

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

SISTEMA DE VENTAS Y CONTROL DE STOCK
PARA BOTILLERÍA EL NONO.

Trabajo de Titulación para optar al
Título de Técnico Universitario en
Informática.

Alumna:
Yessica Belén Gómez Silva.

Profesor Guía:
Dagoberto Cabrera Tapia.

RESUMEN

KEYWORDS: BOTILERÍA, SISTEMA DE VENTAS, CONTROL DE STOCK.

El presente trabajo de título, tiene por objetivo presentar el diseño e implementación de un sistema de escritorio que ayudará en las ventas y control de stock de la botillería "El Nono", ubicada en La Región de Valparaíso, Ciudad de La Calera. Este programa fue diseñado y desarrollado por el lenguaje Visual Basic.net. La base de datos está montada en el motor MySQL.

El objetivo de este sistema es otorgar una herramienta que les permita una mejor productividad al momento de atender un cliente y llevar actualizado el inventario de sus productos, el usuario objetivo de este software son los propios trabajadores de dicho local.

Este texto se organiza en 3 capítulos, cuyos contenidos comprenden los siguientes temas:

Capítulo 1: Descripción de la Organización. Se describe la situación actual y la problemática que se encuentra en la botillería, se detallan los problemas detectados que se han hablado con el usuario y en base a lo que él requiera y nuestras propias vistas del problema se presenta la solución propuesta en términos generales.

Capítulo 2: Se describirá el entorno de desarrollo, respecto al hardware y software que se utilizará para el desarrollo y futura implementación de tal programa. También se describirán las tablas de la base de datos, para así dar un mejor entendimiento y visión del problema en cuestión.

Capítulo 3. Descripción de programas. Esta es la parte en donde se muestra al usuario el avance funcional de tal sistema. Se describirán todas las funcionalidades del programa, para dar un mayor entendimiento se realizarán diagramas y se mostrarán las pantallas. Las cinco principales cualidades más importantes se describirán en detalle y se enseñará el uso de ellas.

Finalmente, se incluye la conclusión donde se explica la experiencia de la creación del sistema, la bibliografía en donde adquirimos el conocimiento necesario para hacer tal programa y parte del código en un anexo.

INDICE

RESUMEN

INTRODUCCIÓN	1
CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO	3
1.1 DESCRIPCIÓN DE LA ORGANIZACIÓN	4
1.2 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL	6
1.3 PROBLEMAS DETECTADOS	8
1.4 DESCRIPCIÓN DEL SISTEMA PROPUESTO	10
CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE LA BASE DE DATOS	20
2.1. DESCRIPCIÓN DE LAS CARACTERÍSTICAS DEL RECURSO COMPUTACIONAL	21
2.1.1. Configuración del Sistema	21
2.1.2. DESCRIPCIÓN DE TABLAS	23
CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS	29
3.1. Diagrama Modular	31
3.2. Diagrama de Menús	32
3.3. Listado de Programas	33
3.4. Descripción de Programas Seleccionados	34
CONCLUSIONES Y RECOMENDACIONES	48
BIBLIOGRAFÍA	49
ANEXOS A: CÓDIGO FUENTE	50

INTRODUCCIÓN

En el día a día, ante la necesidad o el deseo de adquirir algún producto, nos dirigimos a comprar a muchas tiendas, donde esperamos recibir una rápida y excelente experiencia de compra por parte de quienes nos atienden. Hay veces que esto no es posible, y no necesariamente porque no nos quieran atender correctamente, sino también por una falta de optimización en los procesos de atención. Cuando esto sucede, comenzamos a analizar y nos volvemos más críticos con la tienda. Desde una mirada externa, idealizamos cómo podría mejorar la atención si se aplicaran ciertos cambios.

Fue así como llegamos a la botillería “El Nono (Referencia Frigura 1), donde conversamos con su dueña, la señora Elizabeth, quien nos permitió adentrarnos en su lugar de trabajo.

En un principio, logramos observar un conjunto de procesos que, a nuestro parecer, retrasaban los tiempos de compra por cliente, dentro del que destacamos la completa dependencia por los cuadernos donde llevan sus registros. Consideramos que utilizar un cuaderno para cada registro es peligroso y poco óptimo. Peligroso, por el riesgo a que se pierda información importante. Y poco óptimo, ya que tener la información dispersa genera desorden y complica la manipulación de ésta.

Después de observar lo ya mencionado, le planteamos a la señora Elizabeth analizar su botillería una vez más, esta vez, desde una mirada interna, sin idealizar y teniendo la posibilidad de aplicar los cambios directamente, a través de un sistema informático.

Nuestro objetivo será optimizar los tiempos de compra por cliente, concentrando toda la información que se encuentra dispersa en un único sistema.



Figura 1.

CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1.1 DESCRIPCIÓN DE LA ORGANIZACIÓN

Botillería “El Nono” está ubicada en calle Golondrina #479, Población Arboleda, comuna de La Calera, región de Valparaíso, Chile.

Es una empresa de la sociedad “Paycar”, compuesta por el Señor Álvaro Graneris y la Señora Elizabeth Bernal. Elizabeth es la dueña de la botillería, también se desempeña como empleada y administradora de ésta. Su contadora es la señora Isabel Díaz.

La botillería atiende todos los días de la semana y su horario de atención es desde las 11:00 am hasta la 01:00 am, exceptuando los días viernes y sábado, donde se atiende hasta las 03:00 am. Hay dos trabajadoras que tienen turnos “4x4”, es decir, trabajan 4 días y descansan 4 días.

Uno de los objetivos de esta botillería es entregar un servicio de ventas de productos, entre los que se destacan: hielo, cigarrillos, helados, confites, productos elaborados, carbón, abarros, etc. El objetivo más importante es brindar una atención de calidad a los clientes y ofrecerles precios económicos que se acomoden a sus bolsillos.

Esta botillería no posee ningún sistema informático, lo que hace difícil la venta y el control de stock de los productos. Las operaciones son registradas en papel, una práctica algo arriesgada, ya que se exponen a pérdida de información importante. Estos son algunos de los productos más vendidos en el local:



Figura 2.



Figura 3.

1.2 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

El método de ventas que actualmente posee la botillería es completamente manual. El proceso que se cumple es el siguiente: primero llega el cliente, le pide lo que desea comprar a la cajera (en caso de que sea menor de edad, no tendrá permitido comprar productos con alcohol ni cigarrillos), ella verifica la disponibilidad del producto y, en caso de que haya disponible, lo acerca hacia el cliente, luego busca el precio en el “Cuaderno de precios”, o bien, en el estante del producto, después, con una calculadora, efectúa la sumatoria de los precios de cada producto que lleva el cliente. Finalmente, al determinar el total, hace la boleta manual y se la entrega al cliente junto con los productos. En caso de que el cliente desee comprar “fiado” y sea autorizado, se registrará su compra en el “Cuaderno de deudores”, donde la boleta quedará pendiente, para evitar inconsistencias entre el cálculo total de ventas y el dinero total de caja.

- **Control de stock y cierre de caja:**

El siguiente proceso es más extenso, ya que, al final de cada turno, la trabajadora a cargo de la caja tiene que contar todas las boletas hechas en el día y, a la vez, todos los productos que vayan quedando en el local, esto último con el fin de registrarlo en el “Cuaderno de inventario”. El día y la hora en que los proveedores acuden a la botillería es relativo, por lo que es muy importante saber la cantidad (exacta o aproximada) de productos disponibles, para poder hacer un pedido acorde a lo que se necesita. Cuando termina de hacer el conteo de las boletas, tiene que hacer coincidir el total de boletas con el total de dinero en caja. Todos los totales de las boletas tienen que ser anotados en el “Libro diario”, este libro lo entrega el Servicio de Impuestos Internos (SII) y, una vez completado, debe ser firmado por dicho servicio.

En caso de que exista merma o pérdida, esta se registra en el “Cuaderno de inventario” con su respectivo detalle. Para el caso de los productos que se utilicen como consumo interno o se deseen regalar o donar, estos no se registran ni como merma ni como pérdida.

Para hacer los pedidos a los proveedores, se realiza otro proceso, el cual consiste en que la señora Elizabeth revisa el “Cuaderno de inventario”, con el fin de ir haciendo un listado de productos que pedirá. En caso de que llegue el vendedor y no esté la dueña, ella le deja el listado a la otra trabajadora, para que sepa qué productos pedir.



Figura 4.

1.3 **PROBLEMAS DETECTADOS**

El origen de este sistema informático surge a partir del problema más grave que pudimos detectar a modo general: el mal manejo de la información y a la ineficiente atención a los clientes. A continuación, señalaremos detalladamente cada uno de los problemas detectados.

- No hay un registro óptimo de los precios de venta de los productos. Si algún producto exhibido no está con su respectivo precio, se debe buscar en un libro físico.
- Cuando se requiere alguna información histórica, la búsqueda de esta se hace complicada, ya que realizarla implicaría buscar hoja por hoja y/o cuaderno por cuaderno (si existe más de uno) hasta encontrar la fecha deseada. En el peor de los casos, podría no existir registro físico de años anteriores.
- Al hacer el cierre de caja después de cada turno, se debe contar una por una todas las boletas hechas en el día y cuadrar el total de estas con el dinero existente en caja.
- Después de hacer la cuadratura, se debe separar la ganancia, específicamente, lo vendido en cigarrillos de lo vendido en el resto de productos.
- Cuando llega un cliente deudor, este debe ser registrado en el cuaderno de deudores junto con los productos que lleva y su respectivo precio.
- Para saber qué productos faltan y cuáles son los más vendidos se debe hacer un conteo de estos al finalizar cada turno.
- Cada vez que un cliente compra cigarrillos, la venta debe registrarse en otro cuaderno físico, ya que necesitan llevarla separada del resto de las ventas.
- Deben llevar un registro de cada proveedor que le vende a la botillería.

