama de bloques Ingresar producto
Diagrama 3-7.	Diagrama de bloques Consultar Producto
Diagrama 3-8.	Diagrama de bloques Modificar Producto
Diagrama 3-9.	Diagrama de bloques Eliminar Producto
Diagrama 3-10.	Diagrama de bloques Ingresar Cliente
Diagrama 3-11.	Diagrama de bloques Consultar Cliente
Diagrama 3-12.	Diagrama de bloques Modificar cliente
Diagrama 3-13.	Diagrama de bloques eliminar Cliente
Diagrama 3-14.	Diagrama de bloques Ingresar Cuotas
Diagrama 3-15.	Diagrama de bloques Ingresar Reserva

INTRODUCCIÓN

A continuación se expondrá la realización del “sistema informático para la gestión de ventas e inventario” para la empresa Karishmas importaciones ubicada en la ciudad de Santiago, que se dedica a la compra y venta de muebles para tiendas y que su dueño es el señor Suresh Pahuarni.

La idea del proyecto nace a raíz de la necesidad de la automatizar los procesos de compra y venta de la empresa KARISHMAS IMPORTACIONES, siendo el objetivo, el crear un sistema de información que apoye las funciones generales de venta para permitir una mejor administración de las transacciones que se realizan, para también reducir los tiempos de atención al cliente, además buscamos mantener un control del stock, almacenar información detallada tanto de clientes, proveedores y de productos e integrar la opción de reserva con sus posibles estados (pendiente, realizada y cancelada), todo esto dentro de una interfaz amigable y fácil de usar.

Este sistema está construido en lenguaje java para la parte de aplicación y se utiliza MySql como motor para la base de datos.

CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LOGICO

1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN

La empresa KARISHMAS IMPORTACIONES, se fundó en octubre del 2012 en Santiago de Chile por Suresh Pahuarni dentro del rubro de importación y ventas de artículos de armado y/o apertura de locales. En la actualidad, KARISHMAS IMPORTACIONES Ltda. Abastece sus productos directamente desde proveedores de China y cuenta con convenios de pago por transferencia electrónica, tales como REDCOMPRA, MASTERCARD, MAGNA, VISA, AMERICAN EXPRES, DINERS CLUB. Los clientes con los que cuenta la empresa son pequeñas tiendas que necesitan mueblería para mostrar sus productos o necesitan revender los productos. No se cuenta con ningún convenio con alguna empresa externa.

Entre los artículos que vende la tienda, cabe destacar:

- ❖ Vitrinas.
- ❖ Maniqués.
- ❖ Cajas fuertes.
- ❖ Ganchos para panel ranurado.
- ❖ Mostradores (lentes, zapatos, pulseras, joyas, etc.)

El personal de la tienda, está conformado por:

- ❖ **Dueño:** Quien toma las decisiones sobre las acciones que afecten al local, quien autoriza o rechaza un trato sea de compra o venta. Sólo de vez en cuando se presenta en el local.
- ❖ **Vendedor:** Persona que atiende el mesón, interactúa directamente con los clientes, se comunica directamente con el dueño para cualquier consulta, realiza todos los documentos con los que cuenta la empresa, maneja los flujos de dinero.
- ❖ **Bodeguero:** El responsable de conocer el stock con el que se cuenta en la tienda, de cada despacho que se realice, porque es quien revisa si el producto enviado es efectivamente el solicitado por el cliente y si está en excelentes condiciones para ser enviado.

LA SUCURSAL DE LA TIENDA

SALVADOR SANFUENTES 2541, LOCAL N° 6, A PASOS DE METRO

U.L.A. SANTIAGO.

HORARIO DE ATENCIÓN:

LUNES A VIERNES: 10:00 – 19:00

SÁBADO: 10:00 – 13:00

1.2. DESCRIPCIÓN DE SITUACIÓN ACTUAL

Actualmente KARISHMAS IMPORTACIONES Ltda., no cuenta con un sistema automatizado que administre las funciones de venta, inventario, reserva de productos, contacto de clientes y proveedores, ni pérdida de productos, sino que todo está separado en distintas documentaciones de tipo manual. Aproximadamente dos veces por semana se toman las boletas y facturas y se hace un resumen con los datos de ventas (Día de boleta, número de primera boleta del día, número de ultima boleta del día, total ventas con boleta, día de factura, número de factura, nombre de cliente, rut cliente, valor neto de factura, valor IVA de producto, valor total de factura) en un acta manuscrita. Una o dos veces al mes el dueño de la tienda toma este cuaderno, transcribe este resumen a una planilla de Excel, la cual, lleva los registros de las ventas realizadas a lo largo del tiempo.

A continuación los procesos que se realizan en la tienda:

- a) **Proceso de venta:** Llega el cliente y elige su(s) producto(s) para comprar (esto también se puede hacer por teléfono), luego se verifica el stock del producto en bodega manualmente, si no hay stock, manualmente se registran los datos del cliente y el producto elegido en una agenda (nombre, teléfono, producto, cantidad) para su posterior llamado, si hay stock, el cliente decide si se hará una boleta o una factura, si es factura, se toman los datos de éste (fecha, nombre, dirección, comuna, giro, teléfono, rut, ciudad y detalle de productos) , se detallan los productos comprados y se calcula el valor total (todo esto es hecho a mano), si es boleta, se anotan los códigos de los productos en la parte trasera de la boleta y se calcula el valor total de la venta (esto también es hecho a mano), el cliente puede elegir su forma de pago (debito, crédito o efectivo).
- b) **Proceso compra:** Para el pedido de productos, el dueño de la tienda viaja personalmente a la feria de Cantón ubicada en China (Feria realizada dos veces

al año, en donde muchas fábricas se juntan en un solo sitio para exponer sus productos) y elige sus productos, la fábrica toma registro de esto y envía vía email la solicitud de pago al dueño quien decide finalmente si pagar o no su producto, al realizar el pago, el dueño de la tienda elige la empresa de transporte que desea y hace la transferencia bancaria a la fábrica, quienes a su vez, envían una lista de productos asociados a la empresa KARISHMAS IMPORTACIONES Ltda., en este caso, a la compañía de transporte de los productos, ésta crea la packing list (lista de embalaje) con los datos (Código de producto, nombre de producto, cantidad de paquetes, cantidad por paquete, peso de paquete) y la envía al dueño de la tienda (vía e-mail) para que pueda revisar si la lista corresponde fielmente a las facturas realizadas por las fábricas. Una vez se comprueba que los productos son realmente los que deben transportarse, el envío llega al puerto de San Antonio, Chile, y finalmente a la sucursal de la tienda.

- c) **Proceso reserva de producto:** Reservar un producto refiere al hecho de que un producto no puede ser vendido, porque éste fue reservado anteriormente por un cliente (sin necesariamente haberlo pagado). Para esto el cliente puede dirigirse a la tienda o llamar por teléfono, si lo desea, solicitando algún producto, la vendedora da aviso al bodeguero para que cuente manualmente en bodega la existencia de stock que el cliente ha solicitado, si la cantidad en bodega satisface la cantidad que el cliente requiere, se le solicitan los datos correspondientes para reservar su producto (nombre cliente, nombre producto, cantidad de producto) manualmente en una agenda, se le informa del pedido al bodeguero y éste separa los productos para que no sean vendidos, si la cantidad en stock no satisfacen la necesidad del cliente, se registran los datos de pedido para el posterior llamado al cliente en cuanto el stock sea repuesto.
- d) **Proceso venta por teléfono:** El cliente llama por teléfono y elige sus productos, la vendedora da aviso al bodeguero para que cuente manualmente en bodega la existencia de stock que el cliente ha solicitado, si la cantidad en bodega satisface la cantidad que el cliente requiere, se le solicita la transferencia bancaria, al realizarla decide si quiere recibo de boleta o factura, si elige boleta se procede a llenar una guía de despacho con los datos de los productos enviados y datos de cliente (Nombre de cliente, dirección, giro, rut, comuna/ciudad, teléfono y detalles de productos), para que la compañía de encomienda pueda realizar el envío a la dirección expresada en la guía de despacho, si el cliente opta por factura, no es necesaria la guía de despacho porque los datos de cliente y detalles de productos están expresados en la misma factura (fecha, nombre, dirección, comuna, giro, teléfono, rut, ciudad y detalle de productos). Una vez llenado el

documento que registre el recibo de venta el bodeguero se dirige con los productos a la compañía de encomiendas (tur-bus cargo para el caso de esta empresa), los entrega junto a la factura y la compañía de encomienda genera el registro de envío con el estado “por pagar”, el bodeguero va a la tienda, y le entrega a la vendedora el registro de la encomienda, la vendedora le envía al cliente una fotografía del registro vía e-mail.

Los documentos utilizados actualmente, son los siguientes:

- 1) **Guía de Despacho:** Documento utilizado para comprobar los productos que se despachan del local a los clientes que son de regiones y eligen boleto, así mismo, cada vez que un cliente de confianza solicita productos con pago a plazo, se le entregan los productos con este documento para dejar registro de los productos que se han entregado, actualmente no hay una forma de controlar el pago a plazo más que el acuerdo de palabra entre el comprador y el vendedor. Cuando se envían los productos en la guía de despacho se registran los siguientes datos:

- ❖ Fecha.
- ❖ Nombre cliente.
- ❖ Dirección.
- ❖ Giro.
- ❖ Rut.
- ❖ Comuna/Ciudad.
- ❖ Giro.
- ❖ Detalle de productos.
- ❖ Cantidad de producto.
- ❖ Detalle de producto.
- ❖ Precio unitario producto.

- 2) **Factura:** Documento utilizado como recibo de compra por clientes que cuentan con negocio o empresa propia para luego cobrar la retención del IVA. Los datos registrados en el documento de factura son:

- ❖ Fecha.
- ❖ Nombre cliente.
- ❖ Dirección.
- ❖ Comuna.
- ❖ Giro.
- ❖ Teléfono.
- ❖ Rut.

- ❖ Ciudad.
- ❖ Cantidad de producto.
- ❖ Detalle de producto.
- ❖ Precio unitario producto.
- ❖ Total por producto.
- ❖ Valor neto.
- ❖ Valor IVA.
- ❖ Total.

3) **Packing list:** La Packing list o lista de embalaje es el documento enviado por la compañía que transportó los productos desde China hasta Chile, es de utilidad tanto para el dueño de la tienda, la vendedora y el bodeguero, este documento tiene los datos completos por cada producto, sin el precio, pero no es problema porque el dueño manualmente inserta los valores de venta en esta lista, a su vez le sirve a él para saber los productos que se embarcaron son efectivamente los que compró, para la vendedora es de suma utilidad porque puede revisar los precios de los productos si aún no se los sabe, y para el bodeguero es de vital importancia, ya que, lo necesita para tener un registro de los productos que se encuentran en bodega y el código de cada uno. Los datos con los que cuenta la packing list por cada producto son:

- ❖ Código de producto.
- ❖ Nombre de producto.
- ❖ Cantidad de paquetes.
- ❖ Cantidad de productos por paquete.
- ❖ Peso de paquete.

4) **Agenda de reservas:** Es una agenda común y corriente, en la cual, cada vez que no hay productos y un cliente lo solicita, se almacenan los datos mínimos del cliente y productos solicitados. Los datos que se manejan en esta agenda de reserva de productos son:

- ❖ Nombre de cliente.
- ❖ Teléfono de cliente.
- ❖ Nombre de producto.
- ❖ Cantidad producto solicitado.

Este registro provisorio en la agenda no registra la fecha en la cual un cliente ha reservado su producto, por ende no se sabe con certeza cuánto es la demora de espera del producto

5) Cuaderno resumen de ventas: Éste es el cuaderno donde manualmente una o dos veces por semana la vendedora del local toma las boletas y facturas para resumir las ventas diarias hechas por la empresa. Luego de esto, una o dos veces al mes el dueño del local revisa este cuaderno y transcribe estos resúmenes a una planilla Excel, los datos registrados en este resumen para cada boleta y factura emitida por la tienda son:

- ❖ Fecha emisión boleta.
- ❖ Número de primera boleta en el día.
- ❖ Número de última boleta del día.
- ❖ Total ventas por boleta en un día.
- ❖ Fecha de factura.
- ❖ Número de factura.
- ❖ Nombre de cliente asociado a factura.
- ❖ Rut de cliente asociado a factura.
- ❖ Valor neto de factura.
- ❖ Valor IVA.
- ❖ Valor total factura.

1.3. PROBLEMAS DETECTADOS

Luego de analizar la situación actual, se logran identificar una serie de problemas que impiden llevar un correcto control del stock disponible en bodega y su documentación, de las ventas realizadas, el cliente que efectuó dicha transacción y de la compra de los productos junto con su(s) proveedor(es):

A continuación se listan los problemas en forma detallada:

- ❖ No se conoce el stock actual de los productos, por lo que cada vez que se realiza una venta, el stock se debe revisar visualmente, lo que genera pérdida de tiempo.
- ❖ No existe registro alguno de los datos de los productos, por lo que si un cliente necesita un producto con alguna restricción (color, precio, dimensiones, etc.), el tiempo de respuesta es aún mayor.
- ❖ Cada vez que un cliente pide un producto que no tienen en stock, este pedido se registra en una agenda provisoria, lo que ocasiona que de vez en cuando, el nombre y pedido de un cliente se pierda entre las hojas y jamás reciba la llamada avisando la llegada del producto.

- ❖ Las ventas son registradas en un cuaderno, dificultando la búsqueda de alguna venta en específico o incluso perdiendo la información de ésta.
- ❖ Las facturas son escritas a manos, lo que ocasiona de vez en cuando, que no se entienda la letra y haya que obtener los datos del cliente otra vez, generando molestia a los compradores (que pueden ser empleados del cliente).
- ❖ Al momento de comprar los productos, el dueño de la tienda debe buscar el contacto de los proveedores, que se encuentran almacenados en los catálogos de compra o en una agenda, lo que genera pérdida de tiempo.
- ❖ Cuando se genera algún tipo de ajuste en el stock, sea por pérdida de producto o donación, no se registra en ningún lado, dejando sin evidencia el cambio en el stock.
- ❖ Falta fecha de reserva de producto.

1.4. DESCRIPCIÓN DEL SISTEMA PROPUESTO

1.4.1. Objetivo general del sistema propuesto

- ❖ Crear un sistema de información que apoye las funciones generales de venta , para así poseer administración y conocimiento de cada uno de los productos que se tiene en bodega, de tal forma que en cada venta se reduzca considerablemente el tiempo de atención al cliente para así entregar un mejor servicio, así mismo el sistema propuesto busca dar comodidad a la vendedora a la hora de extraer información de algún cliente, proveedor o producto según requiera el caso, y una vista general y detallada de las ventas realizadas en cualquier fecha para el uso del dueño de la tienda.

1.4.2. Objetivos específicos del sistema propuesto

- ❖ Mantener un control actualizado de stock
- ❖ Almacenar información detallada de clientes, proveedores y productos.
- ❖ Mantener un resumen de ventas detallando los productos trazados para su posterior revisión y análisis.
- ❖ Crear una sección o ventana donde puedan realizarse las ventas.
- ❖ Tener registro de la reserva de productos con estados como; pendiente, realizada, cancelada.

- ❖ Calcular automáticamente datos de la venta.
- ❖ Una interfaz amigable, reduciendo el ingreso de datos a la hora de ingresar una venta en el sistema.

1.4.3. Beneficios del sistema propuesto

El sistema propuesto podrá mejorar considerablemente los tiempos de respuesta para cada sección de la empresa, con la ayuda de éste, el trabajo del dueño, vendedora y bodeguero se verán favorablemente afectados con los siguientes beneficios:

- ❖ Se podrá acceder a la información de los proveedores, clientes y productos de forma más rápida y detallada.
- ❖ Ya no se tendrán los datos separados en documentaciones distintas y provisionales, sino que estará completamente detallada en una pantalla.
- ❖ Se reducirá espacio físico de almacenamiento de datos.
- ❖ Se mejorará el tiempo de respuesta ante la petición de algún producto.
- ❖ Al automatizar el cálculo de los valores totales en la venta, se reducirá considerablemente el margen de error a la hora de calcular venta de producto.
- ❖ Al simular la vista de factura, se verán claramente los datos de cliente, y ya no habrán confusiones al no entender la letra de alguna persona.
- ❖ Dar respuesta óptima del bajo stock de algún producto.
- ❖ El dueño podrá evaluar el desempeño de su empresa al contar con la lista de ventas realizadas por fecha.
- ❖ El tiempo transcurrido desde la llegada del cliente hasta que se retira con su producto será disminuido, mejorando la calidad de atención.

1.4.4. Descripción general de solución propuesta

Se propone el desarrollo de una aplicación, utilizando Java.

Se prefiere una aplicación con interfaz de ventanas desplegadas para que el usuario tenga una cierta familiaridad al utilizarlo.

Para el ingreso de los productos, la aplicación debe permitir el ingreso de los datos contenidos en la “packing list” enviada por la compañía de envío anteriormente seleccionada.

La aplicación también debe mostrar los productos con stock disponible para la venta, dando a conocer cuales productos se encuentran sin stock.

Para la venta de producto, la aplicación da a elegir si se usará boleta o factura, si es factura, se desplegará una ventana donde se rellenarán los datos necesarios y

aparecerá una sección donde se registrará el detalle; si es boleta, solo se desplegará la sección del detalle de venta, luego se desplegará una ventana con todos los datos de la venta, si está correcto, se acepta y se hacen los descuentos de stock en inventario.

Para la venta por teléfono, se usará un sistema parecido al de ventas, pero en ésta ocasión, la boleta o factura desplegada será usada para rellenar la guía de despacho para que la compañía haga el posterior envío de los productos.

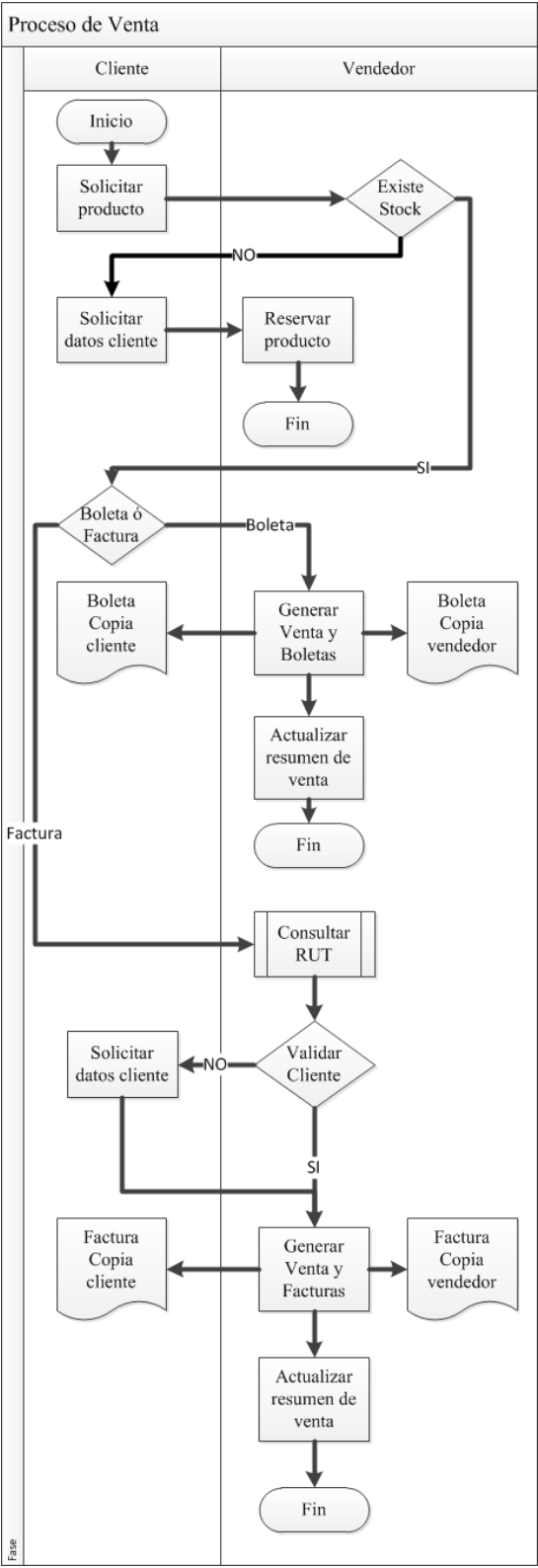
Si se produce algún ajuste en el stock de producto, por pérdida o donación, el programa debe permitir la disminución o aumento del stock.

El desarrollo de esta aplicación está pensado para simplificar el trabajo del empleado que interactúa con los clientes en las ventas y del trabajador que está encargado del stock, además, de facilitar el proceso de compra de productos y optimizar el proceso de venta.

1.4.5. Diagramas de Flujo administrativo

A continuación se presentan los diagramas de flujo administrativo de la situación actual en la empresa.

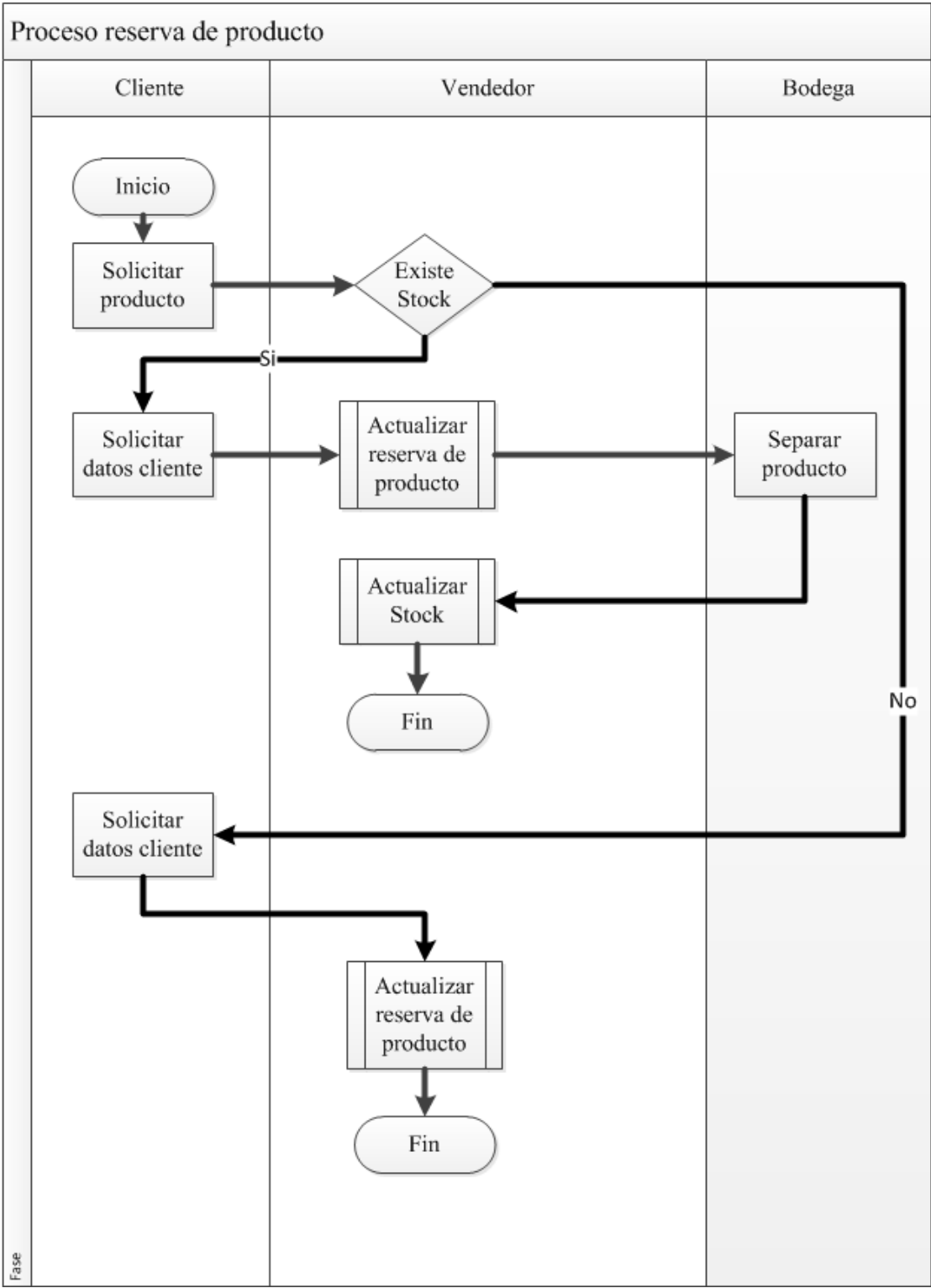
a) Proceso de venta:



Fuente propia

Diagrama 1-1. Proceso de venta

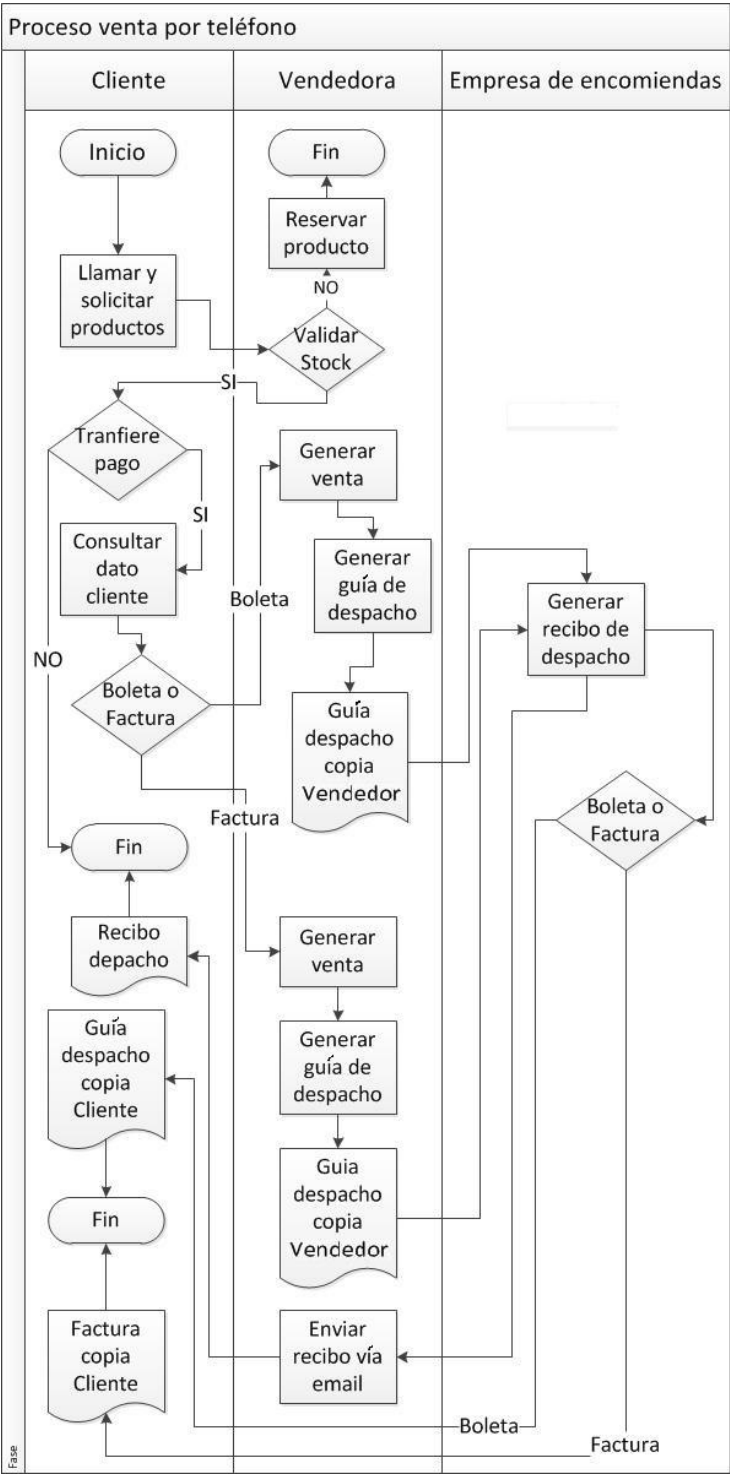
b) Proceso reserva de producto:



Fuente propia

Diagrama 1-2. Proceso de reserva de producto

c) Proceso de venta por teléfono:



Fuente propia

Diagrama 1-3. Proceso de venta por teléfono

1.4.6. Estructura funcional del sistema

❖ A continuación se mostrarán las distintas funcionalidades que realizará este sistema:

- a) Transacción de venta: Permite la transacción de una venta en cuanto es solicitada por el cliente. El vendedor selecciona la opción de boleta o factura según el caso, solicita datos de cliente, en el caso de factura, y luego detalla los productos vendidos sea boleta o factura, finalmente se registrará la venta.
- b) Mantenedor de productos: Mantiene el detalle de los productos permitiendo ingresar, modificar, consultar o eliminar algún producto del sistema cuando se requiera, la información ingresada consiste en: (código, nombre, tipo-producto, precio-venta, proveedor, stock-disponible, fecha-ingreso, foto, nota).
- c) Mantenedor de proveedor: Incluye las funciones de ingreso, modificación, consulta, y eliminación de algún proveedor con el que se cuente, los datos ingresados son: (rut, nombre, el o los teléfonos, el o los email, web-site, código-comuna, logotipo).
- d) Mantenedor de ventas con boleta: Incluye las funciones de ingreso, modificación, consulta, y eliminación de alguna venta realizada con boleta y los datos que se ingresan son: (número-boleta, fecha,(código-producto, nombre-producto, valor-unidad, cantidad) tipo-pago, cantidad-cuotas, tipo-tarjeta, valor-total, valor-total-neto)
- e) Mantenedor de ventas con factura: Incluye las funciones de ingreso, modificación, consulta, y eliminación de alguna venta realizada con factura y los datos que se ingresan son: (número-factura, fecha, rut-cliente, nombre-cliente, (código-producto, nombre-producto, valor-unidad, cantidad), tipo-pago, tipo-tarjeta, cantidad-cuotas, valor-total, valor-total-neto).
- f) Mantenedor de cliente: Tiene las funciones de ingresar, eliminar, consultar o modificar la información de algún cliente y sus datos son: (rut, nombre, el o los teléfonos, el o los email, código-comuna)
- g) Mantenedor de categoría de producto: Se puede ingresar, eliminar, consultar o modificar los datos de los tipos de productos, sus datos son: (código, tipo-producto).
- h) Mantenedor de reserva de productos: Posee las funciones de ingresar, consultar, eliminar o modificar algún pedido que se haya efectuado, sus campos son: (código-reserva, fecha-pedido, rut-cliente, estado (código-producto, nombre-producto, cantidad)).

- i) Mantenedor de ajuste de stock: Incluye las funciones de ingresar, modificar, consultar o eliminar algún ajuste de stock que se haya realizado, sus datos son: (cod_ajuste, cod-producto, tipo, fecha, motivo).
- j) Mantenedor de región: Permite modificar, eliminar o modificar los dato la tabla región, sus datos son: (código-región, nombre).
- k) Mantenedor de comuna: Permite modificar, eliminar o consultar los datos de la tabla comuna, sus campos son: (código-comuna, nombre comuna, código-región).
- l) Mantenedor de IVA: Permita modificar, eliminar o consultar el valor de IVA actual, que está almacenado en un archivo de texto
- m) Mantenedor de usuarios: Permite ingresar, consulta, modificar o eliminar a algún usuario del sistema, sus datos son: (rut, nombre, usuario, contraseña).
- n) Generadores de informe:
 - ❖ Resumen de ventas: Listado con todas las ventas realizadas con o sin cuotas y su detalle.
 - ❖ Inventario: Listado que muestra el stock de producto detallado disponible.
 - ❖ Clientes: Listado con todos los clientes y su información.
 - ❖ Proveedor: Listado con los proveedores y su respectiva información.
 - ❖ Ajuste de stock: Informe con los ajustes de stock.
 - ❖ Reserva de productos: Listado con todas las reservas (activas o no) efectuadas por los clientes.

1.4.7. Descripción de fórmulas y cálculos empleados

Al ser un sistema que gestiona las ventas realizadas por una tienda, es de vital importancia el manejo de ciertas fórmulas que ayuden a los cálculos de las transacciones. A continuación, se presenta el listado de situaciones que ameritan el uso de fórmulas en el sistema:

- ❖ **Valor subtotal de producto**: Esta fórmula se utiliza para poder conocer cuál es el precio subtotal de cada producto para una venta, la fórmula consiste en tomar el valor de venta de un producto y multiplicarlo por la cantidad ingresada por el vendedor, quedando de la siguiente forma:

$$\text{Valor total de producto} = \text{Precio de venta} * \text{cantidad.}$$

- ❖ **Valor total de productos**: Esta forma es para obtener la suma de los subtotales, en un ciclo repetitivo, esta fórmula es utilizada en el área de ventas y el área de reserva de productos, para así conocer la totalidad que implicará monetariamente la transacción. La fórmula es la siguiente:

$$Valor\ total\ de\ producto = \sum_{n=1}^{Cantidad\ productos\ seleccionados} (Valor_subtotal_de_producto_n)$$

❖ **Valor neto subtotal de producto:** Cálculo utilizado a la hora de crear únicamente la factura, es el valor real del producto sin el IVA. La fórmula para este cálculo es la siguiente:

$$Valor\ neto\ subtotal\ de\ producto = \frac{Valor\ subtotal\ de\ producto}{IVA}$$

❖ **Valor neto total de productos:** Al igual que el cálculo anterior, este es para la realización de facturas, y es el valor total de la sumatoria de los precios netos de cada producto. La fórmula para este cálculo es la siguiente:

$$Valor\ neto\ total\ de\ producto = \sum_{n=1}^{Cantidad\ productos\ seleccionados} (Valor_neto_subtotal_de_producto_n)$$

❖ **IVA:** Si bien el sistema incluye dentro de sus cálculos internos la función para calcular productos con IVA, el cual, por defecto será 19%, se permitirá que el súper usuario lo modifique en caso de que la ley de comercio o algún factor que él considere lo requiera. A la hora de ingresar el porcentaje de IVA, lo hará con un número de dos dígitos, es necesaria una pequeña fórmula para que el sistema pueda utilizar este valor. La fórmula para obtener el valor utilizable del IVA es la siguiente

$$IVA = 1 + \frac{Valor\ ingresado\ por\ usuario}{100}$$

1.4.8. Información que se manejará

1.4.8.1. Salidas del sistema

Las salidas que serán generadas automáticamente por el sistema serán las siguientes:

- a) **Listado de productos:** Incluye los datos de todos los productos con los que se cuentan en la tienda, especificando, además, de su información propia, el proveedor, una nota y una fotografía para su identificación, este listado le será de ayuda al dueño de la tienda, a la vendedora y al bodeguero para conocer los productos existentes antes de cada venta o para decidir cuándo volver a abastecer el inventario.

Los datos a mostrar son:

- ❖ Código de producto.
- ❖ Nombre del producto.
- ❖ Nombre de categoría del producto.
- ❖ Stock actual de producto.
- ❖ Stock crítico del producto.
- ❖ Nombre del o los proveedores.
- ❖ Website del o los proveedores.
- ❖ Fotografía.
- ❖ Nota.

- b) **Listado de proveedores:** Incluye los datos de los proveedores sus detalles, la utilidad de este listado es para el dueño de la tienda, ya que, es donde puede visualizar la procedencia de los productos y hacer sus pedidos de manera remota.

Los datos a mostrar son:

- ❖ Rut del proveedor.
- ❖ Nombre del proveedor.
- ❖ Ciudad del proveedor.
- ❖ Comuna del proveedor.
- ❖ Web-site del proveedor.
- ❖ E-mail(s) del proveedor.
- ❖ Número(s) de teléfono del proveedor.
- ❖ Categoría de producto que abastece el proveedor.

- c) **Listado de usuarios:** El listado de los usuarios deberá ser accesible solo para el administrador de la tienda, que es el dueño, este podrá hacer mantenimiento de esta lista según requiera

Los datos a mostrar son:

- ❖ Rut de usuario.
- ❖ Nombre de usuario.
- ❖ El usuario.
- ❖ Contraseña.

d) **Venta con boleta:** El dueño de la tienda o vendedor podrá hacer ingreso a esta lista, en la cual se mostrarán los datos de ventas por boleta, ya sea en una boleta en particular o una lista que resuma las ventas trazadas con boleta. Los datos a mostrar son:

- ❖ Número de boleta.
- ❖ Fecha.
- ❖ Detalle de la venta:
 - Código de producto.
 - Nombre de producto.
 - Cantidad de producto.
 - Precio de producto.
 - Precio total del producto.
- ❖ Total boleta
- ❖ Cuotas (Opcional):
 - Nombre del cliente.
 - Apellido del cliente.
 - Rut del cliente.
 - Nombre de la tarjeta.
 - Número de cuotas.
 - Valor de la cuota.

e) **Venta con factura:** El dueño de la tienda vendedor podrá hacer ingreso a esta lista, en esta lista se mostrarán los datos de ventas por factura, ya sea una factura particular, o el resumen de todas en una lista.

Los datos a mostrar son:

- ❖ Número de factura.
- ❖ Fecha.
- ❖ Rut de cliente.
- ❖ Nombre de cliente.
- ❖ Apellido del cliente.
- ❖ Giro del cliente.
- ❖ Región del cliente.

- ❖ Comuna del cliente.
- ❖ Dirección del cliente.
- ❖ Detalle de venta:
 - Código de producto.
 - Nombre de producto.
 - Cantidad de producto.
 - Valor neto unitario del producto.
 - Valor neto total del producto.
 - Valor unitario del producto.
 - Valor total del producto.
- ❖ Valor neto total.
- ❖ Valor total factura.
- ❖ Cuotas (Opcional):
 - Nombre de la tarjeta.
 - Valor de la cuota.
 - Cantidad de cuotas.

f) **Listado de clientes:** El listado de clientes podrá ser accedido, además del dueño, por el vendedor, será muy útil para la creación de facturas, ya que en este listado se aprecian los datos de los clientes o de uno en particular si es consultado.

Los datos a mostrar son:

- ❖ Rut cliente.
- ❖ Nombre del cliente.
- ❖ Apellido del cliente.
- ❖ Comuna del cliente.
- ❖ Región del cliente.
- ❖ Dirección de cliente.
- ❖ Giro del cliente.
- ❖ E-mail(s) del cliente.
- ❖ Teléfono(s) del cliente.

g) **Listado de categorías:** Este listado servirá para que la persona que esté ingresando productos pueda ver el código de las categorías con las que se cuenta, además, se podrán editar en caso que se requiera.

Los datos a mostrar son:

- ❖ Código de categoría.
- ❖ Categoría.

- h) **Reserva de productos:** Esta lista mostrará la lista de reservas que han sido solicitadas, cada una con su respectivo estado de (pendiente, anulada, realizada).

Los datos a mostrar son:

- ❖ Rut del cliente.
- ❖ Nombre del cliente.
- ❖ Nombre del vendedor que lo reservó.
- ❖ Rut del vendedor que lo reservó.
- ❖ Fecha.
- ❖ Estado total de la reserva.
- ❖ Detalle de reserva:
 - Código de producto,
 - Nombre de producto.
 - Cantidad de producto.
 - Estado del producto.

- i) **Listado de regiones:** Mostrará las regiones de las cuales el sistema tiene en conocimiento, la utilidad de este listado es visualizarlas.

Los datos a mostrar son:

- ❖ Código de región.
- ❖ Nombre de región.

- j) **Listado de Comunas:** Mostrará las comunas de las cuales el sistema tiene en conocimiento, la utilidad de este listado es visualizarlas.

Los datos a mostrar son:

- ❖ Código de Comuna.
- ❖ Nombre de Comuna.
- ❖ Nombre de la región a la que pertenece.

- k) **Valor de IVA:** Sólo mostrará el valor del IVA, para poder editarlo si se desea.

El dato mostrado es:

- 1) IVA.

1.4.8.2. Entradas del sistema

A continuación se presentan las entradas de datos que posee este sistema:

- a) **Productos:** Estos son los datos que poseen los productos que son adquiridos por el dueño de la tienda y que son enviados a la tienda para su posterior venta, estos además incluyen información acerca del proveedor al que fue comprado.

- ❖ Nombre del producto.

- ❖ Código del producto.
- ❖ Cantidad.
- ❖ Precio de compra.
- ❖ Fotografía.
- ❖ Nota (Opcional).
- ❖ Proveedor.
- ❖ Categoría.

b) **Proveedores:** Estos son los datos de aquellos proveedores a los cuales se le ha comprado algún producto al menos una vez:

- ❖ Rut del proveedor.
- ❖ Nombre del proveedor.
- ❖ E-mail del Proveedor.
- ❖ Teléfono del Proveedor.
- ❖ Dirección.
- ❖ Web-site.
- ❖ Comuna.

c) **Clientes:** Se incluyen los datos de los clientes que han efectuado alguna compra en la tienda y que han sido ingresados en el sistema:

- ❖ Rut del cliente.
- ❖ Nombre del cliente.
- ❖ Teléfono del cliente.
- ❖ E-mail del cliente.
- ❖ Dirección.
- ❖ Giro.
- ❖ Comuna.

d) **Usuarios:** Se incluyen los datos de los usuarios que trabajan en la tienda y que operan el sistema:

- ❖ Nombre del usuario.
- ❖ Rut del usuario.
- ❖ Usuario.
- ❖ Contraseña.

e) **Venta con boleta:** Estos serán los datos de entrada que tendrá una venta cuando esta se efectúe:

- ❖ Número de la boleta.

- ❖ Fecha.
- ❖ Medio de pago.
- ❖ Número de cuotas.
- ❖ Tarjeta (opcional).
- ❖ Detalle:
 - Código del producto.
 - Cantidad.
 - Valor unitario.

f) **Venta con factura:** Estos serán los datos de entrada que tendrá una venta cuando esta se efectué:

- ❖ Número de la factura.
- ❖ Fecha.
- ❖ Rut cliente.
- ❖ Medio de pago.
- ❖ Número de cuotas.
- ❖ Tarjeta (opcional).
- ❖ Detalle:
 - Código del producto.
 - Cantidad.
 - Valor neto unitario.

g) **Reserva:** Estos serán los datos que entrarán cuando se efectué alguna reserva:

- ❖ Rut cliente.
- ❖ Fecha.
- ❖ Rut del usuario.
- ❖ Detalle:
 - Código del producto.
 - Cantidad.

h) **Ajuste de stock:** Estos son los datos ingresados cuando se efectúa algún ajuste de stock:

- ❖ Nombre del producto.
- ❖ Fecha.
- ❖ Cantidad.
- ❖ Motivo.
- ❖ Tipo de ajuste.

i) **Categoría:** Estos son los datos requeridos para las categorías:

- ❖ Nombre de la categoría.

j) **Regiones:** Estos son los datos requeridos para las regiones:

- ❖ Nombre de la región.

- ❖ Número de la región.

k) **Comuna:** Estos son los datos requeridos de las comunas:

- ❖ Nombre de la comuna.

- ❖ Región a la que pertenece.

l) **Cuotas:** Estos son los datos requeridos de las cuotas:

- ❖ Tipo recibo.

- ❖ Código de recibo.

- ❖ Número de cuota.

- ❖ Total cuotas.

- ❖ Rut del cliente.

- ❖ Nombre de la tarjeta.

- ❖ Fecha.

m) **IVA:** Presenta el dato de entrada del IVA:

- ❖ Valor IVA en porcentaje.

1.4.8.3. Entidades de información

Las entidades de información manejadas en el sistema son los siguientes:

1) **Ajuste_Stock:** Almacena el registro de ajuste de stock, ya sea por pérdida o donación, este archivo está ligado a **producto**.

2) **Categoría:** almacena la categoría de los productos con los que se cuenta en la tienda, **Producto_proveedor** y **producto** obtienen sus categorías de este archivo.

3) **Producto:** Almacena datos de los productos con sus respectivos detalles.

4) **Boleta:** Registra las ventas por boleta.

5) **Detalle_Boleta:** Almacena datos de los productos registrados en una boleta específica, deriva de los archivos **Producto** y **Boleta**.

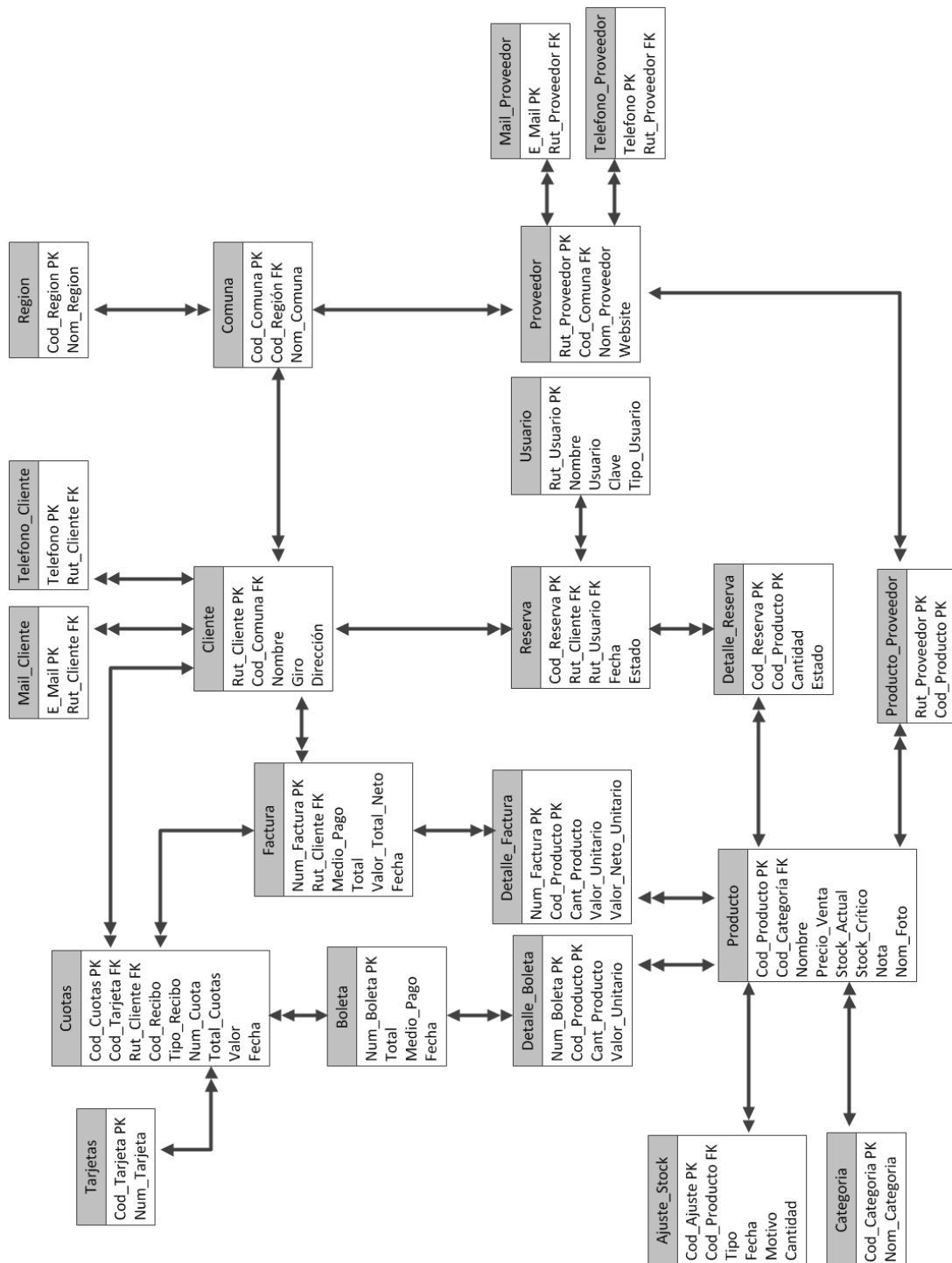
6) **Factura:** Registra las ventas por facturas y está ligada al archivo de **Cliente**.