- No hay un control del consumo interno, por lo que se podrían extraviar algunos productos sin poder determinar su causa.
- La falta de comunicación entre la dueña y la trabajadora implica que se cometan más errores al momento de hacer el pedido al proveedor

Registro fotográfico del cuaderno en donde se mantiene la información de la botillería:

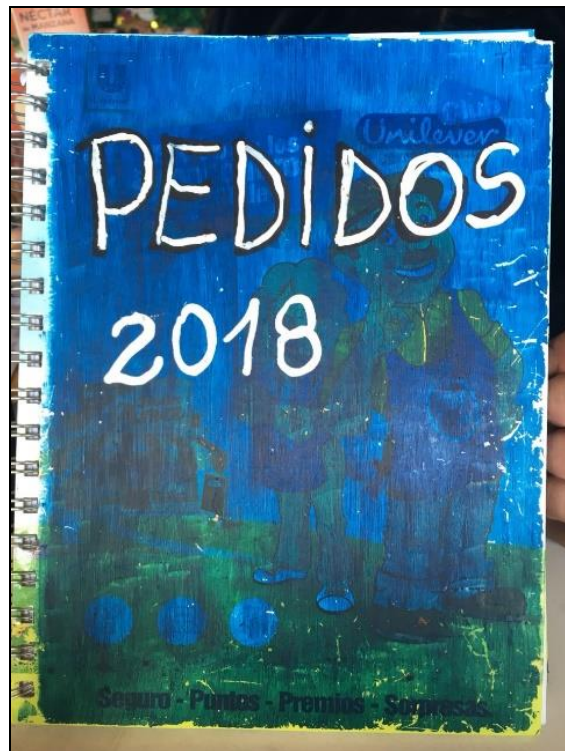


Figura 5.

1.4 DESCRIPCIÓN DEL SISTEMA PROPUESTO

El sistema de ventas y control de stock, busca mejorar el tiempo de espera que tienen los clientes al ir a comprar productos a la botillería y también tener un control de la información que se maneja en ella, es decir, todos los productos que están a la venta en el local, los proveedores existentes que abastecen la botillería y el registro de los clientes deudores.

Más adelante, se detallarán los objetivos, los beneficios, la descripción general y la estructura funcional del sistema propuesto.

1.4.1 Objetivos del Sistema Propuesto

El objetivo principal de este sistema es ordenar toda la información y agilizar el acceso a ésta, con el fin de brindarle una mejor experiencia, tanto a los trabajadores como a los clientes. Podrá ser una atención más rápida, personal y eficiente. Tendrán un control más seguro de sus productos y de los trabajadores.

1.4.2 Descripción General de la solución propuesta

Las soluciones más destacadas que proponemos para enfrentar los problemas que presenta este sistema son las siguientes:

- **Proceso de Ventas:**

- Este sistema logrará que mejore el tiempo de atención hacia los clientes, ya que ante cualquier solicitud el tiempo de respuesta de los trabajadores será más rápido. Tendrá el registro de los clientes deudores y será más rápido el abono a las cuentas.

- **Proceso de Pedidos:**

- Se hará más fácil y rápido hacer un pedido al proveedor correspondiente, ya que existirá una bitácora en donde estará la orden redactada por la dueña. También se implementará una alerta de los productos con bajo stock.

- **Proceso de Inventario:**

- Automáticamente se irán descontando o agregando productos al inventario, para así contar con un monitoreo más preciso de los productos existentes en la botillería. Podrá consultar el stock las veces que quiera del producto seleccionado.

- **Proceso de Cierre de Caja:**

- Se obtendrá un total de las ventas hechas en el transcurso del día, podrá visualizar las ventas históricas, en caso que se haya perdido alguna información o simplemente para sacar alguna estadística de ventas.

1.4.3 Objetivos específicos del Sistema:

- Crear un sistema seguro con identificación de usuario, el cual tendrá el usuario (Rut), y contraseña.
- Crear un menú para el administrador y para su trabajadora, con distintos privilegios.
- Llevar un registro de los productos vendidos y saber cuánto dinero hay en caja.
- Llevar un registro de las mermas, consumo interno e incrementos de productos, con su respectivo detalle.
- Llevar un registro de los clientes deudores, de los abonos y el detalle de su deuda.
- Llevar un registro de los proveedores que venden a la botillería.
- Alertar el stock crítico de productos de una manera rápida y sencilla, que sea visible constantemente.
- Hacer listados de productos vendidos por día, mes y año, para llevar un control más detallado de las ventas.
- Generar un listado de productos comprados a los proveedores, así podrá mantener un constante registro de cada proveedor y productos asociados a él.
- Llevar un registro de los productos vendidos por mes, para así poder sacar estadísticas de los productos más vendidos y facilitar el manejo de stock en fechas importantes en donde hay más flujo de ventas.

1.4.4 Beneficios del Sistema Propuesto

El mayor beneficio de este sistema será conseguir un menor tiempo de espera en atender a los clientes y mejorar el manejo de la información. Dentro de lo que podemos destacar:

- Rápido acceso al control de los productos.
- Obtención inmediata de información actual o histórica.
- Ahorro de tiempo en obtener el total de ventas diarias.
- Fácil acceso a información de proveedores.
- Ayudar a hacer más eficaz los comunicados de jefa a trabajadora.
- Obtención rápida de información de los deudores.
- Rápido acceso al inventario.
- Generación automática de alertas de productos faltantes.
- Rápida compra de productos a proveedor.

1.4.5 Estructura funcional del sistema

- **Venta:**

Este proceso está encargado de registrar cada compra efectuada en el local. Consta de los siguientes procesos:

- Registrar al usuario que hace la venta.
- Ingresar productos que el cliente desea llevar, mediante una búsqueda por nombre, código o categoría.
- Ingresar un cliente deudor, sólo si está registrado.
- Cuando la venta finalice se desplegará por pantalla la boleta con su respectivo detalle de compra (esta boleta es solo para el usuario) y al cliente se le entrega una boleta manual, ya que esa boleta es legal.

2. **Anular Venta:** Este proceso consiste en deshacer una venta determinada cuando un cliente se encuentre insatisfecho con su compra. Se debe regularizar la devolución de los productos al local y la devolución del dinero al cliente.

3. **Consultar Precio:** Este proceso consiste en obtener el precio de algún producto mediante una búsqueda por filtros de categoría, nombre o código.

4. **Cuadratura de Caja:** Este proceso consiste en desplegar por pantalla un informe con los movimientos de dinero durante un período de tiempo establecido por el usuario. Tendrá un inicio y un cierre de caja. Además, mostrará las ventas de los productos restantes y el total de todas las ventas diarias.

5. **Ajustes:** Este proceso está encargado de llevar la cuenta de todos los productos existentes en el local. Se detallará todo ajuste ingresado. Cuenta con las siguientes funcionalidades.

- Incremento de Productos: Permite aumentar el stock los productos que sean comprados a los proveedores.
- Ingreso de Consumo Interno: Permite rebajar el stock de algún producto, detallando su respectivo motivo.
- Ingresos de Mermas: Permite rebajar el stock de algún producto que no esté apto para la venta, detallando su respectivo motivo.
- Otros: Permite rebajar el stock de los productos por alguna razón catastrófica.

6. Mantenedores:

- Mantenedor de Productos: Permite agregar, modificar, consultar y eliminar los productos que lleguen como pedido.
- Mantenedor de Deudores: Permite agregar, modificar, eliminar o consultar algún cliente, dependiendo de si inicia una deuda, la aumenta o disminuye, o la cancela respectivamente.
- Mantenedor de Proveedores: Permite agregar, modificar, consultar y eliminar los proveedores que le venden al local.
- Gestor Administrativo: Permite agregar, modificar, dar de baja o eliminar algún usuario determinado.

7. Generación de Informes:

- **Reportes de merma:** Se llevará un listado de mermas con el nombre y precio de cada producto mermado y la razón por el cual se ingresa a la merma.
- **Alertas de stock crítico:** Permite conocer cuál o cuáles productos se encuentran con bajo stock (usuario determina el valor). Se avisará mediante una alerta al ingresar al sistema.
- **Reportes de consumo interno:** Se llevará un listado de los productos que se extraen de la tienda sin ser pagados.
- **Reportes de los productos más vendidos y menos vendidos:** Permite saber cuál o cuáles son los productos de mayor venta y menor venta. También contará con una consulta de ventas por fecha (indicada por usuario).
- **Reportes de usuario:** Le permite al administrador mostrar todos los usuarios existentes, con sus respectivos datos y permisos asociados.

- **Reportes de proveedores:** Permite mostrar los proveedores registrados por el usuario.
- **Reportes de deudores:** Permite mostrar los clientes deudores con su respectiva deuda.
- **Reporte de precios:** Permite mostrar por pantalla el nombre, descripción y precio algún producto solicitado. También puede ser una lista de todos los productos.

1.4.6. **Información que se manejará.**

1. **Entradas:**

- **Datos de Ventas:** (Rut de usuario, tipo de venta, total de venta de otros productos, total ventas, descuentos).
- **Datos de Usuario:** (Rut, nombre, celular, dirección, tipo de usuario).
- **Datos de Proveedores:** (Rut, nombre, empresa, celular).
- **Datos de Deuda:** (Código de Deuda, Rut de deudor, total de deuda, fecha, pagos saldados).
- **Datos de Deudores:** (Rut, código de deuda, nombre, celular, dirección).
- **Datos de Productos:** (Código, nombre, descripción, precio de compra, precio de venta, tipo de producto, stock actual, stock crítico).
- **Datos de Caja:** (Fecha Ingreso, Rut de usuario, fecha cierre, monto inicial, monto cierre, total caja).
- **Datos de Ventas:** (Código de boleta, código de venta, Rut de usuario, fecha, total de venta).
- **Datos de Compras:** (Código de producto, Rut de Proveedor, nombre de producto, cantidad, total, fecha).