- 7) **Detalle_Factura:** Amacena datos de los productos registrados en una boleta específica, deriva de los archivos **Producto** y **Factura**.
- 8) **Ciente:** Se registran datos de los clientes con los que la empresa cuenta.
- 9) **Mail_Cliente:** Registra los e-mails de los clientes, ya sea que tengan uno o más, el sistema permite registrarlos.
- 10) **Telefono_Cliente:** Registra los teléfonos de los clientes, ya sea cero o más, y está asociado con un cliente.
- 11) **Reserva:** Registra las reservas de productos e incluye detalle y usuario que la realizó por lo que está ligada a **Detalle_Reserva** y **Usuario**.
- 12) **Detalle_Reserva:** registra los productos asociados a una reserva en específico, por lo que se enlaza con **Reserva** y **Productos**
- 13) **Usuario:** Registra los usuarios que tienen contacto con el sistema, se otorgarán permisos según el usuario.
- 14) **Región:** Registra las regiones, y está ligado a **Comuna**.
- 15) **Comuna:** Registra las comunas que el sistema posee en conocimiento.
- 16) **Proveedor:** Registra a los proveedores con los que se cuenta.
- 17) **Mail_Proveedor:** Registra el o los e-mails de los proveedores con los que se cuenta.
- 18) **Telefono_Proveedor:** Registra el o los teléfonos de los proveedores con los que se cuenta.
- 19) **Producto_Proveedor:** Se registra la lista de productos que son abastecidos por proveedores específicos, puede ser uno o más, de modo que está ligada a **Proveedor** y **Producto**.
- 20) **Tarjetas:** Registra las tarjetas con las que se puede realizar una compra en la tienda.
- 21) **Cuotas:** se registran las cuotas que debe pagar el cliente.

A continuación se presenta el modelo de datos del sistema propuesto el cual consta de veinte y una tablas y constituido en siete secciones, las cuales son:

- **Ventas:** Tarjetas, Cuotas, Boleta, Detalle_Boleta, Factura, Detalle_Factura,
- **Productos:** Producto, Categoría, Ajsute_Stock, Categoría, Producto_Proveedor.
- **Clientes:** Cliente, Mail_Cliente, Telefono_Cliente.
- **Reservas:** Reserva, Detalle_Reserva.
- **Proveedores:** Proveedor, Mail_Proveedor, Teléfono Proveedor.
- **Usuarios:** Usuario.

- **Ciudades:** Comuna, Region.



Fuente propia.

Diagrama 1-4. Diagrama modular del sistema

1.4.9. Estructura de códigos

En este punto se mostrará como estarán estructuradas las claves primarias de las tablas utilizadas en el sistema.

En primer lugar se presentan las claves que serán correlativas:

- ❖ Código de ajuste de stock.
- ❖ Código de categoría.
- ❖ Código de reserva.
- ❖ Código de región.
- ❖ Código de comuna.
- ❖ Código de tarjeta.

A continuación las claves que se conocen:

- ❖ Código del producto, que es entregado por la fábrica.
- ❖ Número de boleta.
- ❖ Número de factura.
- ❖ Rut de cliente, que corresponde al rut del propio cliente.
- ❖ Rut de usuario, que corresponde al rut del propio usuario.
- ❖ Rut del proveedor, que corresponde al rut del propio proveedor.
- ❖ E-mail del proveedor.
- ❖ Teléfono cliente.
- ❖ Teléfono proveedor.

1.4.10. Condicionantes de diseño

El sistema está diseñado para un computador estacionario con un sistema operativo Windows XP, para la programación se utilizará el lenguaje Java y la base de datos utilizada será MySql server.

El respaldo de la información se hará mensualmente en unidades grabables de CD-ROM.

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y
DESCRIPCIÓN DE ARCHIVOS

2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2.1. DESCRIPCIÓN DEL RECURSO COMPUTACIONAL

2.1.1. Configuración del sistema

A continuación se describirá el recurso y ambiente computacional que se utilizará para el desarrollo del presente sistema.

Hardware utilizado:

1.- Acer aspire ES1-411-C45Q:

Procesador: Intel Celeron n2840 (2,58 GHz).

RAM: 2GB DDR3.

Pantalla: LED de 14”.

Almacenamiento: 320GB HDD.

Tarjeta de video: Intel HD Graphics.

2.1.2. Software utilizado

1.- **Sistema operativo:** El S.O. que se utilizará será Windows 7 de 32 bits, el cual se ejecutará de manera estacionaria en un computador local de la empresa.

2.- **Lenguajes a utilizar:** Para el desarrollo del sistema propuesto se utilizará el lenguaje Java.

- Java: Es un lenguaje de programación de propósito general (puede ser usado para diversos propósitos), concurrente (simultaneidad de ejecución de tareas) y orientado a objetos, su propósito principal es que una aplicación sea escrita una sola vez y se pueda correr en cualquier dispositivo (esto es conocido como WORA (write once, run anywhere)). Por lo que el código no necesita ser re-compilado para que se pueda ejecutar en otra máquina.

3.- **Base de datos a utilizar:** En el sistema a desarrollar se utilizará MySql para la administración de la base de datos.

- **MySQL:** Este es un sistema de administración de bases de datos, siendo una aplicación para la gestión de archivos relacionales, al ser una base de datos relacional, utiliza múltiples tablas para almacenar y organizar los datos. Fue escrito en C y C++ y goza de una gran adaptabilidad a diferentes entornos de desarrollo, por lo que puede interactuar con lenguajes de programación más utilizados.

2.2. **DESCRIPCIÓN DE ARCHIVOS**

2.2.1. **Boleta**

Nombre Lógico: boleta.
Nombre Físico: Boleta.
Descripción: Entidad que almacenará los datos de una boleta.
Clave Principal: Num_Boleta.
Clave Foránea: No presenta
Estructura de la tabla.

Tabla 2-1. Estructura archivo Boleta

Nombre	Tipo	Longitud	Descripción
Num_Boleta	Mediumint	6	Número de la boleta
Fecha	Date	8	Guarda la fecha en que se emitió la boleta (ddmmaaaa).
Total	Integer	8	Guarda el total de la venta con boleta.
MedioDePago	Char	1	Identificador del medio de pago a utilizar (1: efectivo, 2: cheque, 3: tarjeta).

Fuente: Boletas emitidas por la tienda.

2.2.2. Detalle boleta

Nombre Lógico: detalle boleta.

Nombre Físico: Detalle_boleta.

Descripción: Entidad que almacenará el detalle asociado a una boleta.

Clave Principal: Num_Boleta + Cod_Producto.

Clave Foránea: Num_Boleta que referencia a la tabla Boleta,
Cod_Producto que referencia a la tabla Producto.

Estructura de la tabla.

Tabla 2-2. Estructura archivo detalle boleta

Nombre	Tipo	Longitud	Descripción.
Num_Boleta.	Mediumint	6	Número de la boleta.
Cod_Producto.	Tinyint	3	Guarda el código de un producto.
Cant_Producto	Tinyint	4	Almacena la cantidad de producto comprado.
Valor_Unitario	Mediumint	7	Almacena el valor unitario del producto.

Fuente: Detalle de productos en bola emitida por la tienda.

2.2.3. Producto

Nombre Lógico: producto.

Nombre Físico: Producto.

Descripción: Tabla que guardará los datos de los productos.

Clave Principal: Cod_Producto.

Clave Foránea: Cod_Categoria que referencia a la tabla Categoría.

Estructura de la tabla.

Tabla 2-3. Estructura archivo producto

Nombre	Tipo	Longitud	Descripción.
Cod_Producto.	Tinyint	4	Guarda el código del producto.
Cod_Categoria	Tinyint	2	Almacena la categoría del producto.
Nombre	Char	30	Guarda el nombre del producto.
Precio_Venta	Mediumint	7	Guarda el valor unitario del producto.
Stock_Actual	Smallint	4	Almacena el stock actual del producto.
Stock_Critico	Tinyint	2	Almacena el valor mínimo que debería tener el stock del producto.
Nota	Char	50	Guarda una observación sobre el producto.
Nom_Foto	varchar	100	Almacena el nombre de la foto que está asociada al producto

Fuente: Packing list de la tienda.

2.2.4. Ajuste de stock

- Nombre Lógico:** ajuste de stock.
- Nombre Físico:** Ajuste_Stock.
- Descripción:** Tabla que almacenará los ajustes de stock realizados a los productos.
- Clave Principal:** Cod_Ajuste.
- Clave Foránea:** Cod_Producto que referencia a la tabla Producto.
- Estructura de la tabla.**

Tabla 2-4. Estructura archivo Ajuste de stock

Nombre	Tipo	Longitud	Descripción.
Cod_Ajuste	Tinyint	3	Almacena el código del ajuste
Cod_Producto	Tinyint	4	Guarda el código del producto asociado al ajuste.
Tipo	Char	1	Guarda el tipo de ajuste (1: merma, 2: donación, 3: préstamo, 4: excedente).
Fecha	Date	8	Almacena el día del ajuste (ddmmaaaa).
Motivo	Char	50	Almacena una reseña del ajuste.
Cantidad	Tinyint	4	Almacena la cantidad de algún producto afectado por ajuste.

Fuente: Información proporcionada por la tienda.

2.2.5. Categoría

Nombre Lógico: categoría.

Nombre Físico: Categoría.

Descripción: Tabla que guarda las categorías de los productos.

Clave Principal: Cod_Categoria.

Estructura de la tabla.

Tabla 2-5. Estructura archivo Categoría

Nombre	Tipo	Longitud	Descripción.
Cod_Categoria	Tinyint	2	Almacena el código de la categoría.
Nom_Categoria	Char	30	Guarda el nombre de la categoría.

Fuente: Packing list de la tienda.

2.2.6. Cliente

Nombre Lógico: cliente.

Nombre Físico: Cliente.

Descripción: Tabla que guarda los datos de cliente.

Clave Primaria: Rut_Cliente.

Clave Foránea: Cod_Comuna que referencia a la tabla Comuna.

Estructura de la tabla.

Tabla 2-6. Estructura archivo Cliente

Nombre	Tipo	Longitud	Descripcion
Rut_Cliente	Char	10	Almacena el Rut del cliente como una cadena alfanumérica de formato XXXXXXXX-X.
Cod_Comuna	Tinyint	2	Almacena el código de la comuna en que vive el cliente
Nombre	Char	30	Guarda el nombre del cliente.
Apellido	Char	30	Guarda los apellidos del cliente.
Giro	Char	30	Almacena el giro del cliente.
Dirección	Char	50	Guarda la dirección del cliente.

Fuente: Encabezado de factura.

2.2.7. Mail Cliente

Nombre Lógico: mail cliente.

Nombre Físico: Mail_cliente.

Descripción: Tabla que guarda los mails asociados a un cliente.

Clave Primaria: Email.

Clave Foránea: Rut_Cliente que referencia a la tabla Cliente.

Estructura de la tabla.

Tabla 2-7. Estructura archivo mail cliente

Nombre	Tipo	Longitud	Descripcion
Email	Char	70	Guarda un mail asociado a un cliente.
Rut_Cliente	Char	10	Guarda el Rut del cliente asociado.

Fuente: Libreta de apuntes con datos de clientes.

2.2.8. Fono Cliente

Nombre Lógico: teléfono cliente.

Nombre Físico: Telefono_Cliente.

Descripción: Tabla que almacena los fonos asociado a un cliente.

Clave Primaria: Teléfono.

Clave Foránea: Rut_Cliente que referencia a la tabla Cliente.

Estructura de la tabla.

Tabla 2-8. Estructura archivo teléfono cliente

Nombre	Tipo	Longitud	Descripción
Teléfono	Char	16	Guarda el teléfono asociado a un cliente.
Rut_Cliente	Char	10	Almacena el Rut del cliente.

Fuente: Encabezado de factura.

2.2.9. Región

Nombre Lógico: región.
Nombre Físico: Región.
Descripción: Tabla que guarda las regiones de chile.
Clave Primaria: Cod_Region.
Estructura de la tabla

Tabla 2-9. Estructura archivo Región

Nombre	Tipo	Longitud	Descripción
Cod_Region	Byte	2	Almacena el código de la región
Nom_Region	Char	30	Guarda el nombre de la región.

Fuente: Encabezado de factura.

2.2.10. Comuna

Nombre Lógico: comuna.
Nombre Físico: Comuna.
Descripción: Tabla que almacena las comunas asociadas a una región.
Clave Primaria: Cod_Comuna.
Clave Foránea: Cod_Region que referencia a la tabla Región.

Estructura de la tabla.

Tabla 2-10. Estructura archivo comuna

Nombre	Tipo	Longitud	Descripcion.
Cod_Comuna	Tinyint	2	Almacena el código de la comuna
Cod_Region	Tinyint	2	Almacena el código de región asociado.
Nom_Comuna	Char	30	Almacena el nombre de la Comuna.

Fuente: Encabezado de factura.

2.2.11. Factura

Nombre lógico: factura.

Nombre físico: Factura.

Descripción: Entidad que almacena el encabezado de la factura emitida a un cliente.

Clave principal: Num_Factura.

Clave foránea: Rut_Cliente que referencia a la tabla Cliente.

Estructura de la tabla

Tabla 2-11. Estructura archivo Factura

Nombre	Tipo	Longitud	Descripción
Num_Factura	MediumInt	6	Número de factura emitida
Rut_Cliente	Char	10	Rut de identificación del cliente
Fecha	Date	8	Fecha de emisión de la factura.
Total	Integer	8	Total de la venta
Medio_Pago	Char	1	Número que identificará el medio de pago (1: efectivo, 2: cheque, 3: tarjeta).
Valor_Total_Neto	Integer	8	valor neto total del detalle de la factura

Fuente: Encabezado de factura.

2.2.12. Detalle de factura

Nombre lógico: detalle factura.

Nombre físico: Detalle_Factura.

Descripción: Entidad que almacena el detalle de productos de la factura emitida.

Clave principal: Num_Factura + Cod_Producto.

Clave Foránea: Num_Factura que referencia a la tabla Factura. Cod_Producto que referencia a la tabla Producto.

Estructura de la tabla

Tabla 2-12. Estructura archivo detalle factura

Nombre	Tipo	Longitud	Descripción
Num_Factura	MediumInt	6	Número de factura que está ligada al detalle.
Cod_Producto	SmallInt	4	Número de producto ligado a detalle de factura
Cant_Producto	SmallInt	4	Cantidad de producto expresado en el detalle
Valor_Unitario	MediumInt	6	Valor unitario de producto en venta
Valor_Neto_Unitario	MediumInt	6	Valor Neto unitario de producto en venta

Fuente: Detalle de productos en factura emitida por la tienda.

2.2.13. Reserva

Nombre lógico: reserva.

Nombre físico: Reserva.

Descripción: Entidad que almacena la reserva de algún producto

Clave principal: Cod_Reserva.

Clave foránea: Rut_Cliente que referencia a la tabla Cliente, Rut_Usuario que referencia a la tabla Usuario.

Estructura de la tabla.

Tabla 2-13. Estructura archivo Reserva

Nombre	Tipo	Longitud	Descripción
Cod_Reserva	SmallInt	5	Número de identificación de la reserva
Rut_Cliente	Char	10	Rut del cliente que hizo la reserva
Rut_Usuario	Char	10	Rut del usuario que hizo la reserva.
Fecha	Date	8	Fecha de realización de la reserva
Estado	Char	1	Número que identificará el estado de la reserva (1: en proceso, 2: en bodega, 3: terminada).

Fuente: Libreta de apuntes con datos de clientes.

2.2.14. Detalle de reserva

Nombre lógico: detalle de reserva

Nombre físico: Detalle_Reserva.

Descripción: Entidad que almacena los productos asociado a una reserva particular.

Clave principal: Cod_Reserva + Cod_Producto.

Clave Foránea: Cod_Reserva que referencia a la tabla Reserva,
Cod_Producto que referencia a la tabla Producto.

Estructura de la tabla.

Tabla 2-14. Estructura archivo detalle reserva

Nombre	Tipo	Longitud	Descripción
Cod_Reserva	SmallInt	5	Número de reserva asociada al detalle
Cod_Producto	SmallInt	4	Número de producto asociado al detalle de la reserva
Cantidad	SmallInt	4	Cantidad de producto expresado en el detalle de la reserva.
Estado	Char	1	Número que identificará el estado del producto particular para una reserva (1: en proceso, 2: en bodega, 3: terminada).

Fuente: Libreta de apuntes con datos de clientes.

2.2.15. Usuario

Nombre lógico: usuario.

Nombre físico: Usuario.

Descripción: Entidad que almacena a los usuarios con acceso al sistema informático.

Clave principal: Rut_Usuario.

Estructura de la tabla.

Tabla 2-15. Estructura archivo Usuario

Nombre	Tipo	Longitud	Descripción
Rut_Usuario	Char	10	Rut de identificación del usuario.
Nombre	Char	30	Nombre real del usuario registrado
Usuario	Char	20	Nombre de usuario elegido para ingresar al sistema
Clave	Char	10	Clave de acceso al sistema
Tipo_Usuario	Char	1	Define permisos de usuario (1: Usuario, 2: Súper Usuario)

Fuente: Libro de asistencia de Trabajadores.

2.2.16. Proveedor

- Nombre lógico:** proveedor.
- Nombre físico:** Proveedor.
- Descripción:** Entidad que almacena a los proveedores con los cuales se cuenta.
- Clave principal:** Rut_Proveedor.
- Clave Foránea:** Cod_Comuna que referencia a la tabla Comuna.

Estructura de la tabla.

Tabla 2-16. Estructura archivo Proveedor

Nombre	Tipo	Longitud	Descripción
Rut_Proveedor	Char	10	Rut de identificación del proveedor.
Cod_Comuna	TinyInt	2	Número que identifica la comuna donde se encuentra el proveedor
Nom_Proveedor	Char	50	Nombre del proveedor
Website	Char	30	Página web de contacto del proveedor.

Fuente: Factura de compra a los proveedores.

2.2.17. Mails de proveedor

Nombre lógico: Mail Del Proveedor.
Nombre físico: Mail_Proveedor.
Descripción: Almacena uno o más e-mails que pueda tener un proveedor
Clave principal: Email.
Clave Foránea: Rut_Proveedor que referencia a la tabla Proveedor.

Estructura de la tabla.

Tabla 2-17. Estructura archivo Mail del Proveedor

Nombre	Tipo	Longitud	Descripción
Email	Char	50	Dirección de correo electrónico del proveedor
Rut_Proveedor	Char	10	Rut de identificación del proveedor dueño de la o las direcciones electrónicas.

Fuente: Factura de compra a los proveedores.

2.2.18. Teléfonos de proveedor

Nombre lógico: Teléfono de Proveedor.
Nombre físico: Teléfono_Proveedor.
Descripción: Almacena uno o más números telefónicos que pueda tener un proveedor
Clave principal: Teléfono.
Clave Foránea: Rut_Proveedor que referencia a la tabla Proveedor.

Estructura de la tabla.

Tabla 2-18. Estructura archivo teléfono de proveedor

Nombre	Tipo	Longitud	Descripción
Telefono	Char	16	Número telefónico del proveedor
Rut_Proveedor	Char	10	Rut de identificación del proveedor dueño del o los números telefónicos

Fuente: Factura de compra a los proveedores.

2.2.19. Producto de proveedor

Nombre lógico: producto de proveedor.

Nombre físico: Producto_Proveedor.

Descripción: Almacena los productos que son proporcionados por los proveedores.

Clave principal: Rut_Proveedor + Cod_Producto.

Clave Foránea: Rut_Proveedor que referencia a la tabla Proveedor. Cod_Producto que referencia a la tabla Producto.

Estructura de la tabla.

Tabla 2-19. Estructura archivo producto de proveedor

Nombre	Tipo	Longitud	Descripción
Rut_Proveedor	Char	10	Rut de identificación del proveedor del o los productos
Cod_Producto	SmallInt	4	Número de producto ligado a uno los proveedores

Fuente: Packing list de la tienda.

2.2.20. Tarjetas

Nombre Lógico: tarjetas.

Nombre Físico: Tarjetas.

Descripción: Entidad que almacenará las tarjetas de crédito con las que es posible realizar una compra.

Clave Primaria: Cod_Tarjeta.

Estructura de la tabla.

Tabla 2-20. Estructura archivo Tarjetas

Nombre	Tipo	Longitud	Descripción
Cod_Tarjeta	Tinyint	2	Almacena el código de la tarjeta
Nombre	Char	20	Almacena el nombre de la tarjeta

Fuente: Información proporcionada por la tienda.

2.2.21. Cuotas

Nombre Lógico: cuotas.

Nombre Físico: Cuotas.

Descripción: Tabla que guardará las cuotas asociadas a una boleta o factura.

Clave Primaria: Cod_Cuota.

Clave Foránea: Cod_Tarjeta que referencia a tarjetas, rut_Cliente que referencia al cliente, Cod_Recibo, esta es una referencia lógica y referencia la boleta o factura según corresponda el caso.

Estructura de la tabla.

Tabla 2-21. Estructura archivo cuotas

Nombre	Tipo	Longitud	Descripción
Cod_Cuota	Integer	5	Almacena el código de las cuotas.
Tipo_Recibo	Char	1	Almacena el tipo de recibo (B: boleta, F: factura).
Num_Cuota	Tinyint	2	Almacena el número de cuota a pagar.
Total_Cuotas	Tinyint	2	Almacena el total de cuotas a pagar.
Valor	Integer	6	Guardará el valor de la cuota a pagar
Fecha	Date	8	Almacena la fecha en que se pagó la factura
Cod_recibo	Integer	6	Almacena el código de recibo(boleta o factura) a la que está asociada
Rut_Cliente	varchar	10	Almacena el Rut del cliente a la que está asociada la cuota.
Valor_Cuota	Integer	6	Almacena el valor de cada cuota asociada al pago de una boleta o factura.

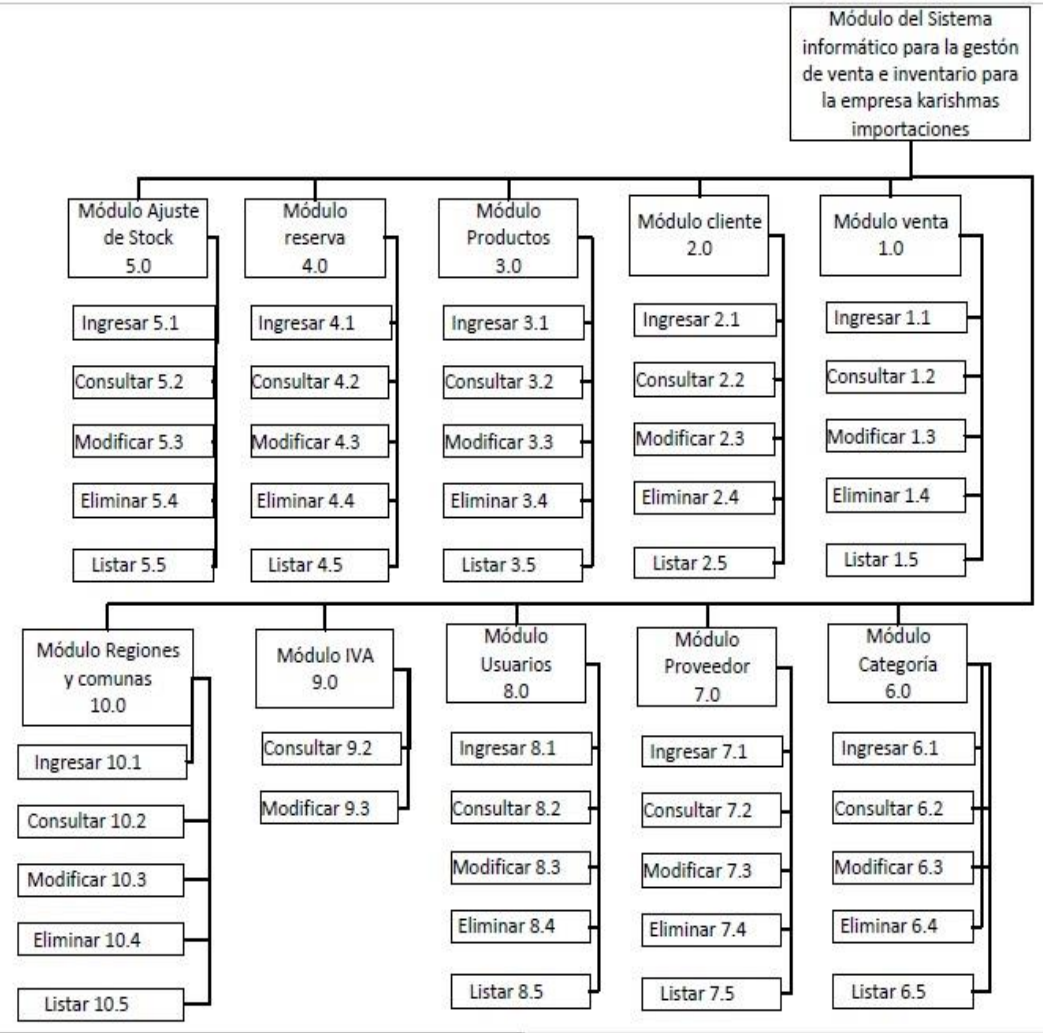
Fuente: Información proporcionada por la tienda.

CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCIÓN DE PROGRAMAS

3.1. DIAGRAMA MODULAR

A continuación se muestra el diagrama modular, el cual muestra el objetivo de cada uno dentro de la aplicación.



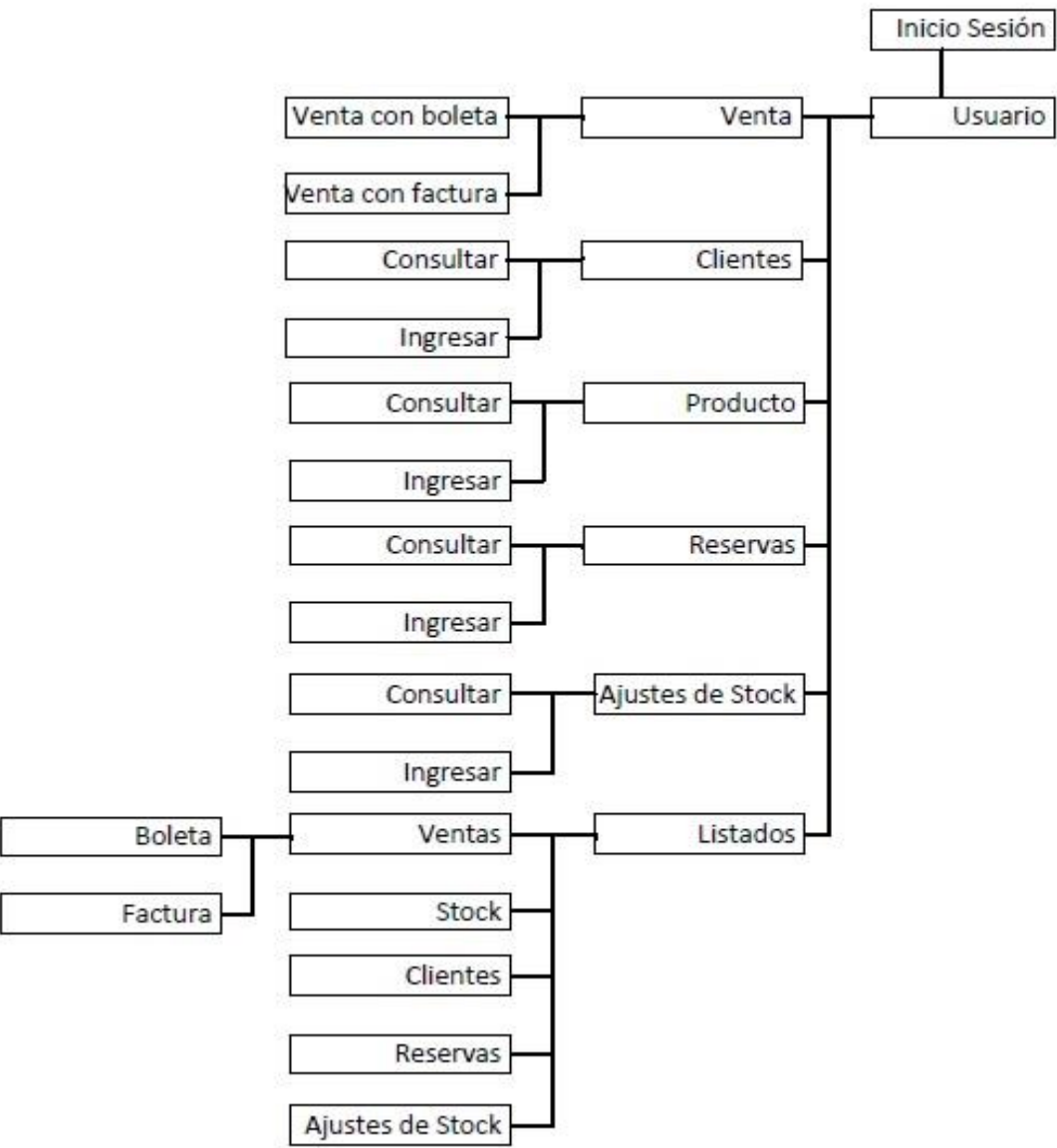
Fuente Propia

Diagrama 3-1. Diagrama modular del sistema propuesto

3.2. DIAGRAMA DE MENÚS

A continuación se presentan los diagramas de menús que permiten reflejar gráficamente la navegación dentro del software según el tipo de usuario que lo utilice.

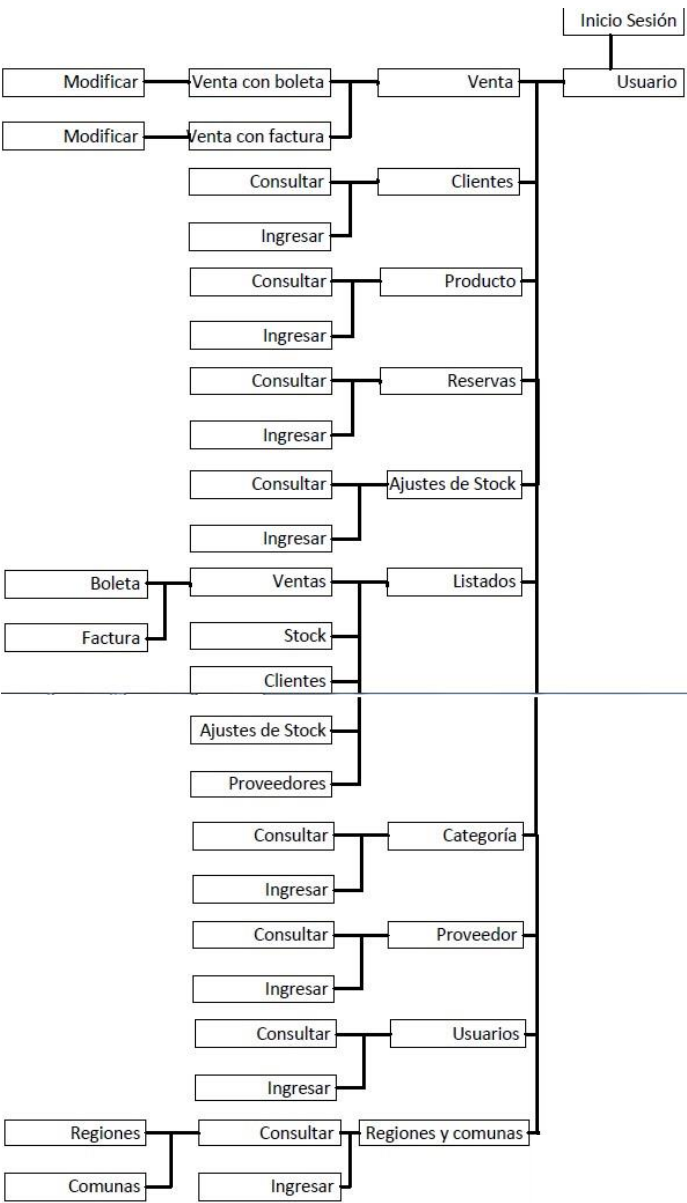
a) **Usuario:**



Fuente Propia

Diagrama 3-2. Despliegue de menú para el nivel usuario

b) Súper usuario:



Fuente propia

Diagrama 3-3. Despliegue de menú para el nivel Súper usuario

3.3. LISTA DE PROGRAMAS

A continuación se muestra una tabla con cuarenta y cuatro programas, cada uno de ellos con su objetivo explicado. Estos programas representan la estructura total del sistema informático que ha sido explicado a lo largo de la memoria.

Doce de estos programas tendrán un asterisco (*) junto a su nombre para indicar que serán explicados con detalle en el punto siguiente.

Tabla 3-1. Lista de programas del sistema propuesto

Nombre	objetivo
Venta con Boleta *	Registrar una venta de productos con la vista de una boleta.
Venta con Factura *	Registrar una venta de productos con la vista de una factura.
Consultar Cliente *	Muestra la lista de clientes y busca alguno específico.
Ingresar Cliente *	Ingresa un nuevo cliente al sistema.
Modificar Cliente *	Modifica un cliente existente.
Eliminar Cliente *	Elimina un cliente ingresado previamente en el sistema.
Consultar Producto *	Muestra la lista de productos y busca alguno específico.
Ingresar Producto *	Ingresa un nuevo producto al sistema.
Modificar Producto *	Modifica un producto existente.
Eliminar Producto *	Elimina un producto ingresado previamente en el sistema.
Ingresar Reserva.*	Ingresa una nueva reserva al sistema.
Ingresar Cuotas.*	Asociar cuotas a un recibo.
Listado Venta Con Boleta.	Lista las ventas con boleta y permite buscar alguna específica.
Listado Venta Con Factura.	Lista las ventas con factura y permite buscar alguna específica.
Modificar IVA	Modifica el valor del IVA.
Cerrar Sesión.	Cierra el Sistema.

Tabla 3-1. Lista de programas del sistema propuesto (continuación)

Nombre	objetivo
Cambio de Clave.	Modifica la clave de acceso para algún usuario particular
Consultar Reserva.	Muestra la lista de reservas y busca alguna en específico.
Inicio de Sesión.	Restringe el ingreso al sistema.
Eliminar Reserva.	Elimina una reserva ingresada previamente en el sistema.
Consultar Ajuste de Stock.	Muestra la lista de ajustes de stock y busca alguno específico.
Ingresar Ajuste de Stock.	Ingresa un nuevo ajuste de stock al sistema.
Modificar Ajuste de Stock.	Modifica un ajuste de stock existente.
Eliminar Ajuste de Stock.	Elimina un ajuste de stock ingresado previamente en el sistema.
Consultar Categoría.	Muestra la lista de categorías y busca alguna en específico.
Ingresar Categoría.	Ingresa una nueva categoría al sistema.
Modificar Categoría.	Modifica una categoría existente.
Eliminar Categoría.	Elimina una categoría ingresada previamente en el sistema.
Consultar Proveedor.	Muestra la lista de proveedores y busca alguno específico.
Ingresar Proveedor.	Ingresa un nuevo proveedor al sistema.
Modificar Proveedor.	Modifica un proveedor existente.
Eliminar Proveedor.	Elimina un proveedor ingresado previamente en el sistema.
Consultar Usuario.	Muestra la lista de usuarios y busca alguno específico.
Ingresar Usuario.	Ingresa un nuevo usuario al sistema.
Modificar Usuario.	Modifica un usuario existente.
Eliminar Usuario.	Elimina un usuario ingresado previamente en el sistema.

Tabla 3-1. Lista de programas del sistema propuesto (continuación)

Nombre	objetivo
Consultar Región.	Muestra la lista de regiones y busca alguna en específico.
Ingresar Región.	Ingresa una nueva región al sistema.
Modificar Región.	Modifica una región existente.
Eliminar Región.	Elimina una región ingresada previamente en el sistema.
Consultar Comuna.	Muestra la lista de comunas y busca alguna en específico.
Ingresar Comuna.	Ingresa una nueva comuna al sistema.
Modificar Comuna.	Modifica una comuna existente.
Eliminar Comuna.	Elimina una comuna ingresada previamente en el sistema.

Fuente: Menú del sistema realizado.

3.4. DESCRIPCIÓN DE PROGRAMAS SEÑALADOS

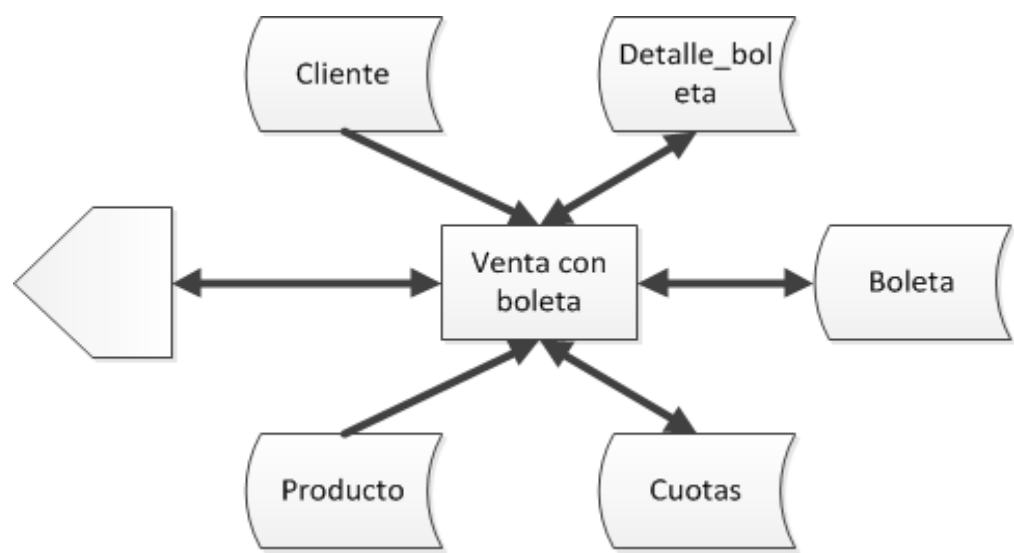
A continuación se presenta la explicación a detallada de cada uno de los programas señalados con un asterisco (*) en el punto 3.3, incluyendo:

- ❖ Nombre.
- ❖ Objetivo.
- ❖ Diagrama de bloques.
- ❖ Reglas de proceso detallado en modo narrado.
- ❖ Diseño de pantalla.
- ❖ Anexo incluyendo número de página en la cual se encuentra el código fuente.

3.4.1. Venta con boleta

- a) **Nombre:** Venta con boleta.
- b) **Objetivo:** Realizar y registrar la venta de un producto, con una vista de boleta.

c) Diagrama de bloques:



Fuente propia.

Diagrama 3-4. Diagrama de bloques venta con boleto

d) Procedimiento:

Se ingresa el número de la boleto manualmente en la etiqueta indicada, enseguida se llena el detalle de la boleto llenando los campos; código de producto, Cantidad de producto y valor unitario.

Se presiona la tecla ENTER y el sistema valida; que se hayan ingresado sólo datos numéricos, que se hayan llenado los campos indicados anteriormente y que el valor de venta del producto sea mayor que el valor de compra.

El sistema calcula el valor total unitario para cada fila del detalle según la cantidad y valor del producto vendido y el valor total de la boleto.

Se procede a elegir pago con tarjeta o efectivo.

Solamente si se elige pago con tarjeta:

Se mostrará una sección para asignar cuotas a un cliente, Solicitando los primeros ocho dígitos del rut, se ha de presionar tecla ENTER para que el sistema corrobore el dígito verificador del número en cuestión, sólo si éste tiene 8 dígitos, si está correcto, se recurre al archivo cliente y su nombre aparecerá automáticamente.

Se elige un número de cuotas y se presiona ENTER, si el valor total de la boleto es mayor a cero, el sistema calculará el valor de las cuotas, sino el sistema solicitará ingresar al menos un producto en el detalle y se procesara la venta.

Si no se elige la opción de pago con tarjeta, se puede proceder a registrar la venta en el sistema y antes de que el sistema escriba el archivo boleto y detalle_boleta validará que el detalle de boleto tenga datos, la boleto tenga un número.

Cabe destacar que al ser una boleto, ésta no requiere algún cliente asociado pero si se elige la opción de pago con tarjeta, esta necesita algún cliente para asociar las cuotas y el sistema validará que el cliente exista y estén poblados los campos: rut del cliente, dígito verificador del rut del cliente, cantidad de cuotas asociadas a la venta y valor de cada cuota.

e) **Diseño de pantalla.**

☐ Pago con tarjeta

VENTA CON BOLETA
16/03/2016

BOLETA
N° : 5644...

Rut : 19171200 — 5

Nombre Cliente Jorge

Nombre tarjeta VISA

Cantidad cuotas2

Valor cuota : 65.000

Agregar línea al detalle

Quitar línea al detalle

Codigo producto	Nombre del Pro...	Cantidad producto	Valor unitario	Valor total unitario
5112	Maniqu niño*	3	30.000	90.000
1234	PERCHEROS*	20	2.000	40.000

Nueva Venta

TOTAL : 130.000

Guardar venta

Fuente propia

Figura 3-1. Vista de venta con boleto

Anexo: el código fuente que guarda la venta con boleto se encuentra en la página 77.

3.4.2. Venta con factura

- a) **Nombre:** Venta con factura.
- b) **Objetivo:** Realizar y registrar la venta de un producto, con una vista de factura.
- c) **Diagrama de bloques:**

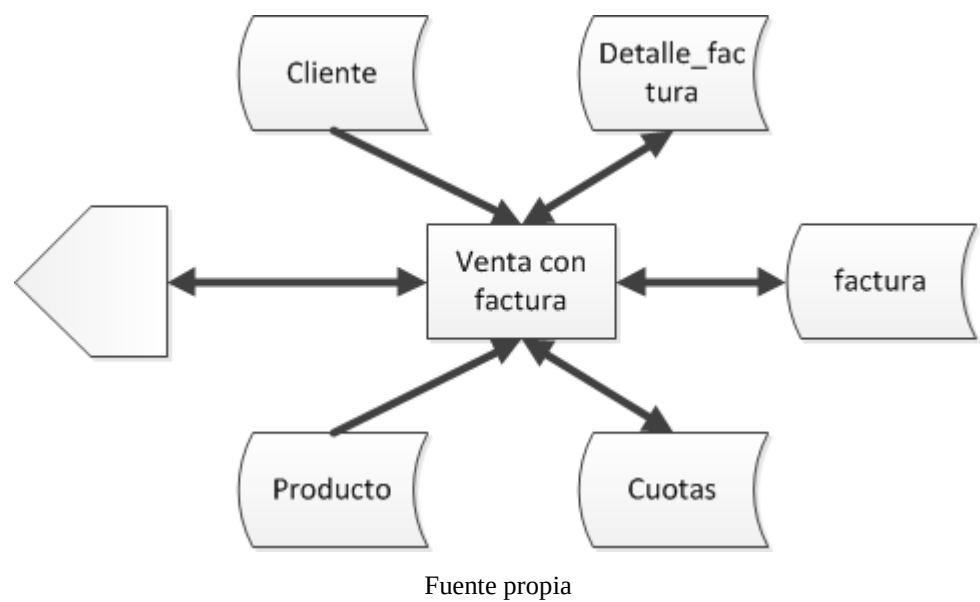


Diagrama 3-5. Diagrama de bloques venta con factura

d) **Procedimiento:**

Se debe ingresar manualmente el número de la factura en la etiqueta indicada en la vista, en el encabezado de factura se solicita el número de rut del cliente de ocho dígitos, presionar la tecla ENTER para que el sistema corrobore el rut y complete el dígito verificador, si el rut existe en el sistema para algún cliente; los datos de éste se llenarán automáticamente al mismo tiempo que el dígito verificador, si no existe, el sistema preguntará si desea ingresar el cliente como nuevo.

Para el detalle de factura, se requiere escribir: código de producto, cantidad de producto y valor unitario, se ha de presionar la tecla ENTER y el sistema valida que estos tres campos estén poblados, y además, que contengan únicamente valores numéricos.

El sistema automáticamente completará; nombre del producto, valor total unitario, valor total neto unitario y valor neto unitario, se sumará el valor total neto de cada producto al valor total neto de la factura y el valor total de cada producto se sumará al valor total de la factura.

Si se decide pagar con tarjeta, se debe elegir la opción de pago con tarjeta y se desplegarán las opciones que permiten esta funcionalidad. Se selecciona una tarjeta de la lista, se ingresa un número de cuotas y se presiona la tecla ENTER para que el sistema calcule el valor de cada una de ellas sólo si el campo TOTAL es distinto de 0.

Para guardar la venta en el sistema, se debe presionar el botón Guardar venta y el sistema validará que estén correctamente ingresados: número de factura de seis dígitos, rut de ocho dígitos, exista el dígito verificador, el cliente exista en el sistema, el total de la factura sea mayor a cero.

Sólo si se elige la opción de pago con tarjeta, se valida; cantidad de cuotas y valor de la cuota.

Si los datos fueron bien ingresados, el sistema almacenará el detalle de la factura en el archivo Detalle_Factura, el encabezado de la factura en el archivo Factura y si se pagó en cuotas; se almacenarán los datos de las cuotas en el archivo Cuotas asociado a la factura.

e) Diseño de pantalla.

☒ Pago con tarjeta

VENTA CON FACTURA

FACTURA N° :1234

Rut : 19171200

5

Nombre : Jorge

Apellido : González

Dirección : Calle falsa 123

Fecha : 20/03/2016

Giro : Compra/venta

Comuna VALPARAÍSO

región : VALPARAÍSO

Nombre tarjeta

VISA

Cantidad cuotas2

Valor cuota : 36000.0

Agregar línea al detalle

Quitar línea al detalle

Código	Prod	Cant	Neto Unit	Total Neto ...	Val Unit	Total Val U...
5321	GANCHO ...	6	2.790	16.740	3.000	18.000
5453	FLORERO*	9	5.580	50.220	6.000	54.000

TOTAL NETO : 66.960

TOTAL : 72.000

Nueva Venta

Guardar venta

Fuente propia

Figura 3-2. Vista de venta con factura

Anexo: el código fuente que guarda la venta con factura se encuentra en la página 83

3.4.2. Ingresar producto

- a) **Nombre:** Ingresar Producto.
- b) **Objetivo:** Ingresa un nuevo producto al sistema.
- c) **Diagrama de bloques.**

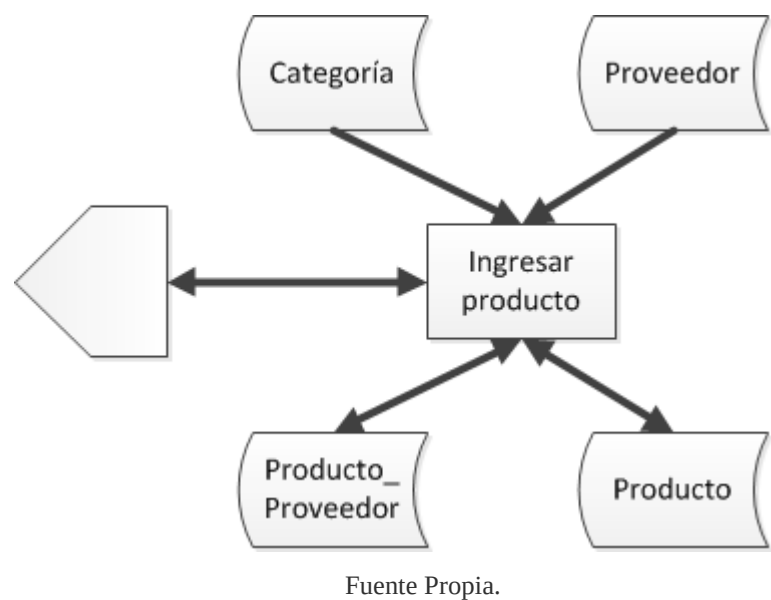


Diagrama 3-6. Diagrama de bloques Ingresar producto

d) **Procedimiento:**

Se debe ingresar: código del producto, nombre del producto, cantidad, precio de compra, stock crítico y la nota sólo si se desea. Y seleccionar; uno de los proveedores, una categoría y una fotografía.

Al presionar el botón Guardar el sistema validará: código de cuatro dígitos, campo nombre poblado, cantidad, stock crítico, que cantidad sea mayor a stock crítico, el precio de compra no puede ser cero y que efectivamente se haya elegido una fotografía.

Si los datos fueron ingresados correctamente el sistema agregará un nuevo registro al archivo Producto y al archivo Producto_Proveedor,

Una vez hecho esto, se puede elegir otro proveedor de la lista y volver a presionar el botón Guardar, el sistema dará aviso de que ya existe el producto, pero asociará un nuevo proveedor al mismo y agregará otro registro al archivo Producto_Proveedor.

e) Diseño de Pantalla.

INGRESAR PRODUCTO

Nombre : Silla metálica...

Código : 9876...

Cantidad : 300...

Categorí... SILLAS

Precio
Compra 2500...

Proveed... DEMO

Stock

Crítico : 30...

Nota :

Silla de aluminio cromada

Nuevo

Cargar Foto



descarga (3).jpg

Guardar

Fuente Propia.

Figura 3-3. Vista de ingresar Producto

Anexo: el código fuente que guarda el producto se encuentra en la página 90.

3.4.3. Consultar Producto

- a) **Nombre:** Consultar Producto
- b) **Objetivo:** Muestra la lista de productos y busca alguno específico.

c) Diagrama de bloques.

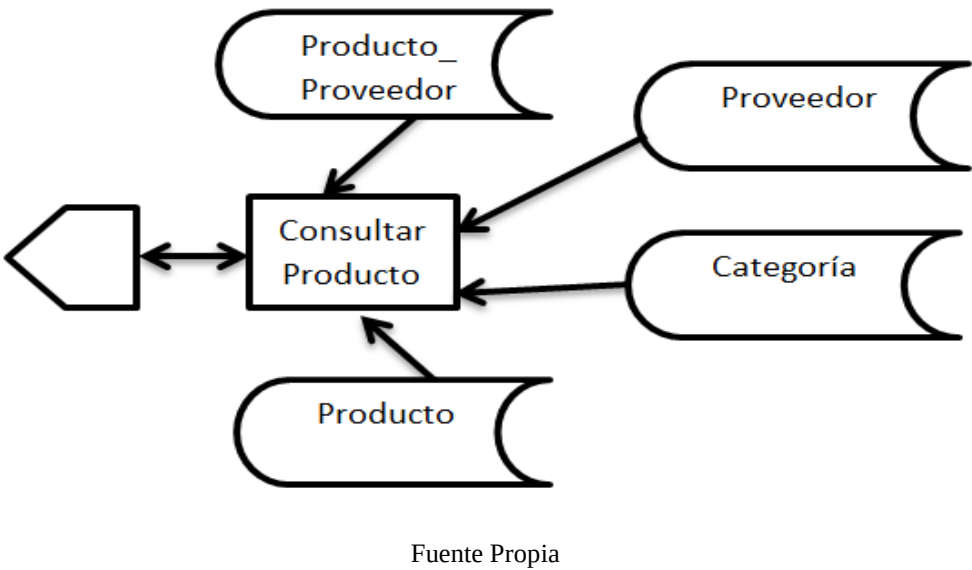


Diagrama 3-7. Diagrama de bloques Consultar Producto

d) Procedimiento:

Únicamente consultar por uno o más productos, se debe escribir en el cuadro de texto junto al botón Buscar alguna palabra que esté asociada al producto, ya sea, Nombre del producto, nombre del proveedor, categoría. Se presiona el botón Buscar y el sistema listará todos los productos que estén asociados a esta palabra, Si la palabra no está asociada a ningún producto, entonces no listará nada.

e) Diseño de Pantalla.

CONSULTAR PRODUCTO

☐ Eliminar

☐ Modificar

Nombre : FLORERO

Código : 5453

Cantidad : 500

Categori... COLGADOR

Precio Compra : 800

Proveed... DEMO

Stock Crítico : 20

Nota : hecho de madera de roble

Buscar

Código	Nomb...	Categ...	Precio...	Stock	Stock...	Nota	foto
1234	PERC...	COLG...	1500	50	20	Perch...	desca...
4677	colga...	COLG...	6000	39	20	aguan...	colga...
5453	FLOR...	COLG...	800	500	20	hecho...	florero...
4444	CAJA ...	ELEC...	150000	20	5	Caja r...	desca...
5266	ganch...	GANC...	25	700	99	metal ...	gancg...
5321	GANC...	GANC...	12	200	99	mide ...	ganch...
4588	Maniq...	MANI...	2300	36	25	compl...	maniq...
4589	Maniq...	MANI...	2300	34	25	es de ...	maniq...



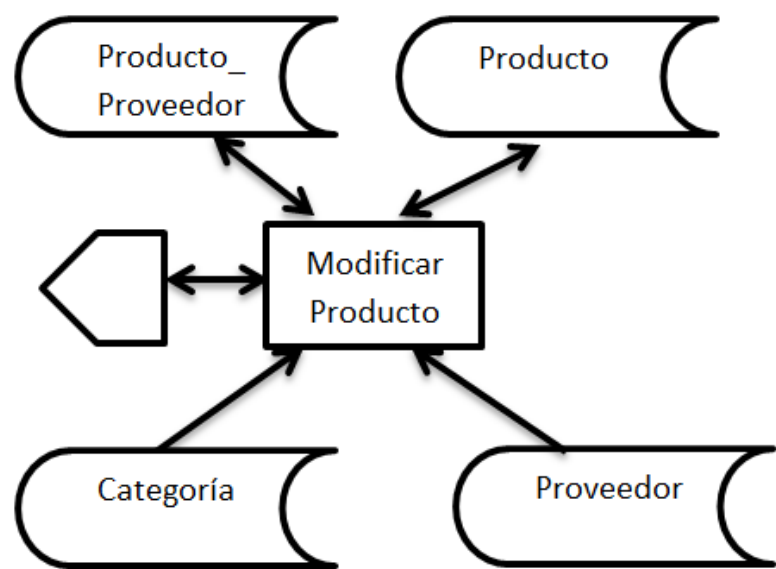
Fuente Propia.

Figura 3-4. Vista Consultar Producto

Anexo: el código fuente que busca algún producto se encuentra en la página 93.

3.4.4. Modificar Producto

- a) **Nombre:** Modificar Producto.
- b) **Objetivo:** Modificar un producto existente.
- c) **Diagrama de bloques:**



Fuente Propia.

Diagrama 3-8. Diagrama de bloques Modificar Producto

d) Procedimiento:

Se puede modificar cualquier valor asociado al producto exceptuando el código, al presionar el botón Guardar el sistema validará únicamente que no haya ningún campo vacío.

Agregar al stock: En la etiqueta de suma que aparece en la pantalla, se ingresa un número seguido de la tecla ENTER, automáticamente el sistema sobre escribirá el archivo Producto con el nuevo stock.

Agregar un proveedor: Se elige un proveedor de la sección que ha aparecido de los proveedores y se presiona el botón Agregar, si el proveedor no ha sido asociado a éste producto, el sistema agregará un nuevo registro al archivo Producto_Proveedor.

Desasociar un proveedor: Elegir un proveedor asociado únicamente al producto y presionar el botón Eliminar existente, automáticamente el sistema eliminará el registro en el archivo Producto_Proveedor que asocia el proveedor al producto.

e) Diseño de pantalla:

CONSULTAR PRODUCTO

☐ Eliminar

☒ Modificar

Nombre : CAJA REGISTRADORA

Código : 4444

Cantidad : 20

Categorí... ELECTRONICA

Precio Compra : 150000

Proveed... DEMO

Stock Crítico : 5

Nota : Caja registradora con pantalla lcd

Proveedores DEMO

Agregar

Eliminar Existente

Categorías COLGADOR

Cambiar

Buscar

Código	Nomb...	Categ...	Precio...	Stock	Stock ...	Nota	foto
1234	PERC...	COLG...	1500	50	20	Perch...	desca...
4677	colga...	COLG...	6000	39	20	aguan...	colga...
5453	FLOR...	COLG...	800	500	20	hecho...	florero...
4444	CAJA ...	ELEC...	150000	20	5	Caja r...	desca...
5266	ganch...	GANC...	25	700	99	metal ...	gancg...
5321	GANC...	GANC...	12	200	99	mide ...	ganch...
4588	Maniq...	MANI...	2300	36	25	compl...	maniq...
4589	Maniq...	MANI...	2300	31	25	es de	maniq...

Guardar



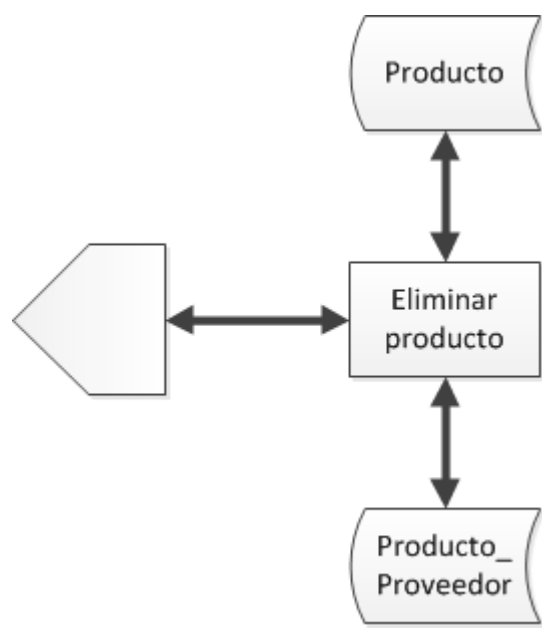
Fuente Propia

Figura 3-5. Vista modificar producto

Anexo: el código fuente Modifica el producto se encuentra en la página 93.

3.4.5. Eliminar producto

- a) **Nombre:** Eliminar Producto,
- b) **Objetivo:** Elimina un producto ingresado previamente en el sistema.
- c) **Diagrama de bloques:**



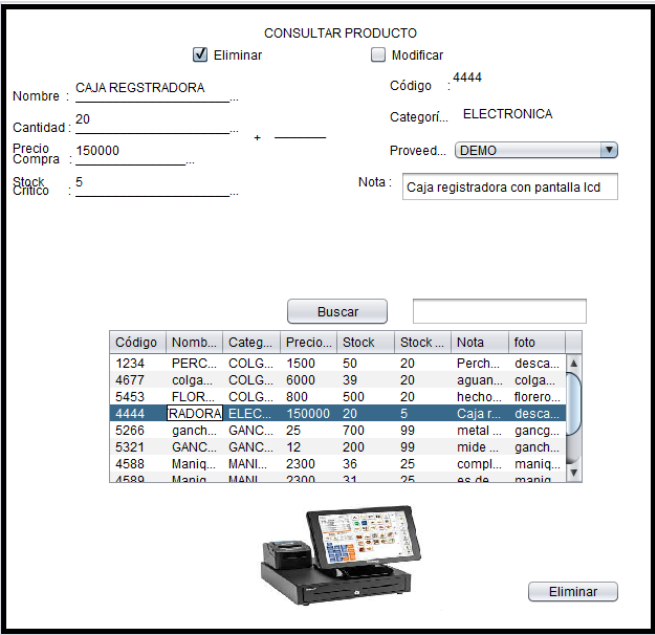
Fuente Propia.

Diagrama 3-9. Diagrama de bloques Eliminar Producto

d) Procedimiento.

Se busca un producto a través del cuadro de texto junto al botón Buscar o se selecciona manualmente haciendo clic en algún elemento de la lista de productos. Seguido a esto se presiona el botón Eliminar y luego de que el sistema muestre un aviso de advertencia, eliminará automáticamente los registros asociados a este producto del archivo Producto y todos los registros del archivo Producto_Proveedor que asocian uno o más proveedores a este producto, si alguna tabla hace referencia a este producto el proceso de eliminación se cancela.

e) **Diseño de pantalla.**



Fuente Propia.

Figura 3-6. Vista eliminar producto

Anexo: el código fuente que elimina un producto se encuentra en la página 97.

3.4.6. Ingresar Cliente

- a) **Nombre:** Ingresar cliente.
- b) **Objetivo:** Ingresar un nuevo cliente al sistema.
- c) **Diagrama de bloques.**

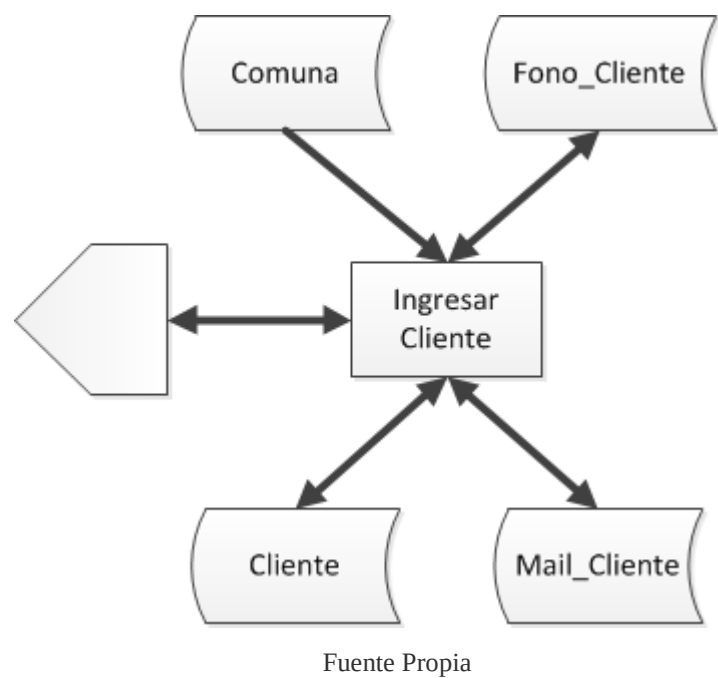


Diagrama 3-10. Diagrama de bloques Ingresar Cliente

d) Procedimiento:

Se ingresa como datos; nombre de cliente, apellido, giro, dirección, rut de ocho dígitos y al presionar la tecla ENTER el sistema generará un dígito verificador para el rut que se está ingresando. Se selecciona una región y una comuna.

Al presionar el botón Guardar, el sistema validará que los campos, nombre, apellido, rut, dígito verificador, giro y dirección estén poblados para proceder a agregar un nuevo registro de cliente al archivo Cliente.

Validará que el teléfono tenga como mínimo once dígitos, y de ser así, agregará un nuevo registro en el archivo Fono_Cliente sólo si no existe previamente , de no ser así, el sistema dará aviso de que el teléfono no es correcto, pero se puede agregar posteriormente, al igual que el ingreso de más números de teléfonos asociados al cliente

Validará que el E-mail: Tenga datos antes del arroba y ningún arroba, que la extensión tenga un punto en los 5 últimos caracteres, si está correcto, agregará un nuevo registro en el archivo Mail_Cliente sólo si no existe previamente, sino, el sistema dará aviso de que el E-mail no es correcto, pero se puede agregar posteriormente, al igual que el ingreso de más mails asociados al cliente.

Se puede seguir agregando datos de contacto al cliente en la misma pantalla, editándolo y presionando en el botón Guardar, si los datos de contacto logran pasar la prueba de validación, el sistema dará un mensaje de que el cliente ya existe, pero que se le ha asociado un nuevo número telefónico o un nuevo E-mail.

e) **Diseño de pantalla.**

INGRESAR CLIENTE

Nombre :Homero

Apellido :Simpson

Rut : 19200171

Giro :Seguridad

Teléfono : + 56912354378

Comuna :CALAMA

Región : ANTOFAG...

Dirección :Avenida Siempre viva 742...

e-mail :camidalover@hotmail.com

Nuevo

Guardar

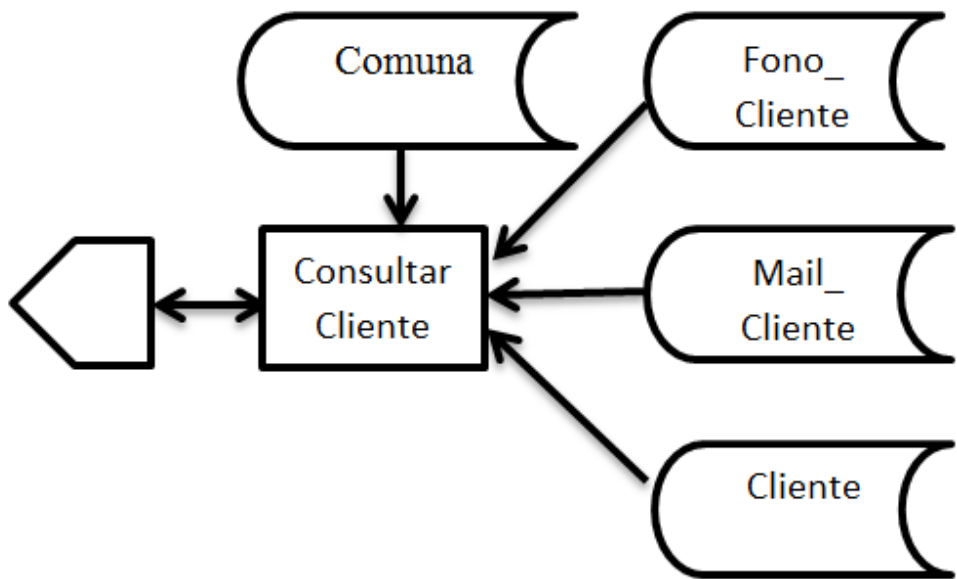
Fuente Propia

Figura 3-7. Vista ingresar cliente

Anexo: el código fuente que ingresa un cliente se encuentra en la página 98.

3.4.7. Consultar Cliente

- a) **Nombre:** Consultar Cliente.
- b) **Objetivo:** Muestra la lista de clientes y busca alguno específico.
- c) **Diagrama de bloques.**



Fuente propia.

Diagrama 3-11. Diagrama de bloques Consultar Cliente

d) Procedimiento:

Únicamente consultar por uno o más clientes, se debe escribir en el cuadro de texto junto al botón Buscar alguna palabra que esté asociada al producto, ya sea, Nombre del Cliente, apellido, ciudad o comuna, rut. Se presiona el botón Buscar y el sistema listará todos los clientes que estén asociados a esta palabra, Si la palabra no está asociada a ningún cliente, entonces no listará nada.

e) Diseño de pantalla.

☒ Modificar
☐ Eliminar

CONSULTAR CLIENTE

Rut : 19171200-5Comuna ANTOFAGASTA

Nombre JorgeRegión ANTOFAGASTA

Apellido GonzálezDirección : Calle falsa 123

Giro : Compra/venta

Nueva Comuna ANTOFAGASTANueva Región ANTOFAGASTA

Teléfonos : + 56996196717AgregarBorrar

e-mails : Jorgegonzalez@gmail.comAgregarBorrar

Palabra Clave

Buscar

Rut	Nombre	Apellido	Giro	Región	Comuna	Dirección
139058...	Kevin	Suarez	ventas	ANTOF...	ANTOFA...	calle fal...
191712...	Jorge	Gonzalez	Compra...	VALPAR...	VALPAR...	Calle fal...
191919...	Kevin	Perez	Venta R...	ANTOF...	ANTOFA...	los Sau...
233213...	Laura	Fuentes	Revend...	VALPAR...	VILLA A...	cale ma...
330477...	Debora	Suarez	Mall	ANTOF...	SIERRA...	Calle lo...
616161	jorge	Brito	feriante	ANTOF	ANTOFA	pasaie t

Guardar

Fuente Propia.

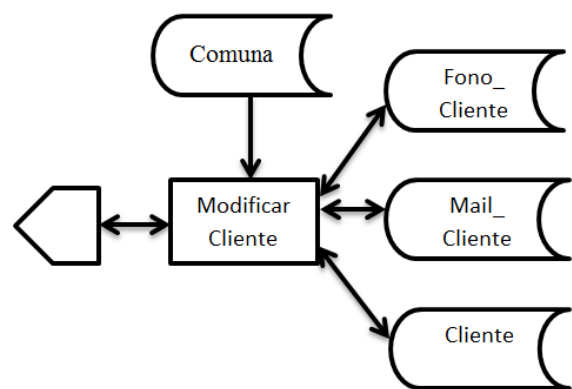
Figura 3-8. Vista de consultar cliente

Anexo: el código fuente que busca algún cliente se encuentra en la página 101.

3.4.8. Modificar cliente

- a) **Nombre:** Modificar Cliente.
- b) **Objetivo:** Modifica un cliente existente.

c) Diagrama de bloques.



Fuente propia.

Diagrama 3-12. Diagrama de bloques Modificar cliente

d) Procedimiento:

Al marcar la opción de modificar, aparecerán los cuadros de Nueva comuna y ciudad, teléfonos e E-mails.

Se puede modificar cualquiera de los atributos del cliente, exceptuando el rut de éste. Al presionar el botón Guardar, el sistema validará que los campos; Nombre, Apellido, Giro y Dirección estén poblados para proceder a sobre-escribir los datos en el archivo Cliente.

Modificar comuna asociada al cliente: En caso que se haya ingresado mal la comuna del cliente, ésta se podrá cambiar seleccionando alguna del cuadro de Nueva comuna y Ciudad y al presionar el botón guardar, el sistema sobre-escribirá los datos.

Modificar un número telefónico: Para poder editar el número telefónico, es necesario seleccionar el número en la lista, presionar el botón Borrar, el sistema eliminará el registro de este número asociado al cliente en el archivo Telefono_Cliente. Una vez realizada la operación se procede a editar el número y presionando el botón Agregar; el sistema valida que el número sea de once dígitos, y si es correcto, asociará el número editado nuevamente al cliente seleccionado.

Modificar un E-mail: Para poder editar el E-mail, es necesario seleccionar el número en la lista, presionar el botón Borrar, el sistema eliminará el registro de este E-mail asociado al cliente en el archivo Mail_cliente. Una vez realizada la operación se procede a editar el E-mail y presionando el botón Agregar; el sistema validará que tenga un arroba (@) en una posición de palabra mayor a cero, y un punto en los últimos caracteres del E-mail, esto para validar la extensión, si es correcto asociará el E-mail editado nuevamente al cliente seleccionado.

e) Diseño de pantalla.

☒ Modificar
☐ Eliminar

CONSULTAR CLIENTE

Rut: 19171200-5Comuna: ANTOFAGASTA

Nombre: JorgeRegión: ANTOFAGASTA

Apellido: GonzalezDirección: Calle falsa 123

Giro: Compra/venta

Nueva Comuna: ANTOFAGASTANueva Región: ANTOFAGASTA

Telefonos: + 56996196717

AgregarBorrar

e-mails: Jorgegonzalez@gmail.com

AgregarBorrar

Palabra Clave:

Buscar

Rut	Nombre	Apellido	Giro	Región	Comuna	Dirección
139058...	felipe	orellana	ventas	ANTOF...	ANTOFA...	calle fal...
191712...	Jorge	Gonzalez	Compra...	VALPAR...	VALPAR...	Calle fal...
191919...	Kevin	Perez	Venta R...	ANTOF...	ANTOFA...	los Sau...
233213...	Laura	Fuentes	Revend...	VALPAR...	VILLA A...	cale ma...
330477...	Debora	Suarez	Mall	ANTOF...	SIERRA...	Calle lo...
616161	iorge	Brto	feriante	ANTOF	ANTOFA	nasaie t

Guardar

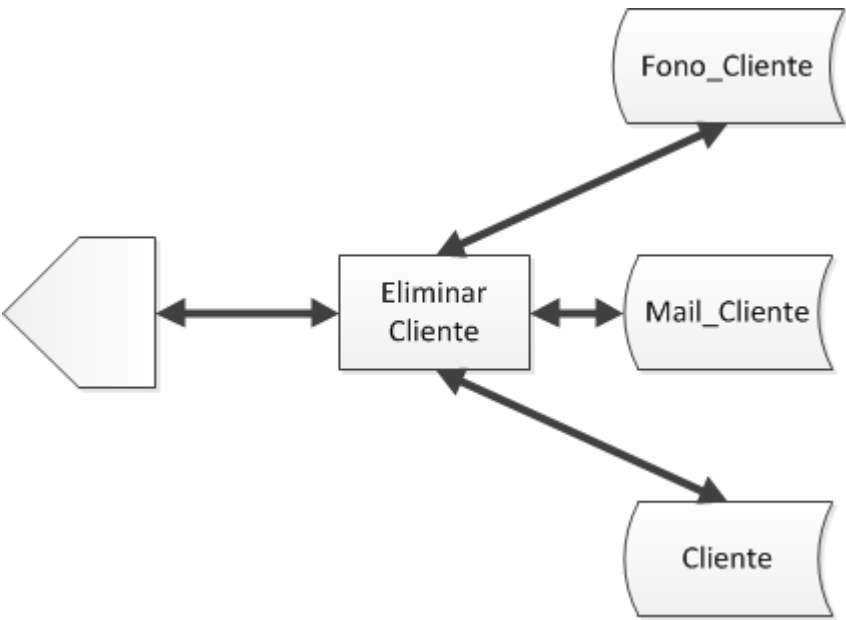
Fuente propia.