2. Salidas:

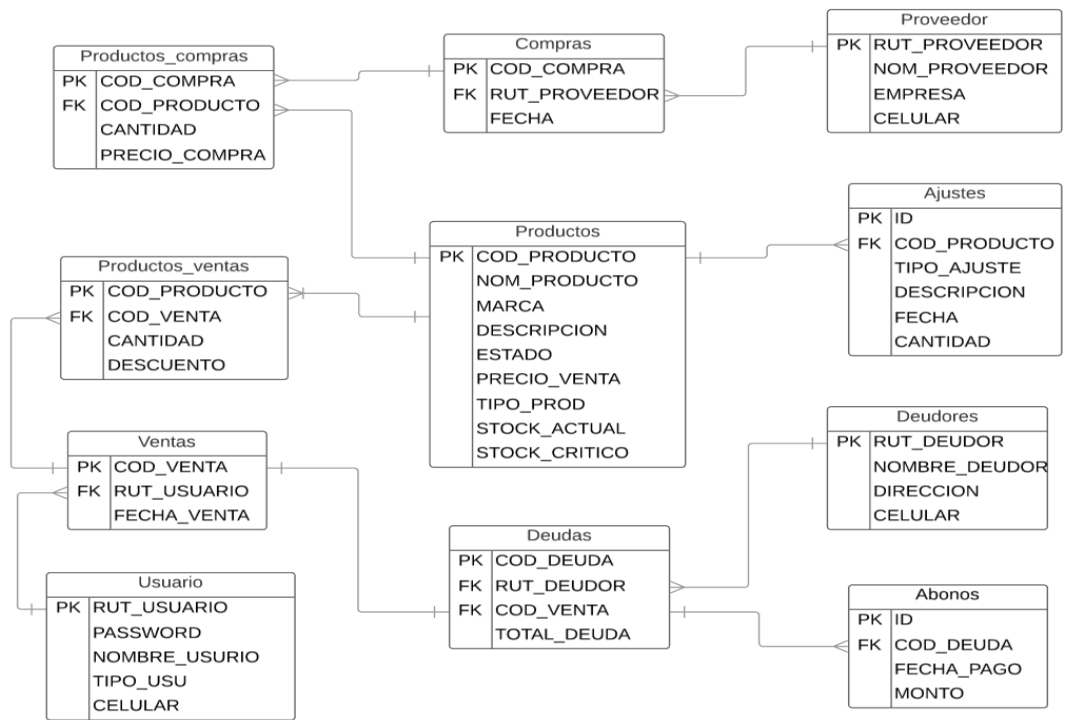
- **Lista de Clientes Deudores:** Se desplegará en pantalla una lista de las personas que tienen deudas pendientes.
- **Lista de Proveedores:** Se desplegará en pantalla una lista de los proveedores que venden a la botillería.
- **Informe de Caja:** Se desplegará por pantalla el total de ganancias por día.
- **Informe de Ventas:** Se desplegará por pantalla todos los productos vendidos por día o por el tramo que requiera el usuario.
- **Emisión de Boleta:** Se desplegará un vale por pantalla del total de la venta por cada cliente.
- **Emisión de Stock Crítico:** Se desplegará por pantalla los productos que tengan bajo stock.
- **Lista de Mermas:** Se desplegará por pantalla todas las mermas diarias.
- **Lista de Consumo Interno:** Se desplegará por pantalla la lista de productos diarios o de alguna fecha requerida por el usuario.

3. **Entidades de Información.**

- **Deudores:** Contiene datos de los clientes que “fían”.
- **Usuarios:** Contiene los datos de los trabajadores de la botillería.
- **Productos:** Contiene los datos de los productos que se encuentran en el local para su posterior venta.

Proveedores: Contiene los datos personales de los proveedores que venden productos a la botillería.

1.4.7 Modelo de Datos.



1.4.8 Estructura de Código.

Los códigos de barras son una combinación de letras o números que identifican un producto, en este sistema ese será su método para registrar un producto. Los largos de estos códigos son de 12 a 13 números enteros.

1.4.9 Condicionante de Diseños.

En la botillería utilizan un computador personal de escritorio stand alone, el cual cuenta actualmente con un procesador Intel Core i5-3570, 4GB de Memoria RAM, un Disco Duro de 500GB y Sistema Operativo Windows 7 Professional.

Para llevar a cabo este sistema, contaremos con Visual Basic .NET como lenguaje de programación, Microsoft Visual Studio como entorno de desarrollo integrado (IDE) y MySQL como gestor de base de datos.

CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE LA BASE DE DATOS

2.1. DESCRIPCIÓN DE LAS CARACTERÍSTICAS DEL RECURSO COMPUTACIONAL

Se detallarán los recursos computacionales que se utilizarán tanto en el desarrollo como en la implementación de este sistema.

2.1.1. Configuración del Sistema

Hardware utilizado para el desarrollo del sistema:

Equipo 1:

- Laptop HP
- Procesador AMD A6-5200.
- Memoria RAM 8 GB.
- Disco Duro 500 GB.
- Sistema Operativo Windows 10.

Hardware para la futura implementación del sistema:

Equipo 2:

- PC stand Alone Gear.
- Procesador Intel Core i5-3570.
- Memoria RAM 4 GB.
- Disco Duro 500 GB.
- Sistema Operativo Windows 7.

2.1.2 Software utilizado

- Sistema Operativo Windows 10.
- Lenguaje de Programación Visual Basic .NET:

Visual Basic .NET es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic.

- Entorno de Desarrollo Integrado Visual Studio:

Visual Studio es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, entre los cuales encontramos Visual Basic .NET

- Gestor de Base de Datos MySQL:

MySQL es un sistema de gestión de base de datos de código abierto, por lo que es gratuito. Su facilidad, rapidez, seguridad y eficiencia nos será de gran utilidad para el manejo de datos. Además, es compatible con Windows y Linux.

2.1.2. DESCRIPCIÓN DE TABLAS

Describiremos todas las tablas que utilizaremos en la base de datos para el futuro desarrollo del Sistema Computacional para Botillería “El Nono”.

2.2.1 Tipos de datos a utilizar

INT : Numérico de 4 bytes con valores de -2.147.483.648 a 2.147.483.647 o de 0 a 4.294.967.295.

SMALLINT : Numérico de 2 bytes con valores de -32.768 a 32.767 o de 0 a 65.535.

TINYINT : Numérico de 1 byte con valores de -128 a 127 o de 0 a 255.

BYTE : Numérico con valor de 0 bits.

VARCHAR : String de longitud variable de 1 a 255 caracteres.

DATE : Fecha que almacena año, mes, día, hora, minuto y segundo.

2.2.2 Tablas del Sistema

2.2.2.1. VENTAS:

Nombre : VENTAS.

Descripción : Esta tabla contiene los datos necesarios de las ventas.

Clave primaria : COD_VENTA.

Claves foráneas : RUT_USUARIO – Referencia tabla USUARIO.

Nombre	Tipo	Descripción
COD_VENTA	INT(11)	Contiene código venta
FECHA	DATE	Contiene fecha venta
RUT_USUARIO	VARCHAR(9)	Contiene Rut usuario

2.2.2.2 ABONOS.

Nombre : ABONOS.

Descripción : Esta tabla contiene los datos necesarios de los abonos.

Clave primaria : ID.

Claves foráneas : COD_DEUDA – Referencia tabla deudas.

Nombre	Tipo	Descripción
ID	INT(11)	Autonumérico
FECHA_PAGO	DATE	Contiene fecha de abono
MONTO	INT(11)	Contiene monto abono
COD_DEUDA	INT(11)	Contiene código deuda

2.2.2.3 USUARIO.

Nombre : USUARIO.

Descripción : Esta tabla contiene los datos personales de cada usuario.

Clave primaria: RUT_USUARIO.

Nombre	Tipo	Descripción
RUT_USUARIO	VARCHAR(9)	Contiene Rut usuario
PASSWORD	VARCHAR(6)	Contiene contraseña
NOMBRE_USUARIO	VARCHAR(30)	Contiene nombre usuario
TIPO_USUARIO	BYTE(1)	Administrador(0) , trabajador(1)
CELULAR	VARCHAR(12)	Contiene celular usuario

2.2.2.4 PRODUCTOS COMPRAS:

Nombre : PRODUCTOS_COMPRAS.

Descripción : Esta tabla contiene los datos de compras a proveedores.

Clave primaria: COD_COMPRA.

Clave Foránea : COD_PRODUCTO – Referencia tabla productos.

Nombre	Tipo	Descripción
COD_PRODUCTO	VARCHAR(13)	Contiene código de barra del producto
COD_COMPRA	INT(11)	Autonumérico.
CANTIDAD	INT(11)	Contiene cantidad de productos.
PRECIO_COMPRA	INT(11)	Contiene precio compra.

2.2.2.5 PRODUCTOS VENTAS

Nombre : PRODUCTOS_VENTAS.

Descripción : Esta tabla contiene los datos necesarios de las ventas.

Clave primaria: COD_PRODUCTO.

Clave Foránea : COD_VENTA – Referencia tabla Ventas.

Nombre	Tipo	Descripción
COD_PRODUCTO	VARCHAR(13)	Contiene código de barras del producto
COD_VENTA	INT(11)	Código de venta
CANTIDAD	INT(11)	Contiene cantidad de productos
DESCUENTO	INT(11)	Contiene descuento producto

2.2.2.6 DEUDAS:

Nombre : DEUDA.

Descripción : Esta tabla contiene datos de deudas entre tabla VENTAS Y DEUDOR.

Clave primaria : COD_DEUDA + COD_VENTA.