Figura 3-9. Vista modificar cliente

Anexo: el código fuente que modifica un cliente se encuentra en la página 101.

3.4.9. Eliminar Cliente

- a) **Nombre:** Eliminar Cliente.
- b) **Objetivo:** Elimina un cliente ingresado previamente en el sistema.
- c) **Diagrama de bloques.**



Fuente propia.

Diagrama 3-13. Diagrama de bloques eliminar Cliente

d) **Procedimiento:**

Se busca un cliente a través del cuadro de texto junto al botón Buscar o se selecciona manualmente haciendo clic en algún elemento de la lista de clientes. Seguido a esto se presiona el botón Eliminar y luego de que el sistema muestre un aviso de advertencia, eliminará automáticamente los datos de contacto registrados en el archivo Telefono_Cliente y Mail_Cliente. Seguido a esto el sistema procederá a eliminar el registro de este cliente en el archivo Cliente y todos los datos asociados a este cliente.

Si el cliente está asociado a alguna otra tabla, ya sea, Cuotas, Facturas o Reservas el sistema dará aviso que no es posible eliminar el cliente porque está siendo referenciado por otros archivos.

e) Diseño de pantalla

☐ Modificar

☒ Eliminar

CONSULTAR CLIENTE

Rut : 19171200-5

Comuna : ANTOFAGASTA

Nombre : Jorge

Región : ANTOFAGASTA

Apellido : González

Dirección : Calle falsa 123

Giro : Compra/venta

Palabra Clave :

Buscar

Rut	Nombre	Apellido	Giro	Región	Comuna	Dirección
139058...	Kevin	Suarez	ventas	ANTOF...	ANTOFA...	calle fal...
191712...	Jorge	Gonzalez	Compra...	VALPAR...	VALPAR...	Calle fal...
191919...	Kevin	Perez	Venta R...	ANTOF...	ANTOFA...	los Sau...
233213...	Laura	Fuentes	Revend...	VALPAR...	VILLA A...	cale ma...
330477...	Debora	Suarez	Mall	ANTOF...	SIERRA...	Calle lo...
616161	iorge	Brito	feriante	ANTOF	ANTOFA	nasaie.t

Eliminar

Fuente Propia.

Figura 3-10. Vista eliminar Cliente

Anexo: el código fuente que elimina un cliente se encuentra en la página 105.

3.4.10. Ingresar Cuotas

- a) **Nombre:** Ingresar Cuotas.
- b) **Objetivo:** Asocia cuotas a alguna venta realizada

c) Diagrama de bloques:

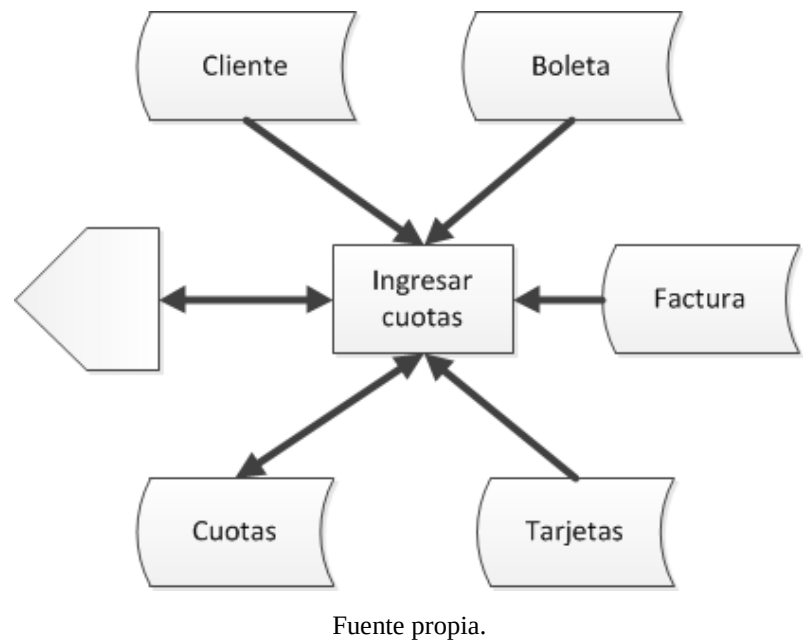


Diagrama 3-14. Diagrama de bloques Ingresar Cuotas

d) Procedimiento:

Se decide con el menú principal si se realizará una venta con boleto o factura, para que la cuota quede asociada a alguno de estos dos recibos.

Luego de esto se debe activar la opción de ingreso por cuotas para desplegar la vista de ingreso.

Se debe ingresar el rut del cliente que solicita el pago por cuotas y presionar la tecla ENTER para que el sistema calcule el dígito verificador y busque el cliente en la base de datos, en caso que este no exista, se mostrará la opción para abrir el programa de ingreso de cliente, y si existe, se mostrará automáticamente su nombre.

Se debe elegir la tarjeta con la que realizará el pago y la cantidad de cuotas que solicita, una vez ingresada la cantidad de cuotas; se presiona la tecla ENTER y el sistema mostrará el valor que tendrá cada cuota.

Validaciones que hará el programa antes de registrar la cuota:

- Verificar que el rut tenga 8 dígitos.
- Verificar que el dígito verificador corresponda al rut ingresado.
- Verificar que el cliente existe en el sistema.
- Verificar que se ha ingresado una cantidad de cuotas.
- Verificar que se haya mostrado un valor de cuota distinto de 0.

e) Diseño de pantalla:

☐ Pago con tarjeta

VENTA CON BOLETA
16/03/2016

BOLETA
N° : 5644...

Rut : 19171200 — 5

Nombre Cliente Jorge

Nombre tarjeta VISA

Cantidad cuotas2Valor cuota : 65.000

Agregar línea al detalleQuitar línea al detalle

Codigo producto	Nombre del Pro...	Cantidad producto	Valor unitario	Valor total unitario
5112	Maniqu niño*	3	30.000	90.000
1234	PERCHEROS*	20	2.000	40.000

Nueva VentaTOTAL : 130.000

Guardar venta

Fuente Propia.

Figura 3-11. Vista Ingresar Cuotas

Anexo: el código fuente que ingresa una cuota se encuentra en la página 106.

3.4.11. Ingresar Reserva

- a) Nombre: Ingresar reserva.
- b) Objetivo: Ingresar una reserva de producto.

c) Diagrama de bloques:

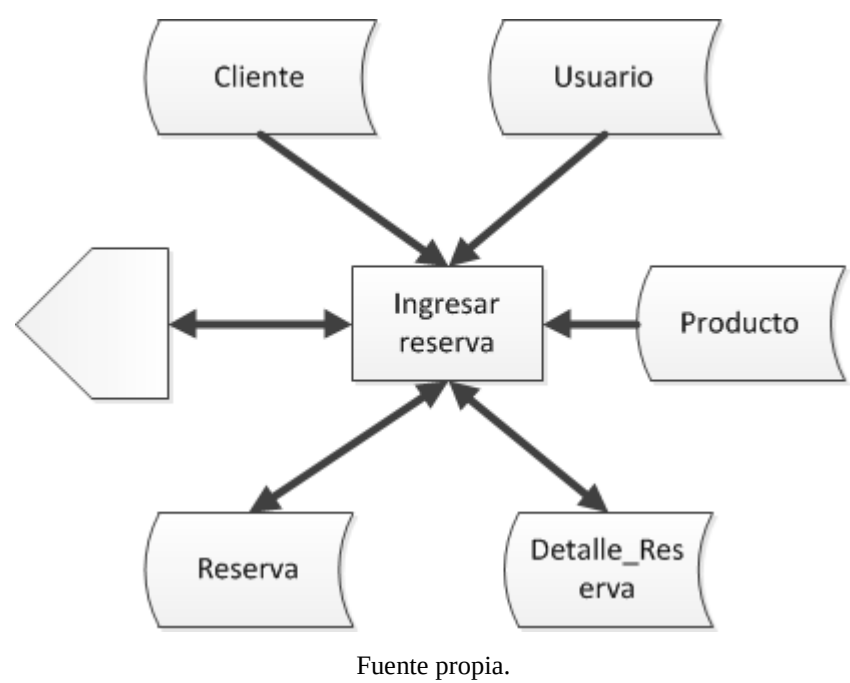


Diagrama 3-15. Diagrama de bloques Ingresar Reserva

d) Procedimiento:

Se ingresa el rut del cliente de ocho dígitos y se presiona la tecla ENTER, el sistema calculará el dígito verificador para el rut y si el cliente existe, entonces mostrará su nombre, si este no existe, entonces mostrará la opción para ir al programa de ingreso cliente. Lo mismo para el usuario; se ingresa el rut y el sistema busca que el usuario exista en el sistema.

Por cada producto que se ingrese en el detalle de reserva se debe agregar una fila al detalle presionando en el botón Agregar línea al detalle, una vez aparezca fila se debe ingresar el código de producto y la cantidad de este que serán reservados, seguido a esto se presiona la tecla ENTER y únicamente para la fila el sistema validará:

- La existencia código del producto en la base de datos.
- La cantidad sea mayor a cero.
- Sólo se hayan ingresado datos numéricos.

Una vez cumplidas estas condiciones, el sistema completara la línea con el nombre del producto y su estado (en bodega, en proceso), además, asignará un estado total para la reserva según el estado de cada fila.

Una vez que se quiera confirmar el ingreso de la reserva a la base de datos, el sistema hará las siguientes validaciones antes de aceptar el registro:

- El rut del cliente debe tener ocho dígitos.
- El dígito verificador del rut del cliente, debe corresponder al mismo.
- El cliente debe existir en el sistema.
- El rut del usuario debe tener ocho dígitos.
- El dígito verificador del rut del usuario, debe corresponder al mismo.
- El usuario debe existir en el sistema.
- Se debe haber ingresado al menos un producto correctamente en el detalle de la reserva.
- La reserva debe contener un estado total de reserva.

e) Diseño de pantalla:

RESERVAR PRODUCTO

Rut Cliente : 18037745...

K

Fecha : 20/03/2016

Cliente : Javier Echeverria

Rut Usuario : 19171200...

5

Estado : EN PROCESO

Usuario : JUAN

Agregar linea al detalle

Quitar linea al detalle

Codigo producto	Nombre	Cantidad	Estado
5112	Maniqu niño*	3	EN BODEGA
5321	GANCHO RANURADO*	250	EN PROCESO

Nueva

Guardar Reserva

Fuente Propia.

Figura 3-12. Vista Ingresar Reserva

Anexo: el código fuente que ingresa una reserva se encuentra en la página 109.

CONCLUSIONES Y RECOMENDACIONES

Al finalizar el proyecto desarrollado a lo largo de toda la memoria, se observó que se han podido cubrir todos los requerimientos descritos al inicio de ésta y que presentaban un problema para la empresa en la cual se implementaría.

A lo largo del desarrollo del software surgieron algunos problemas que no necesariamente estaban contemplados como requerimientos para el mismo, como por ejemplo; las dificultades que presentaron las especificaciones propuestas por el cliente, que no siempre fueron muy claras, de modo que en más de una oportunidad hubieron entrevistas. Hubo que hacer también algunas investigaciones por cuenta propia, como por ejemplo, el desarrollo del contenido de alguna factura, algunas fórmulas matemáticas para obtener los resultados ofrecidos por el software, o cambios que fue presentando la estructura de la base de datos para un uso más óptimo de los datos. Sin embargo, el usuario final nos orientó muy bien en la etapa de diseño y en el equipo de trabajo del proyecto había quien conocía muy bien el negocio.

La superación de estos problemas y el registro de los mismos en la planificación de trabajo que se utilizó, ayudó a que más adelante se puedan enfrentar cualquier tipo de problemas, ya sean de requerimiento o no, de una forma mucho más rápida.

Algunos de los puntos fundamentales para el éxito del proyecto incluyen la planificación del trabajo a plazo, el trabajo en equipo y la correcta distribución de tareas acorde a las habilidades de cada integrante del equipo de trabajo.

Con lo que respecta al proyecto mismo se intentó crear una interfaz amigable para que el usuario no tuviese que invertir demasiado tiempo, los resultados fueron positivos, ya que al presentar una interfaz que reflejaba muy bien los documentos de papel cualquier usuario común y corriente y no necesariamente experto en el rubro, supo qué campos llenar y con qué datos sin necesidad de explicarle demasiado.

Por desgracia el sistema no pudo ser implementado, porque la empresa Karishmas Importaciones LTDA, cerró para cambiar su rubro a perfumería.

Cabe destacar que la institución universitaria y el plan mismo de la carrera nos prepara muy bien y de forma acabada para poder realizar cada una de las tareas que fue necesaria para el desarrollo del proyecto.

BIBLIOGRAFIA

SUBSECRETARÍA DEL DESARROLLO. Regiones, Provincias y Comunas de Chile [en línea]

<<http://www.subdere.cl/documentacion/regiones-provincias-y-comunas-de-chile>>

[consulta: 10 de Septiembre del 2015]

UNCOMO EDUCACIÓN. Cómo sacar un porcentaje de una cifra [en línea]

<[http://educacion.uncomo.com/articulo/como-sacar-un-porcentaje-de-una-cifra-](http://educacion.uncomo.com/articulo/como-sacar-un-porcentaje-de-una-cifra-3733.html)

3733.html> [consulta: 23 de Mayo del 2015]

BRAZÁLEZ, Alfonso, CALLEJA, Jesús, GARCIA, Javier, GARCIA, Jon, IMAZ, Aitor, LARZABAL, Alberto, MINGO, Iñigo y RODRIGUEZ, José. Aprende Java como si estuviera en primero. San Sebastián. Universidad de Navarra. 2000. Colección Aprende..., como si estuviera en primero.

ANEXO A: CÓDIGO FUENTE DEL SISTEMA DE VENTAS E INVENTARIO

DESARROLLADO

A continuación se mostrará la estructura física del Software en su totalidad, además del código fuente de cada uno de los programas que fueron descritos anteriormente en el capítulo tres de la presente memoria.

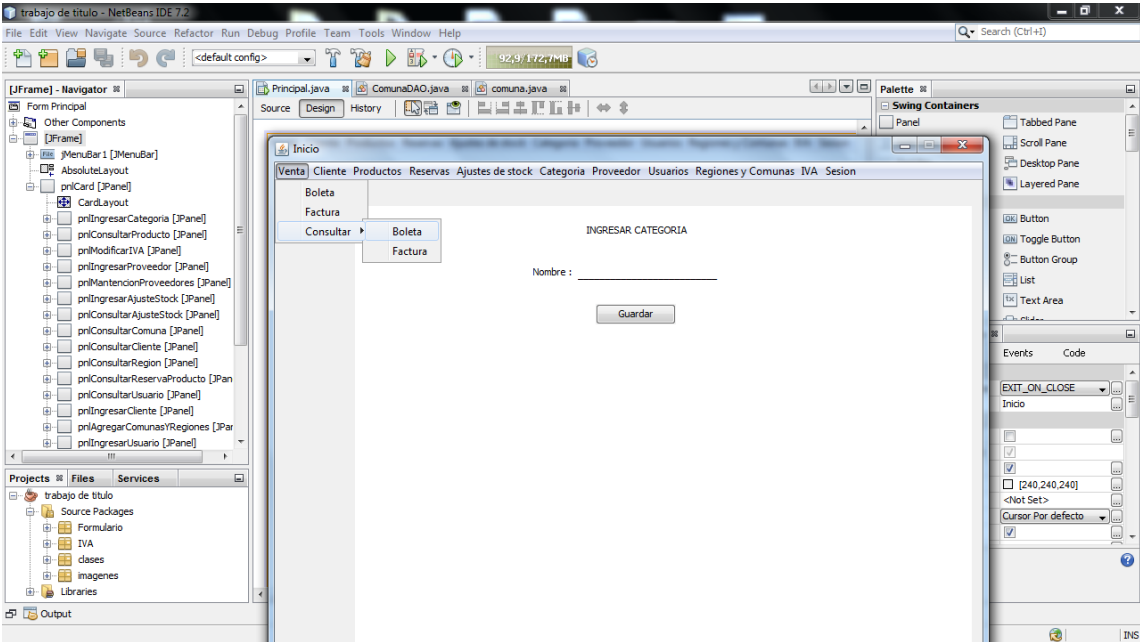
La estructura del Software:

- **Lo visual:** Para la parte visual se utiliza una vista muy similar a lo que sería un formulario impreso en una hoja de papel, esto para que cuando el cliente final deba usarlo, se le deba explicar lo menos posible cuáles son los campos que debe llenar,

La forma de moverse a través de los distintos programas y funcionalidades que ofrece el software es con un menú horizontal ubicado en la parte superior del mismo, y muestra los programas con un layout tipo card, esto para que el usuario pueda moverse libremente por los programas sin necesidad de guardar su avance ni perdiendo la información de un formulario al pasar a otro.

- **La Programación:** La programación se hizo orientada a objetos en la cual cada entidad de la base de datos tiene su representante en la aplicación. Así mismo se utiliza una organización de tareas por capas en las que están:
 - **Dominio:** Clases que representan la base de datos en la aplicación.
 - **Negocio:** Clases que se encargan de hacer algunos cálculos o validaciones.
 - **Presentación:** Clases que muestran la parte visual del sistema, como por ejemplo el formulario de venta.

Se muestra la estructura real de la aplicación, al centro se sitúa la vista previa de la construcción de esta misma, en la parte superior izquierda se muestra el navegador de los objetos visuales, todos ligados al panel con layout tipo card. Mientras que en la parte inferior izquierda se muestra el navegador de los objetos estructurales , entre ellos las imágenes de los datos entrantes y las clases de dominio y negocio en la misma carpeta.



Fuente propia.

Figura A-1. Vista estructural de la aplicación

Código fuente de los programas descritos:

A continuación se presentan los códigos fuentes de los programas descritos en el capítulo anterior, estos códigos corresponden a la parte exacta en la cual se presiona la tecla ENTET y/o el botón aceptar del programa descrito.

AGREGAR VENTA CON BOLETA.

Presionar tecla ENTER en el detalle de la boleta:

```
private void tblProductosVentaBoletaKeyPressed(java.awt.event.KeyEvent evt) {  
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {  
        int fila = tblProductosVentaBoleta.getSelectedRow();
```

```

if (!(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 0))).equals(""))
&& !(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 2))).equals("")
&& !(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 3))).equals("")) {
    String vcod = String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 0));
    String vcant = String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 2));
    String vvalor = String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 3));
    Validaciones val = new Validaciones();
    if (val.numeros(vcod, vcant, vvalor)) {
        if (vcant.length() <= 4) {
            String nombre = "";
            producto prod = new producto();
            int codigo = Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 0)));
            int CA = Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 2)));
            if (!(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 1))).endsWith("*")) {
                producto prod2 = prod.buscar(codigo);
                nombre = prod2.getNombre();
                int valor = prod2.getPrecioVenta();
                String nom = nombre.concat("*");
                if (!nombre.equals("")) {
                    int cantidad = Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 2)));
                    int totalUnitario = cantidad * Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 3)));
                    int CP = prod2.getStockActual();
                    int resultado = CP - CA;
                    if (CP != 0) {
                        if (resultado < 0) {
                            int rneg = CA + resultado;
                            JOptionPane.showMessageDialog(null, "Del producto CODIGO = " + codigo + " hay SOLO " + rneg + " en bodega"
                                + " por tanto se asumirá que el producto existe físicamente y no se ha ingresado al sistema, DE LO CONTRARIO "
                                + " ELIMINE FILA Y AGREGUE NUEVAMENTE");
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
        if (valor < Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(fila, 3)))) {
            tblProductosVentaBoleta.setValueAt(nom, fila, 1);
            tblProductosVentaBoleta.setValueAt(totalUnitario, fila, 4);
            String total = Integer.toString(Integer.parseInt(txtTotalVentaBoleta.getText()) + totalUnitario);
            txtTotalVentaBoleta.setText(total);
        } else {
            JOptionPane.showMessageDialog(null, "Lo sentimos, el precio del producto no puede ser menor al precio de compra");
        }
    } else {
        JOptionPane.showMessageDialog(null, "No se ha encontrado ningún producto con ese código");
    }
}
} else {
    JOptionPane.showMessageDialog(null, "la cantidad no puede ser mayor a 9999");
}

} else {
    JOptionPane.showMessageDialog(null, "Solo debe introducir numeros, se han detectado caracteres");
}
} else {
    JOptionPane.showMessageDialog(null, "Debe completar CODIGO, CANTIDAD y VALOR UNITARIO");
}
}
}

```

Presionar botón de Guardar venta:

```

private void btnGuardarVentaVentaBoletaActionPerformed(java.awt.event.ActionEvent evt) {
    if (chkCuotasVentaBoleta.isSelected()) {

```

```

        if ((txtNumeroBoletaVentaBoleta.getText().length() == 6) &&
(Integer.parseInt(txtNumeroBoletaVentaBoleta.getText()) != 0)) {
            if (!txtTotalVentaBoleta.getText().equals("0")) {
                if (txtRutVentaBoleta.getText().length() == 8) {
                    if (!txtDVVentaBoleta.getText().equals("")) {
                        if (!txtCuotasVentaBoleta.getText().equals("")) {
                            if (!txtValorCuotaVentaBoleta.getText().equals("")) {
                                if (!txtNombreClienteVentaBoleta.getText().equals("")) {
                                    int numFactura;
                                    int total;
                                    boolean MedioPago = chkCuotasVentaBoleta.isSelected();
                                    String medioPago;
                                    numFactura =
Integer.parseInt(txtNumeroBoletaVentaBoleta.getText());
                                    total = Integer.parseInt(txtTotalVentaBoleta.getText());
                                    if (MedioPago) {
                                        medioPago = "1";
                                    } else {
                                        medioPago = "0";
                                    }
                                    Boleta bol = new Boleta();
                                    bol.setNumBoleta(numFactura);
                                    bol.setTotal(total);
                                    bol.setMedioPago(medioPago);
                                    bol.setFecha();
                                    ArrayList detalle = new ArrayList<DetalleBoleta>();
                                    int registros = tblProductosVentaBoleta.getRowCount();
                                    for (int i = 0; i < registros; i++) {
                                        DetalleBoleta det = new DetalleBoleta();
                                        det.setNumBoleta(numFactura);

det.setCodProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValue
At(i, 0))));

det.setCantProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValue
At(i, 2))));

```

```
det.setValorUnitario(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(i, 3))));
```

```
        detalle.add(det);
    }
    String rutCliente = txtRutVentaBoleta.getText();
    rutCliente = rutCliente.concat("-");
    rutCliente = rutCliente.concat(txtDVVentaBoleta.getText());
    VentaFactura vf = new VentaFactura();
    Cuotas cu = new Cuotas();
    cu.setCodTarjeta(vf.cod_tarjeta((String)
cmbTarjetasVentaBoleta.getSelectedItem()));
```

```
cu.setCodRecibo(Integer.parseInt(txtNumeroBoletaVentaBoleta.getText()));
    cu.setTipoRecibo("B");
    cu.setValor((int)
Float.parseFloat(txtValorCuotaVentaBoleta.getText()));
    cu.setFecha(bol.getFecha2());
    cu.setRutCliente(rutCliente);
    cu.setNumCuota(0);
```

```
cu.setTotalCuotas(Integer.parseInt(txtCuotasVentaBoleta.getText()));
    BoletaDAO bolDAO = new BoletaDAO();
    bolDAO.insertarC(bol, detalle, cu);
    } else {
        JOptionPane.showMessageDialog(null, "NO hay ningún cliente
expresado en la seccion de CUOTAS");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Acepte las cuotas
precionando ENTER");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Ingrese la cantidad De
cuotas y aceptelas porfavor");
    }
    } else {
```

```

        JOptionPane.showMessageDialog(null, "Revise el Dígito verificador");
    }
} else {
    JOptionPane.showMessageDialog(null, "Numero de rut a ingresar debe
tener 8 caracteres");
}
} else {
    JOptionPane.showMessageDialog(null, "No puede realizar una venta con
Total = 0");
}
} else {
    JOptionPane.showMessageDialog(null, "La Boleta Debe tener un número de 6
dígitos y no deben ser solo ceros");
}
} else {
    if ((txtNumeroBoletaVentaBoleta.getText().length() == 6) &&
(Integer.parseInt(txtNumeroBoletaVentaBoleta.getText()) != 0)) {
        if (!txtTotalVentaBoleta.getText().equals("0")) {
            int numFactura;
            int total;
            boolean MedioPago = chkCuotasVentaBoleta.isSelected();
            String medioPago;
            numFactura = Integer.parseInt(txtNumeroBoletaVentaBoleta.getText());
            total = Integer.parseInt(txtTotalVentaBoleta.getText());
            if (MedioPago) {
                medioPago = "1";
            } else {
                medioPago = "0";
            }
            Boleta bol = new Boleta();
            bol.setNumBoleta(numFactura);
            bol.setTotal(total);
            bol.setMedioPago(medioPago);
            bol.setFecha();
            ArrayList detalle = new ArrayList<DetalleBoleta>();
            int registros = tblProductosVentaBoleta.getRowCount();
            for (int i = 0; i < registros; i++) {

```

```

        DetalleBoleta det = new DetalleBoleta();
        det.setNumBoleta(numFactura);

        det.setCodProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(i, 0))));

        det.setCantProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(i, 2))));

        det.setValorUnitario(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValueAt(i, 3))));

        detalle.add(det);
    }
    BoletaDAO bolDAO = new BoletaDAO();
    bolDAO.insertar(bol, detalle);
} else {
    JOptionPane.showMessageDialog(null, "No puede realizar una venta con
Total = 0");
}
} else {
    JOptionPane.showMessageDialog(null, "La Boleta Debe tener un número de 6
dígitos y no pueden ser solo ceros");
}
}
}
}

```

GUARDAR VENTA CON FACTURA.

Al presionar la tecla ENTER en el detalle de la factura:

```

private void tblProductosVentaFacturaKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        int fila = tblProductosVentaFactura.getSelectedRow();
    }
}

```

```

        if (!(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 0))).equals(""))
        && !(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 2))).equals("")) &&
        !(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 5))).equals("")) {
            String vcod = String.valueOf(tblProductosVentaFactura.getValueAt(fila, 0));
            String vcant = String.valueOf(tblProductosVentaFactura.getValueAt(fila, 2));
            String vvalor = String.valueOf(tblProductosVentaFactura.getValueAt(fila, 5));
            Validaciones val = new Validaciones();
            if (val.numeros(vcod, vcant, vvalor)) {
                if (vcant.length() <= 4) {
                    String nombre = "";
                    producto prod = new producto();
                    int                                codigo                                =
Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 0)));
                    int                                CA                                =
Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 2)));
                    if      (!(String.valueOf(tblProductosVentaFactura.getValueAt(fila,
1))).endsWith("*")) {
                        producto prod2 = prod.buscar(codigo);
                        nombre = prod2.getNombre();
                        String nom = nombre.concat("*");
                        if (!nombre.equals("")) {
                            float iva = 0;
                            try {
                                iva = prod.sacarIVA();
                            } catch (IOException ex) {
                                JOptionPane.showMessageDialog(null, "Problemas al intentar
acceder al IVA");
                            }
                            float      netoUnitario      =      iva      *
Float.parseFloat(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 5)));
                            int      cantidad      =
Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 2)));
                            int      totalUnitario      =      cantidad      *
Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 5)));
                            int CP = prod2.getStockActual();
                            int resultado = CP - CA;
                            if (CP != 0) {

```

```

        if (resultado < 0) {
            int rneg = CA + resultado;
            JOptionPane.showMessageDialog(null, "Del producto
CODIGO = " + codigo + " hay SOLO " + rneg + " en bodega"
            + " por tanto se asumirá que el producto existe físicamente
y no se ha ingresado al sistema, DE LO CONTRARIO "
            + " ELIMINE FILA Y AGREGUE NUEVAMENTE");
        }
    }
    int valor = prod2.getPrecioVenta();
    if (valor <
Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(fila, 5)))) {
        tblProductosVentaFactura.setValueAt(nom, fila, 1);
        tblProductosVentaFactura.setValueAt(totalUnitario, fila, 6);
        String total =
Integer.toString(Integer.parseInt(txtTotalVentaFactura.getText()) + totalUnitario);
        txtTotalVentaFactura.setText(total);
        float totalNetoUnitario = (float) cantidad * netoUnitario;
        String totalNeto =
Float.toString(Float.parseFloat(txtTotalNetoVentaFactura.getText()) + (float)
totalNetoUnitario);
        txtTotalNetoVentaFactura.setText(totalNeto);
        netoUnitario = (float) Math rint(netoUnitario);

        tblProductosVentaFactura.setValueAt(netoUnitario, fila, 3);
        tblProductosVentaFactura.setValueAt(totalNetoUnitario, fila, 4);
    } else {
        JOptionPane.showMessageDialog(null, "Lo sentimos, el precio
del producto no puede ser menor al precio de compra");
    }
} else {
    JOptionPane.showMessageDialog(null, "Debe ingresar números y se
han detectado caracteres");
}
} else {
    JOptionPane.showMessageDialog(null, "No se ha encontrado ningún
producto con ese código");
}

```

```

        }
    } else {
        JOptionPane.showMessageDialog(null, "la cantidad no puede exceder los
9999");
    }
}
} else {
    JOptionPane.showMessageDialog(null, "Debe completar CODIGO,
CANTIDAD y VALOR UNITARIO");
}
}
}

```

Al presionar la tecla ENTER en el rut del cliente:

```

private void txtRutVentaFacturaKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        if (txtRutVentaFactura.getText().length() == 8) {
            Validaciones val = new Validaciones();
            txtDVVentaFactura.setText(val.sacarDV(txtRutVentaFactura.getText()));
            String rut = txtRutVentaFactura.getText().concat("-");
            rut = rut.concat(txtDVVentaFactura.getText());
            cliente cli = new cliente(rut);
            comuna com = new comuna(cli.getCodComuna());
            region reg = new region(com.getCod_region());
            txtNombreVentaFactura.setText(cli.getNombre());
            txtApellidoVentaFactura.setText(cli.getApellido());
            txtDireccionVentaFactura.setText(cli.getDireccion());
            txtGiroVentaFactura.setText(cli.getGiro());
            txtComunaVentaFactura.setText(com.getNom_comuna());
            txtRegionVentaFactura.setText(reg.getNombre());
            if (txtNombreVentaFactura.getText().equals("")) {
                int resp = JOptionPane.showConfirmDialog(null, "El rut ingresado no
pertenece a ningún cliente, ¿Desea ir al panel de ingreso?");
                if (resp == 0) {
                    txtRutIngresarCliente.setText(rut);
                    txtDVIngresarCliente.setText(txtDVVentaFactura.getText());
                }
            }
        }
    }
}

```

```

        desplegar(pnlIngresarCliente);
    }
}
} else {
    JOptionPane.showMessageDialog(null, "El rut debe ser de 8 dígitos");
}
}
}
}

```

Al presionar el botón Guardar venta:

```

private void
btnGuardarVentaVentaFacturaActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtRutVentaFactura.getText().length() == 8) {
        if (!txtDVVentaFactura.getText().equals("")) {
            if (!txtNumeroFacturaVentaFactura.getText().equals("")) {
                if (txtNumeroFacturaVentaFactura.getText().length() == 6) {
                    if (!txtTotalVentaFactura.getText().equals("0")) {
                        if (!txtNombreVentaFactura.getText().equals("")) {
                            int numFactura;
                            String rutCliente;
                            int total;
                            int totalNeto;
                            boolean MedioPago = chkCuotasVentaFactura.isSelected();
                            String medioPago;
                            numFactura =
Integer.parseInt(txtNumeroFacturaVentaFactura.getText());
                            rutCliente = txtRutVentaFactura.getText().concat("-");
                            rutCliente = rutCliente.concat(txtDVVentaFactura.getText());
                            total = Integer.parseInt(txtTotalVentaFactura.getText());
                            totalNeto =
Float.parseFloat(txtTotalNetoVentaFactura.getText());
                            if (MedioPago) {
                                medioPago = "1";
                            } else {
                                medioPago = "0";
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

factura fac = new factura();
fac.setFecha();
fac.setNumFactura(numFactura);
fac.setRutCliente(rutCliente);
fac.setTotal(total);
fac.setValorNetoTotal(totalNeto);
fac.setMedioPago(medioPago);
FacturaDAO facDAO = new FacturaDAO();
ArrayList detalle = new ArrayList<DetalleFactura>();
int registros = tblProductosVentaFactura.getRowCount();
for (int i = 0; i < registros; i++) {
    DetalleFactura det = new DetalleFactura();
    det.setNumFactura(numFactura);

det.setCod_producto(Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(i, 0))));

det.setCantProducto(Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(i, 2))));

int valorNeto = (int)
Float.parseFloat(String.valueOf(tblProductosVentaFactura.getValueAt(i, 3)));
det.setValorNetoUnitario(valorNeto);

det.setValorUnitario(Integer.parseInt(String.valueOf(tblProductosVentaFactura.getValueAt(i, 5))));

    detalle.add(det);
}
DetalleFacturaDAO detF = new DetalleFacturaDAO();
if (medioPago.equals("1")) {
    if (!txtCuotasVentaFactura.getText().equals("")) {
        if (!txtValorCuotaVentaFactura.getText().equals("")) {
            VentaFactura vf = new VentaFactura();
            Cuotas cu = new Cuotas();
            cu.setCodTarjeta(vf.cod_tarjeta((String)
cmbTarjetasVentaFactura.getSelectedItem()));

cu.setCodRecibo(Integer.parseInt(txtNumeroFacturaVentaFactura.getText()));

```

```

        cu.setTipoRecibo("F");
        cu.setValor((int)
Float.parseFloat(txtValorCuotaVentaFactura.getText()));
        cu.setFecha(fac.getFecha2());
        cu.setRutCliente(rutCliente);
        cu.setNumCuota(0);

cu.setTotalCuotas(Integer.parseInt(txtCuotasVentaFactura.getText()));
        CuotasDAO insertarCuota = new CuotasDAO();
        insertarCuota.insertar(cu);
        facDAO.insertar(fac);
        detF.insertar(detalle);
    } else {
        JOptionPane.showMessageDialog(null, "Para aceptar el
valor de las cuotas, por favor PRESIONE ENTER en el campo CANTIDAD
CUOTAS");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Campo CANTIDAD
CUOTAS está VACIO");
    }
    } else {
        facDAO.insertar(fac);
        detF.insertar(detalle);
    }
    } else {
        JOptionPane.showMessageDialog(null, "No hay ningún cliente
asociado a esta factura");
    }
    } else {
        JOptionPane.showMessageDialog(null, "No puede Vender una Factura
con Valor 0");
    }
    } else {
        JOptionPane.showMessageDialog(null, "El numero de factura debe ser de
6 dígitos");
    }

```

```

    } else {
        JOptionPane.showMessageDialog(null, "El campo NUMERO FACTURA
está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Para aceptar el Dígito verificador
presione ENTER en el RUT");
    }
    } else {
        JOptionPane.showMessageDialog(null, "El rut debe tener 8 dígitos");
    }
    }
}

```

INGRESAR PRODUCTO.

Al presionar el botón Guardar:

```

btnGuardarProductoIngresarProductoActionPerformed(java.awt.event.ActionEvent evt)
{
    if (txtFotoIngresarProducto.getText().endsWith(".jpg")) ||
txtFotoIngresarProducto.getText().endsWith(".png") ||
txtFotoIngresarProducto.getText().endsWith(".gif") ||
txtFotoIngresarProducto.getText().endsWith(".jpeg")) {
        if (!txtNombreIngresarProducto.getText().equals("")) {
            if (!txtCodigoIngresarProducto.getText().equals("")) {
                if (txtCodigoIngresarProducto.getText().length() == 4) {
                    if (!txtCantidadIngresarProducto.getText().equals("")) {
                        if (!txtPrecioVentaIngresarProducto.getText().equals("")) {
                            if (!txtStockCriticoIngresarProducto.getText().equals("")) {
                                if (txtNotaIngresarProducto.getText().equals("")) {
                                    JOptionPane.showMessageDialog(null, "El producto se agregó
sin nota, de cualquier forma puede ingresar una luego");
                                }
                                int numero =
Integer.parseInt(txtCodigoIngresarProducto.getText());

```

```

        int                stocking                =
Integer.parseInt(txtCantidadIngresarProducto.getText());

        int                precio                =
Integer.parseInt(txtPrecioVentaIngresarProducto.getText());

        int                critico                =
Integer.parseInt(txtStockCriticoIngresarProducto.getText());

        if (numero != 0 && stocking > 0 && precio > 0 && critico > 0) {
            stocking                =
Integer.parseInt(txtCantidadIngresarProducto.getText());

            if (critico < stocking) {
                int                codP                =
Integer.parseInt(txtCodigoIngresarProducto.getText());

                int                pVenta                =
Integer.parseInt(txtPrecioVentaIngresarProducto.getText());

                int                canP                =
Integer.parseInt(txtCantidadIngresarProducto.getText());

                int                stockC                =
Integer.parseInt(txtStockCriticoIngresarProducto.getText());

                categoria cat = new categoria();
                String                nomc                =                (String)
cmbCategoriaIngresarProducto.getSelectedItemAt();

                int codC = cat.getCodigoCategoria(nomc);
                String                nombre                =
txtNombreIngresarProducto.getText().toUpperCase();

                String nota = txtNotaIngresarProducto.getText();
                producto prod = new producto();
                String foto = txtFotoIngresarProducto.getText();
                String                nomP                =                (String)
cmbProveedorIngresarProducto.getSelectedItemAt();

                prod.insertar(codP, codC, nombre, pVenta, canP, stockC,
nota, foto, nomP);

            } else {
                JOptionPane.showMessageDialog(null, "El Stock critico
debe ser menor a la cantidad ingresada");
            }
        } else {

```

```

        JOptionPane.showMessageDialog(null, "El Código, la cantidad
ingresada, el precio y stock critico no pueden ser cero");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el campo STOCK
CRITICO está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el campo PRECIO
COMPRA está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el campo CANTIDAD está
VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el código de producto debe ser
de 4 dígitos");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el campo CODIGO PRODUCTO
está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "el campo NOMBRE PRODUCTO
está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Por favor ingrese un tipo de imagen
correcto");
    }
}
}

```

CONSULTAR PRODUCTO.

Al presionar el botón Buscar:

```

private void
btnbuscarConsultarProductoActionPerformed(java.awt.event.ActionEvent evt) {
    ProductoDAO pDAO = new ProductoDAO();
    if (!txtBuscadorConsultarProducto.getText().equals("")) {
        pDAO.buscarPalabra(txtBuscadorConsultarProducto.getText(),
tblProductoConsultarProducto);
    } else {
        pDAO.buscarTodo(tblProductoConsultarProducto);
    }
}

```

MODIFICAR PRODUCTO

Al presionar ENTER en la etiqueta de suma:

```

private void txtSumaConsultarProductoKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        int actual = Integer.parseInt(txtcantidadConsultarProducto.getText());
        int agregado = Integer.parseInt(txtSumaConsultarProducto.getText());
        actual = actual + agregado;
        ProductoDAO pDAO = new ProductoDAO();
        pDAO.modificar(Integer.parseInt(txtcodigoConsultarProducto.getText()), actual,
"stock_actual");
        JOptionPane.showMessageDialog(null, "Stock agregado correctamente");
        txtSumaConsultarProducto.setText("");
        txtcantidadConsultarProducto.setText(Integer.toString(actual));
        llenarTablaProductoConsultarProducto();
    }
}

```

Al presionar en el botón Agregar proveedor:

```

private void
btnAgregarProveedorConsultarProductoActionPerformed(java.awt.event.ActionEvent
evt) {
    int codP = Integer.parseInt(txtcodigoConsultarProducto.getText());
    productoProveedor prov = new productoProveedor();
    prov.guardarProductoProveedor(codP, (String)
cmbNuevoProveedorConsultarProducto.getSelectedItem());
    ProductoDAO pDAO = new ProductoDAO();
    cmbProveedorConsultarProducto.removeAllItems();
    pDAO.BuscarProductoProveedor(cmbProveedorConsultarProducto,
Integer.parseInt(txtcodigoConsultarProducto.getText()));
}

```

Al presionar en el botón Eliminar proveedor existente:

```

private void
btnBorrarProveedorConsultarProductoActionPerformed(java.awt.event.ActionEvent evt)
{
    if(cmbProveedorConsultarProducto.getItemCount() > 1){
        int resp = JOptionPane.showConfirmDialog(null, "Borrará asociación a un
proveedor para producto código " + txtcodigoConsultarProducto.getText() + " ,
¿Continuar?");
        if(resp == 0){
            proveedor pro = new proveedor();
            String rutP = pro.getCodigoProveedor((String)
cmbProveedorConsultarProducto.getSelectedItem());
            productoProveedor pDAO = new productoProveedor();
            pDAO.eliminar(Integer.parseInt(txtcodigoConsultarProducto.getText()), rutP);
        }
        else{
            JOptionPane.showMessageDialog(null, "No puede quitarle todos los proveedores
a un producto");
        }
    }
}

```

Al presionar en el botón Cambiar Categoría:

```

private                                                                 void
btnCambiarConsultarProductoActionPerformed(java.awt.event.ActionEvent evt) {
    txtCategoriaConsultarProducto.setText((String)
cmbNuevaCategoriaConsultarProducto.getSelectedItem());
    categoria cat = new categoria();
    int codigoC = cat.getCodigoCategoria(txtCategoriaConsultarProducto.getText());
    ProductoDAO pDAO = new ProductoDAO();
    pDAO.modificar(Integer.parseInt(txtcodigoConsultarProducto.getText()), codigoC,
"cod_categoria");
}

```

Al presionar el botón Guardar:

```

private                                                                 void
btnGuardarConsultarProductoActionPerformed(java.awt.event.ActionEvent evt) {
    if (!txtnombreConsultarProducto.getText().equals("")) {
        if (!txtcantidadConsultarProducto.getText().equals("")) {
            if (!txtprecioventaConsultarProducto.getText().equals("")) {
                if (!txtstockcriticoConsultarProducto.getText().equals("")) {
                    if (Integer.parseInt(txtprecioventaConsultarProducto.getText()) != 0 &&
Integer.parseInt(txtstockcriticoConsultarProducto.getText()) != 0) {
                        int resp = JOptionPane.showConfirmDialog(null, "Modificará
permanentemente los valores del producto código " +
txtcodigoConsultarProducto.getText() + " , ¿Continuar?");
                        if (resp == 0) {
                            int fila = tblProductoConsultarProducto.getSelectedRow();
                            producto produ = new producto();
                            categoria cat = new categoria();

produ.setCodProducto(Integer.parseInt(txtcodigoConsultarProducto.getText()));

produ.setCodCategoria(cat.getCodigoCategoria(txtCategoriaConsultarProducto.getText(
)));

                            produ.setNombre(txtnombreConsultarProducto.getText());

```

```

produ.setPrecioVenta(Integer.parseInt(txtprecioventaConsultarProducto.getText()));

produ.setStockActual(Integer.parseInt(txtcantidadConsultarProducto.getText()));

produ.setStockCritico(Integer.parseInt(txtstockcriticoConsultarProducto.getText()));

produ.setNomFoto(String.valueOf(tblProductoConsultarProducto.getValueAt(fila, 7)));
    produ.setNota(txtnotaConsultarProducto.getText());
    ProductoDAO pDAO = new ProductoDAO();
    pDAO.modificar(produ);
    llenarTablaProductoConsultarProducto();
}
} else {
    JOptionPane.showMessageDialog(null, "El precio de compra y stock
Critico no pueden ser cero");
}
} else {
    JOptionPane.showMessageDialog(null, "el campo PRECIO esta
VACIO");
}
} else {
    JOptionPane.showMessageDialog(null, "el campo PRECIO COMPRA esta
VACIO");
}
} else {
    JOptionPane.showMessageDialog(null, "el campo CANTIDAD esta
VACIO");
}
} else {
    JOptionPane.showMessageDialog(null, "el campo NOMBRE esta VACIO");
}
}

```

ELIMINAR PRODUCTO.