Claves foráneas: RUT_DEUDOR- Referencia tabla DEUDORES.

Nombre	Tipo	Descripción
COD_DEUDA	INT(11)	Código deuda
TOTAL_DEUDA	INT(11)	Total deuda
RUT_DEUDOR	VARCHAR(9)	Rut deudor
COD_VENTA	INT(11)	Código venta

2.2.2.7 DEUDORES.

Nombre : DEUDORES.

Descripción : Esta tabla almacena datos personales de los deudores.

Clave primaria: RUT_DEUDOR.

Nombre	Tipo	Descripción
RUT_DEUDOR	VARCHAR(9)	Rut deudor
NOM_DEUDOR	VARCHAR(30)	Nombre deudor
DIRECCION	VARCHAR(20)	Dirección deudor
CELULAR	VARCHAR(12)	Celular deudor

2.2.2.8 PRODUCTOS.

Nombre : PRODUCTOS

Descripción : Esta tabla almacena los datos de los Productos.

Clave primaria: COD_PRODUCTO.

Nombre	Tipo	Descripción
COD_PRODUCTO	VARCHAR(13)	Código de barra del producto.
NOM_PRODUCTO	VARCHAR(30)	Nombre del producto.
MARCA	VARCHAR(20)	Descripción de la marca del producto.
DESCRIPCION	VARCHAR(255)	Descripción del producto.
ESTADO	TINYINT	Descripción del estado del producto (0: No vigente, 1: Vigente).
PRECIO_VENTA	INT	Precio de venta del producto.
TIPO_PRODUCTO	VARCHAR(15)	Tipo de producto.
STOCK_ACTUAL	SMALLINT	Stock actual del producto.
STOCK_CRITICO	SMALLINT	Stock crítico del producto.

2.2.2.9 COMPRAS:

Nombre : COMPRAS.

Descripción : Esta tabla contiene los datos de las compras efectuadas a los proveedores.

Clave primaria: COD_PRODUCTO + FECHA.

Claves foráneas: RUT_PROVEEDOR – Referencia tabla PROVEEDOR.

Nombre	Tipo	Descripción
COD_COMPRA	INT(11)	Código compra
FECHA	DATE	Fecha venta
RUT_PROVEEDOR	VARCHAR(9)	Rut Proveedor

2.2.2.10 PROVEEDOR:

Nombre : PROVEEDOR.

Descripción : Esta tabla contiene los datos personales de los proveedores.

Clave primaria: RUT_PROVEEDOR.

Nombre	Tipo	Descripción
RUT_PROVEEDOR	VARCHAR(9)	Rut proveedor
NOM_PROVEEDOR	VARCHAR(30)	Nombre proveedor
EMPRESA	VARCHAR(20)	Nombre empresa
CELULAR	VARCHAR(12)	elular proveedor

2.2.2.11 AJUSTES:

Nombre : AJUSTES.

Descripción : Esta tabla almacena datos de posibles ajustes en el stock.

Clave primaria: COD_PRODUCTO + FECHA_AJUSTE.

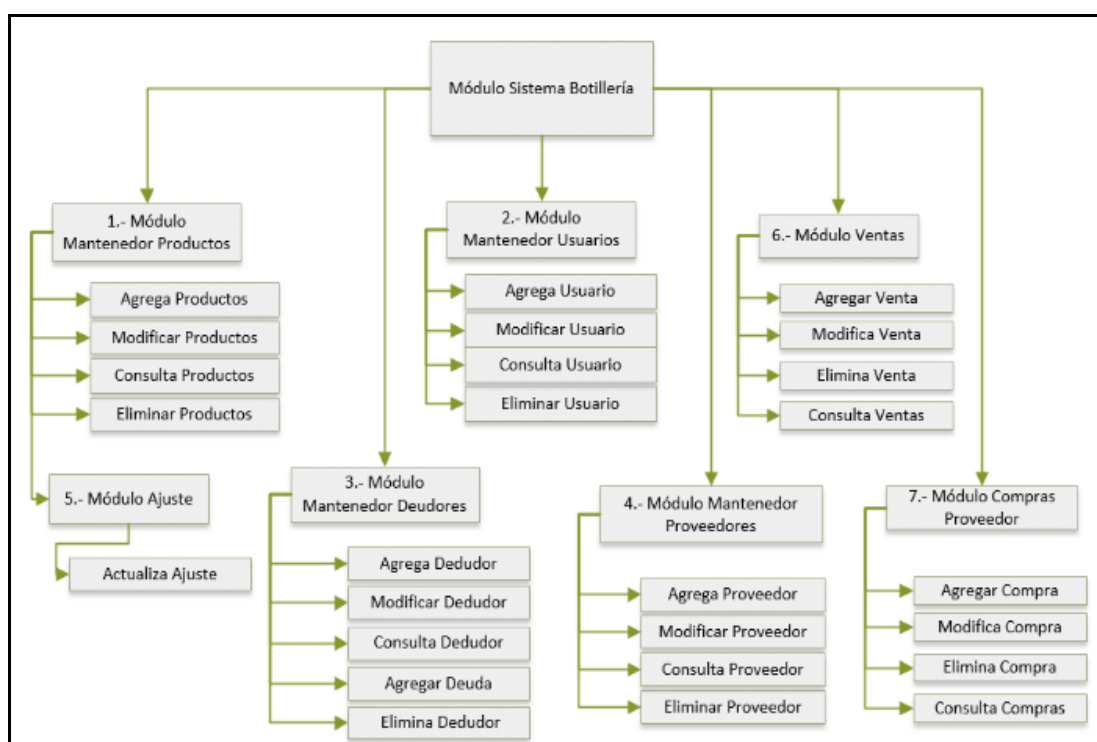
Nombre	Tipo	Descripción
ID	INT(11)	Auto numérico
TIPO_AJUSTE	BYTE(1)	Contiene tipo ajuste (merma (0), incremento productos (1), consumo interno (2), otros(3))
DESCRIPCION	VARCHAR(255)	Descripción de ajuste.
FECHA	DATE	Fecha ajuste.
CANTIDAD	INT(11)	Cantidad ajustada.
COD_PRODUCTO	VARCHAR(13)	ódigo de barra del producto.

CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCIÓN DEL RECURSO COMPUTACIONAL

A continuación, se mostrará el diagrama modular y de menú, para obtener una presentación gráfica y más estructurada del sistema.

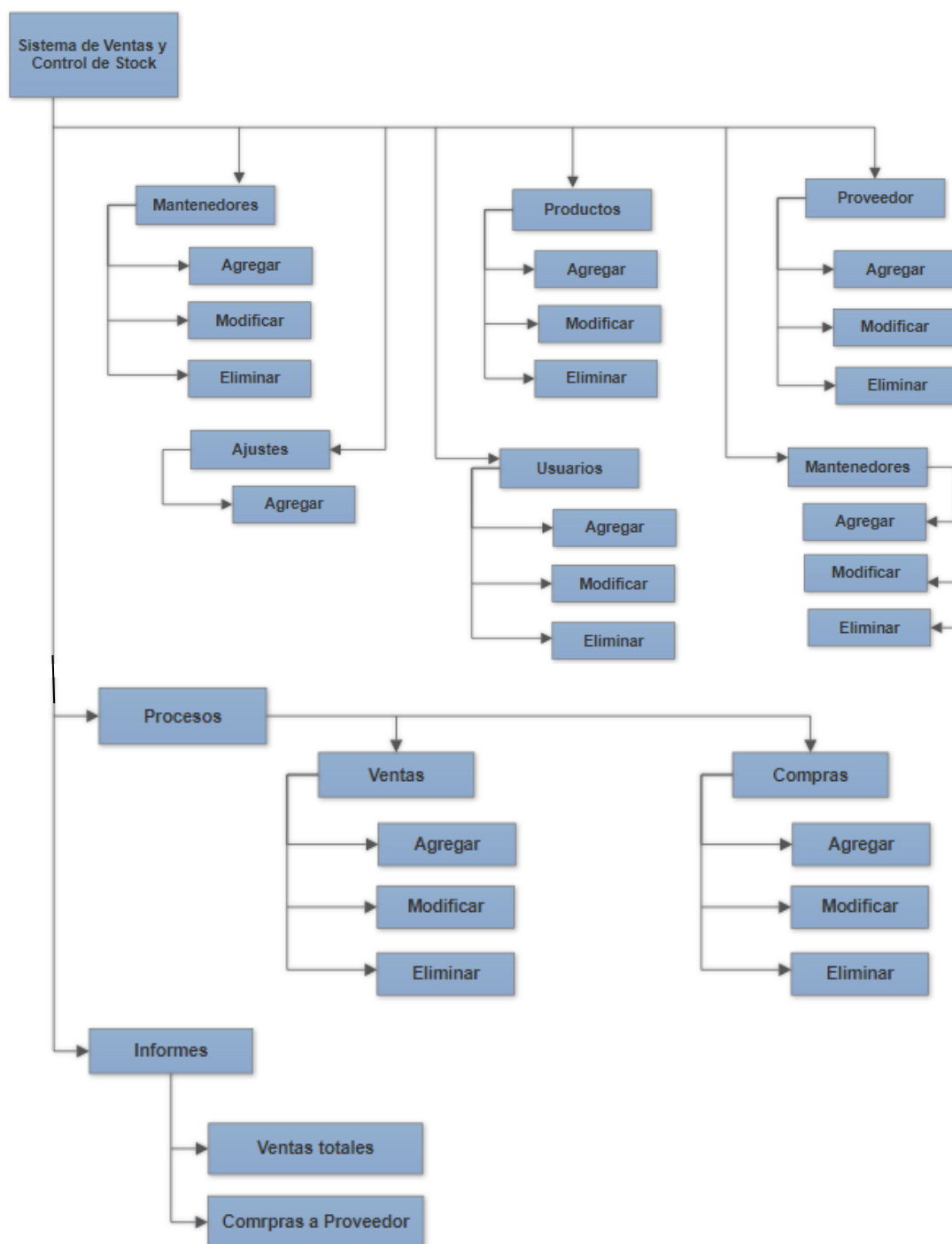
3.1. Diagrama Modular



Fuente: Elaboración Propia.

Figura 7.

3.2. Diagrama de Menús



Fuente: Elaboración Propia.

Figura 8.

3.3. Listado de Programas

Se describirán brevemente cada funcionalidad del sistema. Los que tienen (*), se describirán detalladamente en el siguiente punto.

Nombre	Objetivo
Login (*)	Validar el ingreso de usuario, mediante su Rut y contraseña.
Ajustes (*)	Permite hacer cambios de stock y precios.
Compras Proveedor(*)	Permite comprar productos a los proveedores (mostrando sus datos). Se facilita el stock de todos los productos de la botillería para así mejorar el pedido. También saca la suma total del pedido, para así después poder cancelar. Se agregan los nuevos productos comprados y se actualiza de forma inmediata el nuevo stock de cada uno.
Listar Compras	Entrega al trabajador una lista de todos los productos comprados a sus respectivos Proveedores, les muestra toda la información requerida para tener un seguimiento de precios.
Deuda	Permite agregar una deuda a un cliente que compró fiado.
Deudas_Clientes	Muestra toda la información del cliente deudor. Se pueden ver los abonos y deuda total.
Informe Compras	Muestra toda la información de las ventas que los proveedores le han hecho al local. Todo el detalle se puede buscar por año
Informe Ventas	Muestra todas las ventas hechas en el día, también se puede filtrar por otros días, mes o año. Indica que ventas se han fiado y quien es el deudor.
Informes	Es el menú para ingresar a los informes(Informe de Ventas, Compras de Productos)
Principal (*)	Es el menú inicial para poder ingresar a las ventas, compras, mantenedores o informes.
Mantenedores	Es el menú el cual da las opciones para ingresar a los mantenedores (productos, proveedores, deudores, usuarios, ajustes).
Deudores	Permite agregar, eliminar y actualizar a clientes deudores. También se puede ver la deuda en caso que exista.

Productos (*)	Permite agregar, eliminar y actualizar los productos que abastece la botillería.
Proveedores	Permite agregar, modificar, eliminar y actualizar los proveedores que trabajan con la botillería.
Usuarios	Permite agregar, eliminar y actualizar los usuarios que trabajan en la botillería. Esto solo lo puede hacer la Administradora.
Ventas (*)	Esta funcionalidad permite hacer las ventas de los productos de la botillería, automáticamente se puede dejar como fiado solo a los clientes inscritos.
Listar Ventas	Muestra las ventas según día, mes y año. Se permite eliminar una venta, también se muestra el detalle de cada una.

3.4. Descripción de Programas Seleccionados

3.4.1 Login y Menú Principal

Diagrama de Bloques:

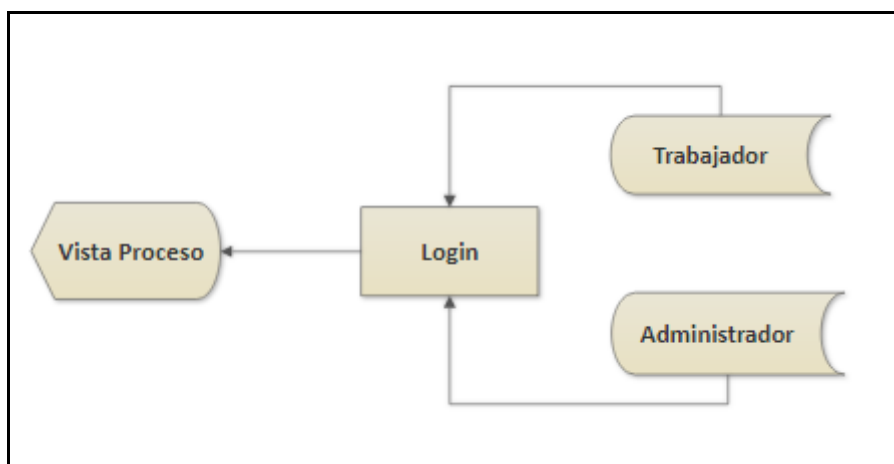
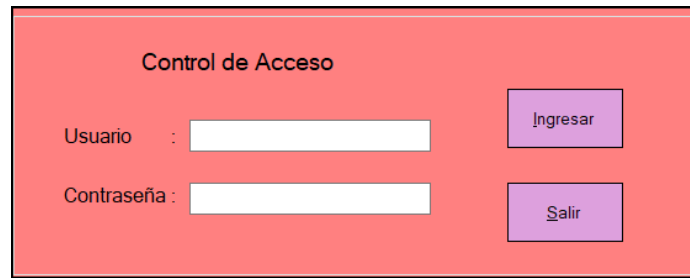


Figura 9.

Nombre: Login y Menú Principal.

Objetivos: Mediante un control de acceso, permitir el ingreso de las trabajadoras de la botillería a la página principal del sistema.

Reglas del Proceso: Al teclear el Rut y contraseña del usuario, permite ingresar al uso total del sistema referencia Figura 10. En caso que el usuario no se encuentre registrado en la base de datos saldrá un mensaje de error diciendo que "El usuario no se encuentra registrado", referencia Figura 10.

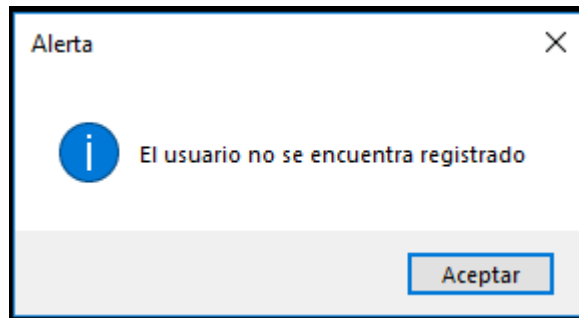


Control de Acceso

Usuario :

Contraseña :

Figura 10.

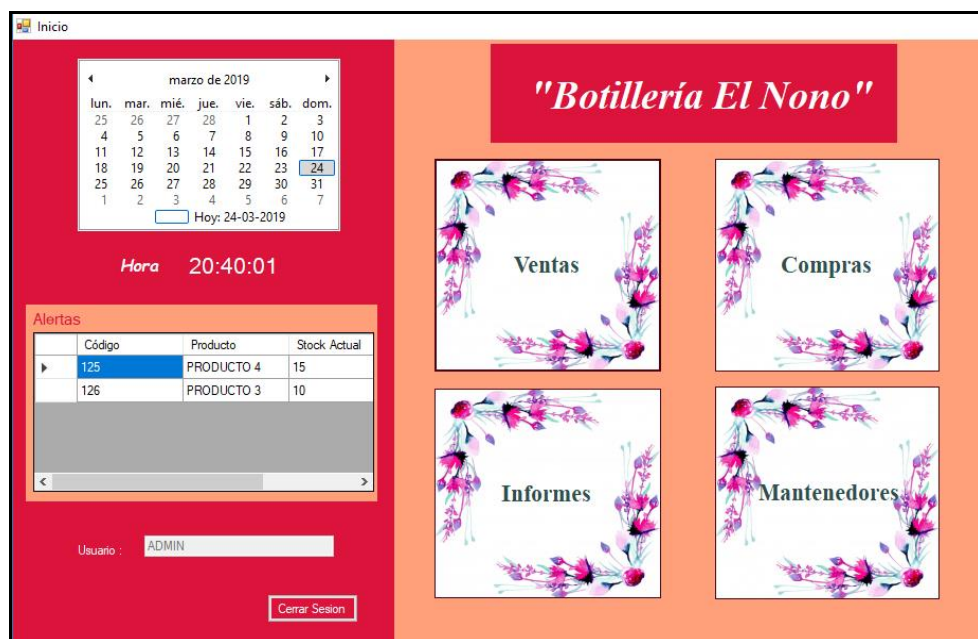


Alerta [X]

i El usuario no se encuentra registrado

Figura 11.

Si el ingreso del usuario es exitoso, se abra una nueva ventana la cual será el menú principal de tal sistema referencia Figura 12.



Inicio

marzo de 2019

lun.	mar.	mié.	jue.	vie.	sáb.	dom.
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Hoy: 24-03-2019

Hora 20:40:01

Alertas

	Código	Producto	Stock Actual
▶	125	PRODUCTO 4	15
	126	PRODUCTO 3	10

Usuario :

"Botillería El Nono"



Ventas



Compras



Informes



Mantenedores

Figura 12.

La página principal cuenta con el acceso a ventas, compras, informes y mantenedores, también indica la fecha y hora del día, cuenta con una funcionalidad de "ALERTAS" la cual indica los productos que están con stock crítico. Dependiendo los permisos que tenga cada personal de la botillería se van

desbloqueando la ventanas emergentes. El administrador tiene acceso total, pero el trabajador tiene acceso a las ventas, compras, informes, mantenedor de productos y ajustes.

Referencia código fuente:Anexo página 65.

3.4.2 Ajustes

Diagrama de Bloques:

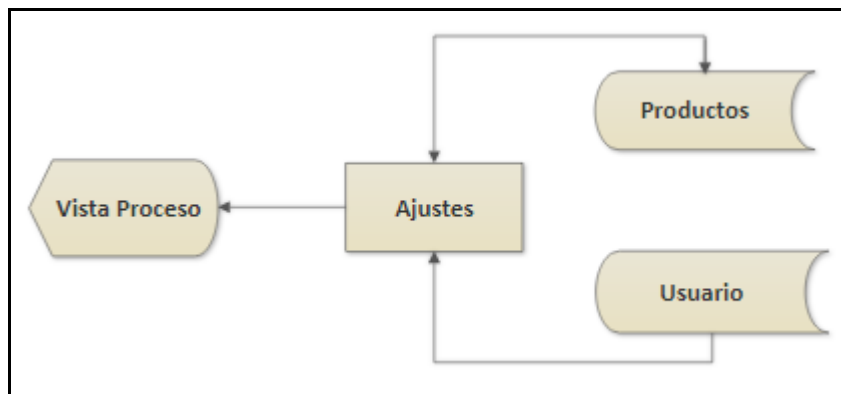


Figura 13.