Al presionar en el botón Eliminar:

```

private                                                                 void
btnEliminarConsultarProductoActionPerformed(java.awt.event.ActionEvent evt) {
    if (!txtcodigoConsultarProducto.getText().equals("")) {
        int resp = JOptionPane.showConfirmDialog(null, "Borrará permanentemente el
producto código " + txtcodigoConsultarProducto.getText() + " ¿Continuar?");
        int fila = tblProductoConsultarProducto.getSelectedRow();
        if (resp == 0) {
            int cod = Integer.parseInt(txtcodigoConsultarProducto.getText());
            ProductoDAO prod = new ProductoDAO();
            prod.eliminar(cod);
            llenarTablaProductoConsultarProducto();
            txtnombreConsultarProducto.setText("");
            txtcantidadConsultarProducto.setText("");
            txtSumaConsultarProducto.setText("");
            txtprecioventaConsultarProducto.setText("");
            txtstockcriticoConsultarProducto.setText("");
            txtcodigoConsultarProducto.setText("");
            txtCategoriaConsultarProducto.setText("");
            cmbProveedorConsultarProducto.removeAllItems();
            txtnotaConsultarProducto.setText("");
            ProductoDAO pDAO = new ProductoDAO();
            pDAO.buscarTodo(tblProductoConsultarProducto);
        }
    } else {
        JOptionPane.showMessageDialog(null, "Por favor Seleccione un producto");
    }
}

```

INGRESAR CLIENTE

Al presionar ENTER en el rut del cliente:

```
private void txtRutIngresarClienteKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        txtDVIngresarCliente.setText("");
        if (txtRutIngresarCliente.getText().length() == 8) {
            Validaciones val = new Validaciones();
            txtDVIngresarCliente.setText(val.sacarDV(txtRutIngresarCliente.getText()));
        }else{
            JOptionPane.showMessageDialog(null, "El rut debe tener 8 dígitos");
        }
    }
}
```

Al presionar el botón Guardar:

```
private void btnGuardarIngresarClienteActionPerformed(java.awt.event.ActionEvent
evt) {
    if (!txtNombreIngresarCliente.getText().equals("")) {
        if (!txtApellidoIngresarCliente.getText().equals("")) {
            if (txtRutIngresarCliente.getText().length() == 8) {
                if (!txtDVIngresarCliente.getText().equals("")) {
                    if (!txtGiroIngresarCliente.getText().equals("")) {
                        if (!txtDireccionIngresarCliente.getText().equals("")) {
                            String telefono = "";
                            String Email = "";
                            if (!txtTelefonoIngresarCliente.getText().equals("")) {
                                if (txtTelefonoIngresarCliente.getText().length() >= 10) {
                                    telefono = txtTelefonoIngresarCliente.getText();
                                    telefono = "+" + telefono;
                                    telefono = telefono.replaceAll(" ", "");
                                } else {
                                    JOptionPane.showMessageDialog(null, "El Teléfono debe tener
como mínimo 10 caracteres");
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```

    }
}
if (!txtEmailIngresarCliente.getText().equals("")) {
    if (!txtExtensionIngresarCliente.getText().equals("")) &&
!txtExtensionIngresarCliente.getText().endsWith(".")) {
        Validaciones val = new Validaciones();
        if (val.punto(txtExtensionIngresarCliente.getText())) {
            Email = txtEmailIngresarCliente.getText().replaceAll("@",
"");

            Email = Email.concat("@");
            Email
            =
Email.concat(txtExtensionIngresarCliente.getText().replaceAll("@", ""));
            Email = Email.replaceAll(" ", "");;
        } else {
            JOptionPane.showMessageDialog(null, "Hay un error en la
Extencion de su E-mail");
        }
    } else {
        JOptionPane.showMessageDialog(null, "La extencion del E-
mail esta vacia");
    }
}
String com = (String)
cmbComunaIngresarCliente.getSelectedItem();
cliente cli = new cliente();
cli.setNombre(txtNombreIngresarCliente.getText());
cli.setApellido(txtApellidoIngresarCliente.getText());
String rut = txtRutIngresarCliente.getText();
rut = rut.concat("-");
rut = rut.concat(txtDVIngresarCliente.getText());
cli.setRutCliente(rut);
cli.setGiro(txtGiroIngresarCliente.getText());
cli.setDireccion(txtDireccionIngresarCliente.getText());
comuna comu = new comuna();
int Ccom = comu.getCodigoComuna(com);
cli.setCodComuna(Ccom);
if (telefono.equals("") || Email.equals("")) {

```

```

        int resp = JOptionPane.showConfirmDialog(null, "Los Datos de
Email o telefono estan incompletos, Puede ingresarlos sin problema posteriormente,
continuar");

        if (resp == 0) {
            cli.insertar(telefono, Email);
        }
        } else {
            cli.insertar(telefono, Email);
        }
        } else {
            JOptionPane.showMessageDialog(null, "El campo DIRECCION
está vacío");
        }
        } else {
            JOptionPane.showMessageDialog(null, "El campo GIRO está vacío");
        }
        } else {
            JOptionPane.showMessageDialog(null, "Revise Dígito Verificador
presionando ENTER en el texto RUT");
        }
        } else {
            JOptionPane.showMessageDialog(null, "El rut ingresado no es válido");
        }
        } else {
            JOptionPane.showMessageDialog(null, "El apellido está vacío");
        }
        } else {
            JOptionPane.showMessageDialog(null, "El nombre está vacío");
        }
    }
}

```

CONSULTAR CLIENTE.

Al presionar el botón Buscar:

```

private void btnBuscarConsultarClienteActionPerformed(java.awt.event.ActionEvent
evt) {
    ClienteDAO cDAO = new ClienteDAO();
    cDAO.buscarPalabra(txtBuscadorConsultarCliente.getText(),
tblClientesConsultarCliente);
}

```

MODIFICAR CLIENTE.

Al presionar en el botón Agregar teléfono:

```

private void
btnAgregarTelefonoConsultarClienteActionPerformed(java.awt.event.ActionEvent evt)
{
    if (!txtRutConsultarCliente.getText().equals("")) {
        String telefono = (String) cmbTelefonoConsultarCliente.getSelectedItemAt();
        Validaciones val = new Validaciones();
        telefono = val.repararFono(telefono.trim());
        cmbTelefonoConsultarCliente.setSelectedItemAt(telefono);
        if (telefono.length() > 10 && telefono.length() < 16) {
            int fila = tblClientesConsultarCliente.getSelectedRow();
            String rut = String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0));
            ContactoDAO con = new ContactoDAO();
            con.agregarTelefono("+".concat(telefono), rut);
            ContactoDAO tel = new ContactoDAO();
            ArrayList numeros =
tel.buscarNumeros(String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0)));
            cmbTelefonoConsultarCliente.removeAllItems();
            for (int j = 0; j < numeros.size(); j++) {
                contactoCliente detF = new contactoCliente();

```

```

        detF = (contactoCliente) numeros.get(j);
        cmbTelefonoConsultarCliente.addItem(detF.getTelefono().replace("+", ""));
    }
    } else {
        JOptionPane.showMessageDialog(null, "El telefono debe ser mínimo de 10
dígitos y maximo de 15");
    }
    } else {
        JOptionPane.showMessageDialog(null, "Porfavor seleccione un elemento de la
tabla");
    }
}
}

```

Al presionar el botón Borrar teléfono:

```

private void
btnBorrarTelefonoConsultarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    if(!txtRutConsultarCliente.getText().equals("")){
        int resp = JOptionPane.showConfirmDialog(null, "El Teléfono
Seleccionado se eliminará permanentemente, ¿Continuar?");
        if (resp == 0) {
            String tel = (String) cmbTelefonoConsultarCliente.getSelectedItem();
            ContactoDAO cDAO = new ContactoDAO();
            cDAO.eliminarTelefono("+".concat(tel));
        }
        int fila = tblClientesConsultarCliente.getSelectedRow();
        ContactoDAO tel = new ContactoDAO();
        ArrayList numeros =
tel.buscarNumeros(String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0)));
        cmbTelefonoConsultarCliente.removeAllItems();
        for (int j = 0; j < numeros.size(); j++) {
            contactoCliente detF = new contactoCliente();
            detF = (contactoCliente) numeros.get(j);
            cmbTelefonoConsultarCliente.addItem(detF.getTelefono());
        }
    }else{

```

```

        JOptionPane.showMessageDialog(null, "Porfavor seleccione un elemento de la
tabla");
    }
}

```

Al presionar en el botón Agregar Email

```

private void
btnAgregarEmailConsultarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    if(!txtRutConsultarCliente.getText().equals("")){
        String email = (String) cmbEmailConsultarCliente.getSelectedItemAt();
        email = email.replaceAll(" ", "");
        Validaciones val = new Validaciones();
        if (!email.equals("")) {
            if (val.emailValido(email)) {
                int fila = tblClientesConsultarCliente.getSelectedRow();
                String rut = String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0));
                ContactoDAO con = new ContactoDAO();
                con.agregarEmail(email, rut);
                ContactoDAO mail = new ContactoDAO();
                ArrayList emails =
mail.buscarMails(String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0)));
                cmbEmailConsultarCliente.removeAllItems();
                for (int j = 0; j < emails.size(); j++) {
                    contactoCliente detF = new contactoCliente();
                    detF = (contactoCliente) emails.get(j);
                    cmbEmailConsultarCliente.addItem(detF.getEmail());
                }
            } else {
                JOptionPane.showMessageDialog(null, "Su E-mail no parece ser válido");
            }
        } else {
            JOptionPane.showMessageDialog(null, "No puede agregar un E-Mail vacio");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Porfavor seleccione un elemento de la
tabla");
    }
}

```

```

    }
}

```

Al presionar el botón Borrar Email:

```

private void
btnBorrarEmailConsultarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    if(!txtRutConsultarCliente.getText().equals("")){
        int resp = JOptionPane.showConfirmDialog(null, "El E-mail Seleccionado
se eliminará permanentemente, ¿Continuar?");
        if (resp == 0) {
            String tel = (String) cmbEmailConsultarCliente.getSelectedItemAt();
            ContactoDAO cDAO = new ContactoDAO();
            cDAO.eliminarEmail(tel);
        }
        int fila = tblClientesConsultarCliente.getSelectedRow();
        ContactoDAO mail = new ContactoDAO();
        ArrayList emails =
mail.buscarMails(String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0)));
        cmbEmailConsultarCliente.removeAllItems();
        for (int j = 0; j < emails.size(); j++) {
            contactoCliente detF = new contactoCliente();
            detF = (contactoCliente) emails.get(j);
            cmbEmailConsultarCliente.addItem(detF.getEmail());
        }
    }else{
        JOptionPane.showMessageDialog(null, "Porfavor seleccione un elemento de la
tabla");
    }
}

```

Al presionar el botón Guardar:

```

if (!txtNombreConsultarCliente.getText().equals("")) {
    if (!txtApellidoConsultarCliente.getText().equals("")) {
        if (!txtGiroConsultarCliente.getText().equals("")) {
            if (!txtDireccionConsultarCliente.getText().equals("")) {

```

```

        cliente cli = new cliente();
        comuna com = new comuna();
        ClienteDAO cDAO = new ClienteDAO();
        cli.setRutCliente(txtRutConsultarCliente.getText());
        int          codigo          =          com.getCodigoComuna((String)
cmbComunaConsultarCliente.getSelectedItem());
        cli.setCodComuna((int) codigo);
        cli.setNombre(txtNombreConsultarCliente.getText());
        cli.setApellido(txtApellidoConsultarCliente.getText());
        cli.setGiro(txtGiroConsultarCliente.getText());
        cli.setDireccion(txtDireccionConsultarCliente.getText());
        cDAO.modificar(cli);
    } else {
        JOptionPane.showMessageDialog(null, "El campo DIRECCION está
VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "El campo GIRO está VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "El campo APELLIDO está
VACIO");
    }
    } else {
        JOptionPane.showMessageDialog(null, "El campo NOMBRE está VACIO");
    }
    }
}

```

ELIMINAR CLIENTE

Al presionar el botón Eliminar:

```

private void btnEliminarConsultarClienteActionPerformed(java.awt.event.ActionEvent
evt) {

```

```

if (!txtRutConsultarCliente.getText().equals("")) {
    int fila = tblClientesConsultarCliente.getSelectedRow();
    String rut = String.valueOf(tblClientesConsultarCliente.getValueAt(fila, 0));
    int resp = JOptionPane.showConfirmDialog(null, "Borrará el cliente Rut " + rut
+ " ,se eliminarán sus reservas y cuotas asociadas ¿Continuar?");
    if (resp == 0) {
        ClienteDAO CDAO = new ClienteDAO();
        CDAO.eliminar(rut);
    }
    ClienteDAO cDAO = new ClienteDAO();
    cDAO.buscarTodo(tblClientesConsultarCliente);
    txtRutConsultarCliente.setText("");
    txtComunaConsultarCliente.setText("");
    txtRegionConsultarCliente.setText("");
    txtNombreConsultarCliente.setText("");
    txtApellidoConsultarCliente.setText("");
    txtGiroConsultarCliente.setText("");
    txtDireccionConsultarCliente.setText("");
    cmbTelefonoConsultarCliente.removeAllItems();
    cmbEmailConsultarCliente.removeAllItems();
} else {
    JOptionPane.showMessageDialog(null, "Profavor seleccione un cliente");
}
}

```

INGRESAR CUOTAS

Al presionar la tecla ENTER en el rut del cliente:

```

private void txtRutVentaBoletaKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        txtDVVentaBoleta.setText("");
        if (txtRutVentaBoleta.getText().length() == 8) {
            Validaciones val = new Validaciones();
            txtDVVentaBoleta.setText(val.sacarDV(txtRutVentaBoleta.getText()));
        }
    }
}

```

```

String rut = txtRutVentaBoleta.getText();
rut = rut.concat("-");
rut = rut.concat(txtDVVentaBoleta.getText());
cliente cli = new cliente(rut);
txtNombreClienteVentaBoleta.setText(cli.getNombre());
if (cli.getNombre().equals("")) {
    int resp = JOptionPane.showConfirmDialog(null, "El rut ingresado no
pertenece a ningún cliente, Es necesario para guardar las cuotas, ¿Desea ir al panel de
ingreso CLIENTE?");
    if (resp == 0) {
        txtRutIngresarCliente.setText(txtRutVentaBoleta.getText());
        txtDVIngresarCliente.setText(txtDVVentaBoleta.getText());
        desplegar(pnlIngresarCliente);
    }
}
} else {
    JOptionPane.showMessageDialog(null, "El rut debe contener 8 digitos");
}
}
}
}

```

Al precionar la tecla ENTER en la cantidad de cuotas:

```

private void txtCuotasVentaBoletaKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        if (!txtTotalVentaBoleta.getText().equals("0")) {
            if (!txtCuotasVentaBoleta.getText().equals("")) {
                float resultado = (float)
Math rint((Integer.parseInt(txtTotalVentaBoleta.getText())) /
(Integer.parseInt(txtCuotasVentaBoleta.getText())));
                txtValorCuotaVentaBoleta.setText(Float.toString(resultado));
            }
        } else {
            JOptionPane.showMessageDialog(null, "el detalle debe tener almenos un
elemento");
        }
    }
}

```

```
}
```

Al aceptar la venta con cuotas:

(Para boleta o factura el procedimiento es el mismo).

```

        int numFactura;
        int total;
        boolean MedioPago = chkCuotasVentaBoleta.isSelected();
        String medioPago;
        numFactura
Integer.parseInt(txtNumeroBoletaVentaBoleta.getText());
        total = Integer.parseInt(txtTotalVentaBoleta.getText());
        if (MedioPago) {
            medioPago = "1";
        } else {
            medioPago = "0";
        }

        Boleta bol = new Boleta();
        bol.setNumBoleta(numFactura);
        bol.setTotal(total);
        bol.setMedioPago(medioPago);
        bol.setFecha();
        ArrayList detalle = new ArrayList<DetalleBoleta>();
        int registros = tblProductosVentaBoleta.getRowCount();
        for (int i = 0; i < registros; i++) {
            DetalleBoleta det = new DetalleBoleta();
            det.setNumBoleta(numFactura);

det.setCodProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValue
At(i, 0))));

det.setCantProducto(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValue
At(i, 2))));

det.setValorUnitario(Integer.parseInt(String.valueOf(tblProductosVentaBoleta.getValue
At(i, 3))));

```

```

        detalle.add(det);
    }
    String rutCliente = txtRutVentaBoleta.getText();
    rutCliente = rutCliente.concat("-");
    rutCliente = rutCliente.concat(txtDVVentaBoleta.getText());
    VentaFactura vf = new VentaFactura();
    Cuotas cu = new Cuotas();
    cu.setCodTarjeta(vf.cod_tarjeta((String)
cmbTarjetasVentaBoleta.getSelectedItem()));

cu.setCodRecibo(Integer.parseInt(txtNumeroBoletaVentaBoleta.getText()));
    cu.setTipoRecibo("B");
    cu.setValor((int)
Float.parseFloat(txtValorCuotaVentaBoleta.getText()));
    cu.setFecha(bol.getFecha2());
    cu.setRutCliente(rutCliente);
    cu.setNumCuota(0);

cu.setTotalCuotas(Integer.parseInt(txtCuotasVentaBoleta.getText()));
    BoletaDAO bolDAO = new BoletaDAO();
    bolDAO.insertarC(bol, detalle, cu);

```

INGRESAR RESERVA

Al presionar la tecla ENTER en el rut del cliente:

```

private void txtRutClienteReservarProductoKeyPressed(java.awt.event.KeyEvent evt)
{
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        if(txtRutClienteReservarProducto.getText().length() == 8){
            String rutC = txtRutClienteReservarProducto.getText();
            Validaciones val = new Validaciones();
            txtDVClienteReservarProducto.setText(val.sacarDV(rutC));
            rutC = rutC.concat("-");
            rutC = rutC.concat(txtDVClienteReservarProducto.getText());

```

```

        cliente cli = new cliente(rutC);
        txtClienteReservarProducto.setText(((cli.getNombre()).concat("
    ")).concat(cli.getApellido()));
        if (txtClienteReservarProducto.getText().equals(" ")){
            int resp = JOptionPane.showConfirmDialog(null, "El rut ingresado no
pertenece a ningún cliente, ¿Desea ir al panel de ingreso?");
            if(resp == 0){
                txtRutIngresarCliente.setText(txtRutClienteReservarProducto.getText());
                txtDVIngresarCliente.setText(txtDVClienteReservarProducto.getText());
                desplegar(pnlIngresarCliente);
            }
        }
        }else{
            JOptionPane.showMessageDialog(null, "El rut ingresado debe conetner 8
dígitos?");
        }
    }
}

```

Al presionar la tecla ENTER en el rut del usuario:

```

private void txtRutUsuarioReservarProductoKeyPressed(java.awt.event.KeyEvent
evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        if(txtRutUsuarioReservarProducto.getText().length() == 8){
            String rutU = txtRutUsuarioReservarProducto.getText();
            Validaciones val = new Validaciones();
            txtDVUsuarioReservarProducto.setText(val.sacarDV(rutU));
            rutU = rutU.concat("-");
            rutU = rutU.concat(txtDVUsuarioReservarProducto.getText());

            UsuarioDAO usrDAO = new UsuarioDAO();

            Usuario usr = usrDAO.buscar(rutU);
            txtUsuarioReservarProducto.setText(usr.getNombre());
            if (txtUsuarioReservarProducto.getText().equals("")){

```

```

        int resp = JOptionPane.showConfirmDialog(null, "El rut ingresado no
pertenece a ningún Usuario, ¿Desea ir al panel de ingreso?");
        if(resp == 0){
            txtRutIngresarUsuario.setText(rutU.substring(0,8));
            txtDVIngresarUsuario.setText(rutU.substring(9));
            desplegar(pnlIngresarUsuario);
        }
    }
}
}else{
    JOptionPane.showMessageDialog(null, "El rut ingresado debe conetner 8
dígitos?");
}
}
}
}

```

Al presionar ENTER en alguna de las filas del detalle de la reserva:

```

private void tblProductoReservarProductoKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        int fila = tblProductoReservarProducto.getSelectedRow();
        if
            (!(String.valueOf(tblProductoReservarProducto.getValueAt(fila,
0))).equals("")) && !(String.valueOf(tblProductoReservarProducto.getValueAt(fila,
2))).equals("")) {
            String valor1, valor2;
            valor1 = String.valueOf(tblProductoReservarProducto.getValueAt(fila, 0));
            valor2 = String.valueOf(tblProductoReservarProducto.getValueAt(fila, 2));
            Validaciones val = new Validaciones();
            if (val.numeros(valor1, valor2)) {
                if (valor2.length() <= 4) {
                    String nombre = "";
                    producto prod = new producto();
                    int
                        codigo
                        =
Integer.parseInt(String.valueOf(tblProductoReservarProducto.getValueAt(fila, 0)));
                    int
                        CA
                        =
Integer.parseInt(String.valueOf(tblProductoReservarProducto.getValueAt(fila, 2)));
                    if
                        (!(String.valueOf(tblProductoReservarProducto.getValueAt(fila,
1))).endsWith("*")) {

```

```

producto prod2 = prod.buscar(codigo);
nombre = prod2.getNombre();
if (!nombre.equals("")) {
    String estado = "";
    int CP = prod2.getStockActual();
    int resultado = CP - CA;
    if (CP != 0) {
        if (resultado >= 0) {
            estado = "EN BODEGA";
        }
        if (resultado < 0) {
            int rneg = CA + resultado;
            JOptionPane.showMessageDialog(null, "Del producto
CODIGO = " + codigo + " hay SOLO " + rneg + " en bodega");
            tblProductoReservarProducto.setValueAt(*rneg*/CA, fila, 2);
            estado = "EN PROCESO";
        }
    } else {
        JOptionPane.showMessageDialog(null, "NO HAY existencia en
bodega para el producto CODIGO = " + codigo);
        estado = "EN PROCESO";
    }
    tblProductoReservarProducto.setValueAt(estado, fila, 3);
    int registros = tblProductoReservarProducto.getRowCount();
    boolean bodega = true;
    for (int i = 0; i < registros; i++) {
        if (bodega) {
            String nomb =
String.valueOf(tblProductoReservarProducto.getValueAt(i, 1));
            String std =
String.valueOf(tblProductoReservarProducto.getValueAt(i, 3));
            if (!std.equals("EN BODEGA")) {
                bodega = false;
            }
        }
    }
}
if (bodega) {

```

```

        txtEstadoReservarProducto.setText("EN BODEGA");
    } else {
        txtEstadoReservarProducto.setText("EN PROCESO");
    }
    String nom = nombre.concat("*");
    tblProductoReservarProducto.setValueAt(nom, fila, 1);
} else {
    JOptionPane.showMessageDialog(null, "No se ha encontrado ningún
producto con ese código");
}
}
} else {
    JOptionPane.showMessageDialog(null, "la cantidad debe ser inferior a
9999");
}
} else {
    JOptionPane.showMessageDialog(null, "Debe ingresar sólo números, y se
han detectado caracteres");
}
} else {
    JOptionPane.showMessageDialog(null, "Debe Ingresar CODIGO y
CANTIDAD");
}
}
}
}

```

Al presionar el botón Guardar Reserva:

```

private void btnGuardarReservarProductoActionPerformed(java.awt.event.ActionEvent
evt) {
    if (txtRutClienteReservarProducto.getText().length() == 8) {
        if (!txtDVClienteReservarProducto.getText().equals("")) {
            if (txtRutUsuarioReservarProducto.getText().length() == 8) {
                if (!txtDVUsuarioReservarProducto.getText().equals("")) {
                    if (!txtEstadoReservarProducto.getText().equals("")) {
                        if (!txtClienteReservarProducto.getText().equals("")) {
                            if (!txtUsuarioReservarProducto.getText().equals("")) {

```

```

Reserva r = new Reserva();
String rutC = txtRutClienteReservarProducto.getText();
rutC = rutC.concat("-");
rutC = rutC.concat(txtDVClienteReservarProducto.getText());
String rutU = txtRutUsuarioReservarProducto.getText();
rutU = rutU.concat("-");
rutU = rutU.concat(txtDVUsuarioReservarProducto.getText());
r.setRutCliente(rutC);
r.setRutUsuario(rutU);
r.setFecha();
if (txtEstadoReservarProducto.getText().equals("EN
PROCESO")) {
    r.setEstado("1");
}
if (txtEstadoReservarProducto.getText().equals("EN BODEGA"))
{
    r.setEstado("2");
}
ReservaDAO rdao = new ReservaDAO();
rdao.insertar(r);
r = rdao.buscar(r.getFecha2(), r.getRutCliente());
int codigo = r.getCodReserva();
ArrayList detalle = new ArrayList<DetalleReserva>();
int registros = tblProductoReservarProducto.getRowCount();
for (int i = 0; i < registros; i++) {
    DetalleReserva det = new DetalleReserva();
    det.setCodReserva(codigo);

det.setCodProducto(Integer.parseInt(String.valueOf(tblProductoReservarProducto.getVa
lueAt(i, 0))));

det.setCantidad(Integer.parseInt(String.valueOf(tblProductoReservarProducto.getValue
At(i, 2))));

    String estado = "";
    String est =
String.valueOf(tblProductoReservarProducto.getValueAt(i, 3));
    if (est.equals("EN PROCESO")) {

```

```

        estado = "1";
    }
    if (est.equals("EN BODEGA")) {
        estado = "2";
    }
    det.setEstado(estado);
    detalle.add(det);
}
DetalleReservaDAO detR = new DetalleReservaDAO();
detR.insertar(detalle);
DetalleReservaDAO DDAO = new DetalleReservaDAO();
DDAO.vaciarTabla(tblProductoReservarProducto);
txtEstadoReservarProducto.setText("");
} else {
    JOptionPane.showMessageDialog(null, "Profavor acepte usuario
valido");
}
} else {
    JOptionPane.showMessageDialog(null, "Profavor acepte cliente
valido");
}
} else {
    JOptionPane.showMessageDialog(null, "Lista de productos
incompleta");
}
} else {
    JOptionPane.showMessageDialog(null, "Acepte el rut de usuario
Precionando ENTER en él");
}
} else {
    JOptionPane.showMessageDialog(null, "El rut del usuario debe contener 8
dígitos");
}
} else {
    JOptionPane.showMessageDialog(null, "Acepte el rut de cliente Precionando
ENTER en él");
}
}

```

```
    } else {  
        JOptionPane.showMessageDialog(null, "El rut del cliente debe contener 8  
dígitos");  
    }  
}
```