Nombre: Ajustes

Objetivos: Realizar descuentos de stock asociada a una razón.

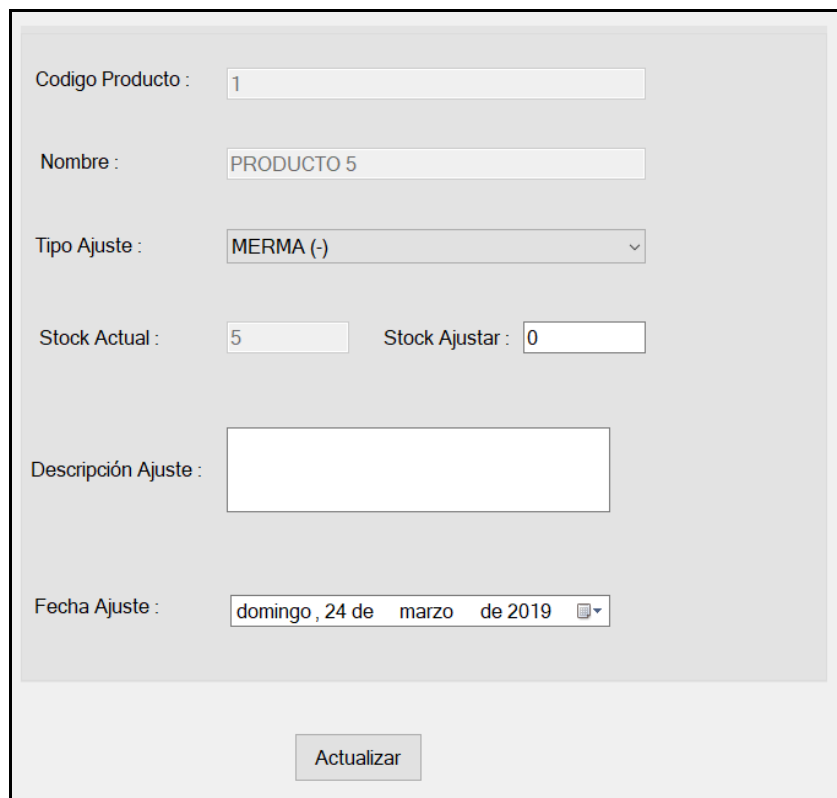
Reglas del Proceso: Para poder utilizar esta funcionalidad deberá ingresar al menú de Mantenedores referencia Figura 14.

Reglas del Proceso:



Figura 14.

Los ajustes lo pueden utilizar los administradores y trabajadores, no tiene restricciones. A continuación se explicará el uso de tal funcionalidad.



Formulario de ajuste de stock con los siguientes campos:

- Código Producto : 1
- Nombre : PRODUCTO 5
- Tipo Ajuste : MERMA (-)
- Stock Actual : 5 Stock Ajustar : 0
- Descripción Ajuste : (campo vacío)
- Fecha Ajuste : domingo, 24 de marzo de 2019
- Botón: Actualizar

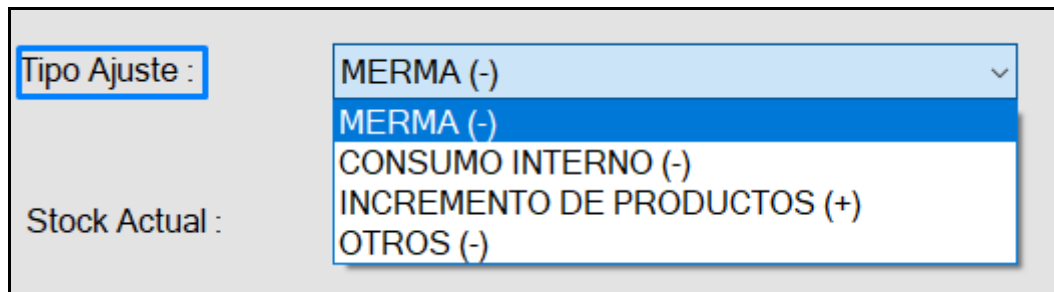
Figura 15.

Al seleccionar un producto se cargan automáticamente los textbox con el código producto, nombre producto y stock actual, referencia Figura 15-16. Estos se seleccionan de la DataGridView que se encuentra al lado del formulario.

Productos			
	COD_PRODUCTO	NOM_PRODUCTO	STOCK_ACTUAL
▶	1	PRODUCTO 5	8
	123	PRODUCTO 1	11
	124	PRODUCTO 2	19
	125	PRODUCTO 4	15
	126	PRODUCTO 3	10

Figura 16.

Para poder hacer un ajuste se necesita seleccionar algun tipo de ajuste, los cuales son Merma (-), Consumo Interno (-), Incremento de Productos (+) y Otros (-). Referencia Figura 17.

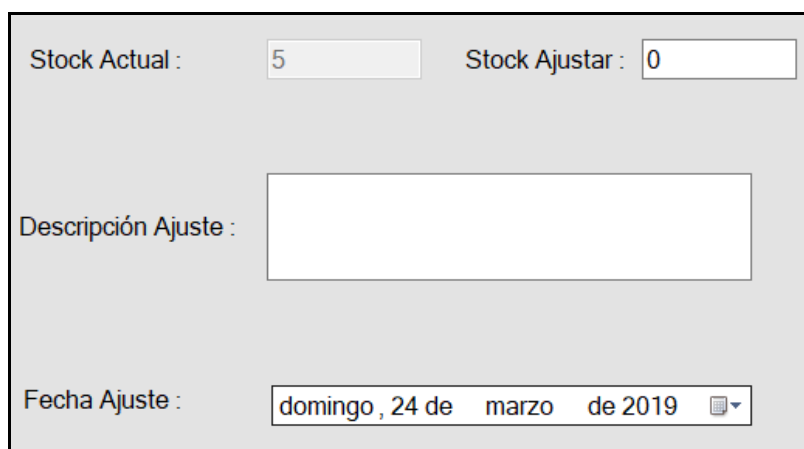


The image shows a software interface with a label 'Tipo Ajuste :' and a dropdown menu. The dropdown menu is open, displaying five options: 'MERMA (-)', 'MERMA (-)', 'CONSUMO INTERNO (-)', 'INCREMENTO DE PRODUCTOS (+)', and 'OTROS (-)'. The first two options are highlighted in blue. To the left of the dropdown, the text 'Stock Actual :' is visible.

Figura 17.

- Merma: Se indica cuando algo ya venció, se rompió o el producto esta sin sello.
- Consumo Interno: Se indica cuando la persona saca algo sin pagarlo para consumo propio.
- Incremento de Productos: Se indica cuando hay mas stock del normal, o cuando los proveedores “regalan” mercadería por alguna oferta.
- Otros: Se indica cuando ya es pérdida por algun fenómeno natural.

Luego de seleccionar el “Tipo de Ajuste” se debe ingresar la cantidad de stock a modificar en el textbox “Stock Ajustar”, y se agrega una pequeña descripción del ajuste, referencia Figura 18.



The image shows a software form for stock adjustment. It contains four main sections: 'Stock Actual :' with a text box containing the number '5'; 'Stock Ajustar :' with a text box containing the number '0'; 'Descripción Ajuste :' with a large empty text area; and 'Fecha Ajuste :' with a date picker showing 'domingo, 24 de marzo de 2019' and a small calendar icon.

Figura 18.

La fecha se guarda automáticamente en la base de datos. Todos los ajustes quedan guardados en la tabla "Ajustes" de la base de datos, pero para visualizarlos se muestran en una DataGridView referencia Figura 19, la cual se muestra el tipo de producto, cantidad, fecha y descripción.

Ajustes				
	tipo_ajuste	cantidad	fecha	descripcion
▶	MERMA	10	08-12-2018	Se rompió
	Consumo Interno	1	07-12-2018	test 123

Figura 19.

Referencia código fuente:Anexo página 60.

3.4.3 Mantenedor Productos

Diagrama de Bloques:

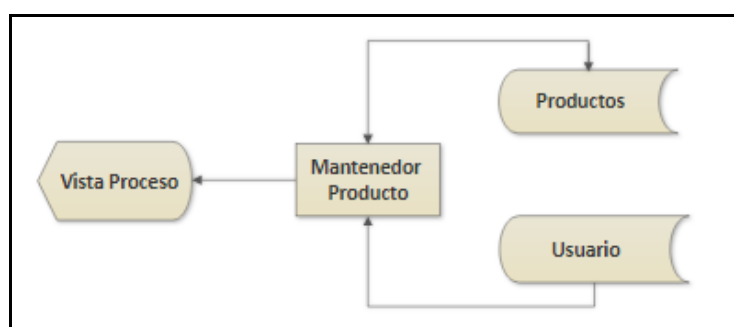


Figura 20.

COD_PRODUCTO	NOM_PRODUCTO	MARCA	DESCRIPCION	ESTADO	PRECIO_VENTA	TIPO_PROD	STOCK_ACTUAL	STOCK_CRITICO
1	PRODUCTO 5	MARCA E	DESCRIP PROD 5	DISPONIBLE	123	LICORES	8	5
123	PRODUCTO 1	MARCA A	Descríp prod 1	DISPONIBLE	1500	BEBIDAS	11	10
124	PRODUCTO 2	MARCA B	DESCRIP PROD 2	DISPONIBLE	3000	LICORES	19	15
125	PRODUCTO 4	MARCA D	DESCRIP PROD 4	DISPONIBLE	5000	ABARROTES	15	20
126	PRODUCTO 3	MARCA C	DESCRIP PROD 3	DISPONIBLE	500	ABARROTES	10	10

* Campos Obligatorios

Productos

* Código: 1

* Nombre: PRODUCTO 5

* Descripción: DESCRIP PROD 5

* Marca: MARCA E

* Tipo: LICORES

Stock

* Actual: 8

* Crítico: 5

* Estado: DISPONIBLE

Precios

* Venta: 123

Agregar Actualizar Eliminar Salir

Figura 21.

Nombre: Mantenedor de Productos, referencia Figura 21.

Objetivos: Permite agregar, actualizar, eliminar los productos de la botillería.

Reglas del Proceso: No requiere permisos de administrador para utilizar esta funcionalidad. Como se ve en la figura 21, para poder registrar un producto, necesitamos el nombre, descripción, marca, tipo (Licores, Bebidas, Abarrotes, Cigarrillos), stock actual y crítico, estado (Disponible, No Disponible) y precio venta. Al presionar Aceptar se inserta de inmediato en la base de datos y queda disponible para todas las funcionalidades del sistema.

En caso que el código de producto esté repetido, se alertará al usuario mediante una ventana emergente, indicando "Ingreso fallido, producto ya existe". referencia Figura 22.

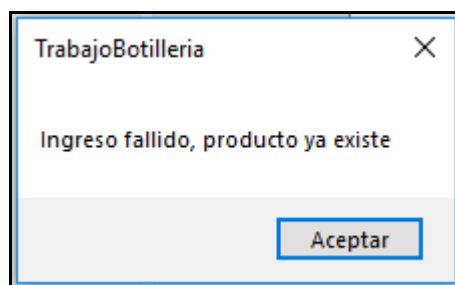


Figura 22.

Si desea modificar un producto, se habilitan todos los campos de la funcionalidad exceptuando el código del producto. Si el stock crítico es mayor al stock actual

que desea modificar, surgirá una ventana en donde le mostrará una alerta, pero solo es para informar tal inconveniente. Referencia Figura 23

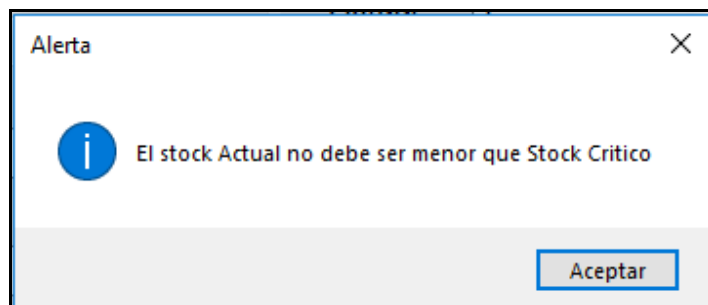


Figura 23.

Restricciones de eliminación.

Si desea eliminar un producto debe considerar que no tenga ventas, compras o deudas asociadas. Referencia Figura 24.

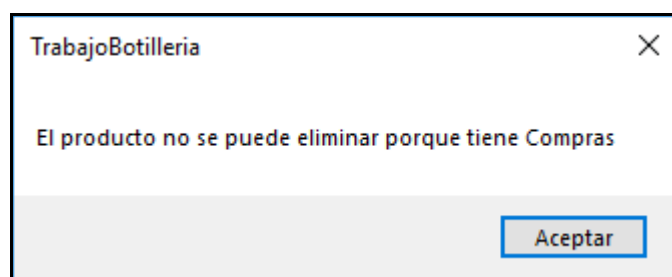


Figura 24.

Si tiene algún tipo de ajuste el producto tampoco dejará eliminar, ya que es una medida de protección para evitar fraudes. Referencia Figura 25

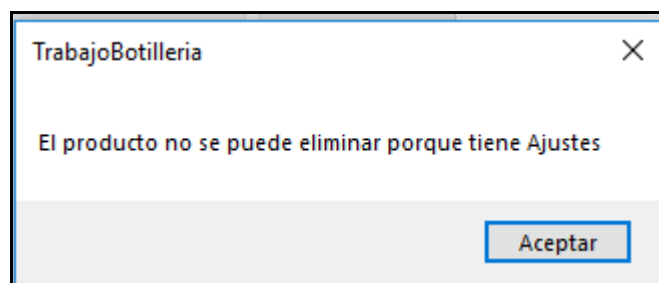


Figura 25

Referencia código fuente:Anexo página 66.

3.4.4 Ventas.

Diagrama de Bloques:

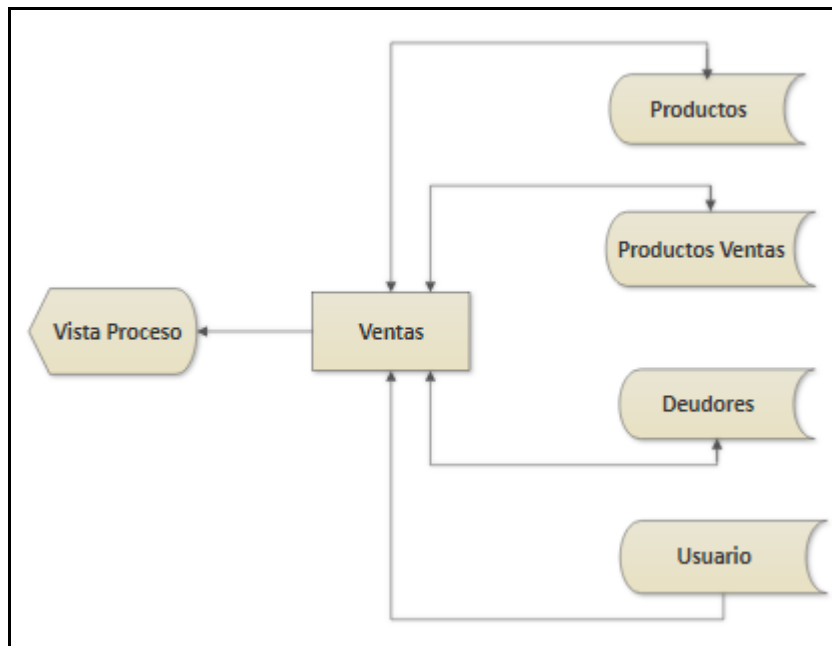


Figura 26.

Nombre: Ventas.

Objetivos: Permite hacer ventas de productos de una manera mas fácil y rapido.

Reglas del Proceso: Si necesita hacer una venta a un cliente solo debe seleccionar el producto que se muestra en la tabla superior, no se necesitan permisos de administrador para poder ocupar esta funcionalidad. Referencia Figura 27.

Fecha de Venta domingo, 24 de marzo de 2019							
	COD_PRODUCTO	NOM_PRODUCTO	MARCA	DESCRIPCION	ESTADO	PRECIO_VENTA	STOCK_ACTUAL
▶	1	PRODUCTO 5	MARCA E	DESCRIP PROD 5	DISPONIBLE	123	8
	123	PRODUCTO 1	Marca A	Descrip prod 1	DISPONIBLE	1500	11
	124	PRODUCTO 2	MARCA B	DESCRIP PROD 2	DISPONIBLE	3000	19
	125	PRODUCTO 4	MARCA D	DESCRIP PROD 4	DISPONIBLE	5000	15
	126	PRODUCTO 3	MARCA C	DESCRIP PROD 3	DISPONIBLE	500	10

Figura 27.

Luego de seleccionar el producto deberá indicar la cantidad que comprarán y en caso que haya algún descuento indicarlo. Referencia Figura 28.

Mostrar Ventas

Codigo Producto :

Nombre Producto:

Precio :

Cantidad :

Descuentos :

Agregar

Eliminar

Cancelar

Figura 28.

Estas ventas se van acumulando en una DataGridView de compras. En donde automaticamente se va mostrando el total de la compra.

Para eliminar una venta antes de ser pagada, solo selecciona la fila de la tabla y presionar eliminar. Si ya hizo una venta y el cliente se retracta por cualquier razon, puede dirigirse a mostrar venta y eliminarla totalmente (esta funcionalidad se necesitan permisos de administrador). Se pueden ver las ventas del día, mes o año en el boton "Mostrar Ventas". Referencia Figura 29.

	CODIGO	NOMBRE	PRECIO	CANTIDAD	DCTO	SUBTOTAL
▶	1	PRODUCTO 5	123	1	0	123
	124	PRODUCTO 2	3000	6	0	18000
	126	PRODUCTO 3	500	9	500	4000

Total Venta :

22123

Pago Efectivo :

0

Pagar

Figura 29.

Si el cliente quiere "fiar" lo que desea comprar, se busca su nombre en la esquina derecha y se agrea la deuda. En caso que el cliente no se encuentre en la base de datos del sistema se podrá agregar sin tener que salir de la venta (esta funcionalidad se necesitan permisos de administrador). Referencia Figura 30.

Agregar A Deudores

Nueva Venta

Salir

Fiar

Rut :

DEUDOR

DEUDOR

TEST3

TEST2

NOMBRE 1

NOMBRE 1

Figura 30.

Referencia código fuente:Anexo página 77.

3.4.5. Compras a Proveedor

Diagrama de Bloques:

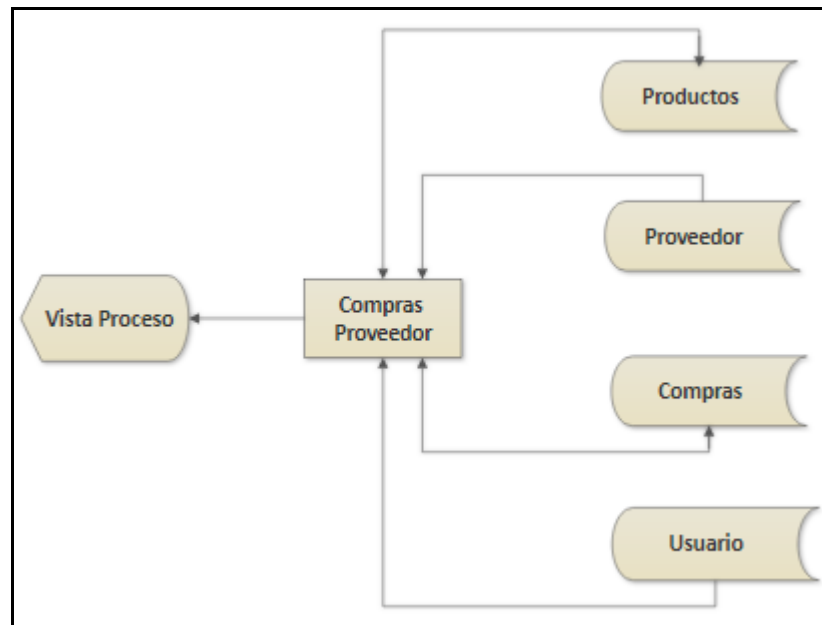


Figura 31.

Nombre: Compras a Proveedor

Objetivos: Permite realizar la compra de productos que abastecen a la botillería.

Reglas del Proceso: No se necesita permisos de administrador para esta funcionalidad. Para realizar una compra de mercadería, es necesario saber si el proveedor está en la lista del sistema, para eso se busca en la tabla superior y se selecciona para así dejar el registro de quien esta vendiendo los productos.

Al finalizar la compra de productos, se pueden validar todo lo obtenido en "Ver Compras", en donde lista todos los productos vendidos por cada proveedor. Se puede filtrar por día, mes y año. Referencia Figura 32.

Captura de pantalla de la interfaz de usuario para Compras a Proveedor. La interfaz tiene un botón 'Ver Compras' en la parte superior izquierda. A la izquierda hay un formulario con campos para 'Proveedor', 'Rut:', 'Nombre:' y 'Empresa:'. El campo 'Rut:' contiene el valor '18878616-2'. El campo 'Nombre:' contiene el valor 'PROVEEDOR 5'. El campo 'Empresa:' contiene el valor 'EMPRESA B'. A la derecha del formulario hay una tabla titulada 'Proveedores' con las siguientes columnas: RUT_PROVEEDOR, NOM_PROVEEDOR, EMPRESA y CELULAR. La tabla contiene cuatro filas de datos, la tercera de las cuales está seleccionada con un fondo azul.

RUT_PROVEEDOR	NOM_PROVEEDOR	EMPRESA	CELULAR
11111111-1	PROVEEDOR 1	EMPRESA A	+569876543
12345678-9	PROVEEDOR2	EMPRESA1	+5698765431
18878616-2	PROVEEDOR 5	EMPRESA B	+56923567842

Figura 32.

En la tabla del medio, se visibilizan todos los productos que se encuentran en la botillería, con su respectivo stock actual y stock crítico, así se hace mas fácil la compra de productos. En caso que el producto no se encuentre en la lista, existe una opción "Producto Nuevo", y se ingresa automaticamente al sistema. referencia Figura 33-34.

Productos									
	COD_PRODUCTO	NOM_PRODUCTO	MARCA	DESCRIPCION	ESTADO	PRECIO_VENTA	TIPO_PROD	STOCK_ACTUAL	
►	1	PRODUCTO 5	MARCA E	DESCRIP PROD 5	DISPONIBLE	123	LICORES	8	5
	123	PRODUCTO 1	Marca A	Descrip prod 1	DISPONIBLE	1500	BEBIDAS	11	10
	124	PRODUCTO 2	MARCA B	DESCRIP PROD 2	DISPONIBLE	3000	LICORES	19	15
	125	PRODUCTO 4	MARCA D	DESCRIP PROD 4	DISPONIBLE	5000	ABARROTES	15	20
	126	PRODUCTO 3	MARCA C	DESCRIP PROD 3	DISPONIBLE	500	ABARROTES	10	10

Figura 33.

Pedido

Código Producto:

☒ Producto Nuevo

Nombre Producto:

Marca:

Descripción:

Tipo:

BEBIDAS

▼

Precio Venta:

Stock crítico:

Cantidad:

Precio Compra Unitario:

Agregar

Cancelar

Figura 34.

Al agregar algún producto e ingresar un precio de compra menor o igual al precio que se vende el producto, emerge un mensaje de alerta, que es el siguiente. Referencia Figura 35.

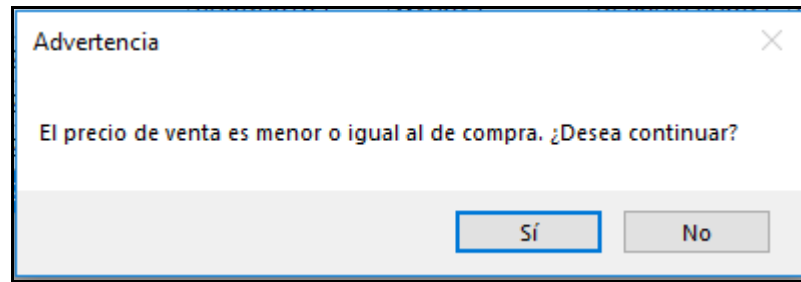


Figura 35.

Para finalizar la compra todos los productos seleccionados se agregan a una DataGridView con código, nombre, cantidad, precio compra y subtotal. Luego se aprieta el botón Comprar. Referencia Figura 26.

En caso que no se quiera comprar algún producto, existe la opción de “Eliminar producto de compra”.

	CODIGO	NOMBRE	CANTIDAD	PRECIO COMPRA	SUBTOTAL
▶	126	PRODUCTO 3	12	600	7200
	124	PRODUCTO 2	400	100	40000
	1	PRODUCTO 5	123	123	15129

Eliminar producto de compra

PEDIDO TOTAL:

62329

Comprar

Salir

Figura 36.

Referencia código fuente:Anexo página 62.

CONCLUSIONES Y RECOMENDACIONES

El desarrollo de un proyecto se ve fácil al saber manejar algún lenguaje de programación, pero a medida que uno va avanzando se encuentra con inconvenientes, ya sean malos modelos de base de datos, las tablas no se encuentran con toda la información o simplemente no es lo que quería el usuario.

El desarrollo del proyecto demoró 1 año, donde el primer paso fue el análisis y diseño del sistema. En esa etapa, nos dimos cuenta que teníamos varios errores. Con el paso del tiempo y las revisiones mensuales de nuestros profesores modificamos todo lo que estaba incorrecto y se obtuvo un modelo de datos consistente.

En la etapa del segundo semestre, ya comenzamos con el desarrollo de nuestro sistema. Aprendimos a manejar el lenguaje de programación Visual Basic.net y también el sistema de gestión de base de datos MySQL Server.

Para finalizar, se obtuvo un buen resultado al realizar este sistema, adquirimos bastante conocimiento respecto a lenguajes y uso de base de datos. Interactuamos con nuestro usuario y entendimos bien lo que requería para su botillería.

Durante todo este Trabajo de Título, tuve varios inconvenientes, pero el más crítico fue que mi compañero de trabajo tiro todo por la borda. Es difícil pasar por un momento así, uno piensa que todo termina ya que faltaba mucho por hacer y poco tiempo. Gracias al apoyo constante de mi profesor guía, seguí adelante y pude lograr terminar el sistema. Es una experiencia bastante estresante, pero te das cuenta que puedes hacerlo, pase lo que pase. Si uno se propone lo logra, a pesar de las trancas hay que seguir a delante.

BIBLIOGRAFÍA

- Conexión Base de Datos: <https://social.msdn.microsoft.com/Forums/es-ES/9114dfef-a83f-4e6e-be2c-192f23cac87e/como-unir-visual-basicnet-con-cnet?forum=dotnetes>.
- Diseño diagrama de bloques: <https://cloud.smartdraw.com/>
- Ejemplo de Trabajo de Título anteriores: Rediseño página web bienestar estudiantil: Implementación de módulo de postulación y seguimiento becas Federación.

ANEXOS A: CÓDIGO FUENTE

Clases:

```
Module Biblioteca
Public Structure Productos
    Public cod_producto As String
    Public nom_producto As String
    Public marca As String
    Public descripcion As String
    Public estado As String
    Public precio_venta As Integer
    Public tipo_prod As String
    Public stock_actual As Integer
    Public stock_critico As Integer
End Structure
Public Structure Deudores
    Public rut_deudor As String
    Public nom_deudor As String
    Public direccion As String
    Public celular As String
End Structure
Public Structure Proveedor
    Public rut_proveedor As String
    Public nom_proveedor As String
    Public empresa As String
    Public celular As String
End Structure

Public Structure Ajustes
    Public cod_producto As String
    Public tipo_ajuste As String
    Public descripcion As String
    Public fecha As Date
    Public cantidad As Integer
End Structure
End Module
```

BsnNegocios:

```
Imports TrabajoBotilleria.DaoUsuario
Imports TrabajoBotilleria.Usuario
Imports TrabajoBotilleria.Biblioteca
Public Class BsnNegocio
    Dim daoUsuario As New DaoUsuario
    Dim daoDatos As New DaoDatos
    Public Function ValidarUsuario(usuario As Usuario) As Usuario
        Return daoUsuario.ValidarUsuario(usuario)
    End Function
    Public Function GetProductos()
        Return daoDatos.GetProductos()
    End Function
    Public Function GetProductosBajoStock()
        Return daoDatos.GetProductosBajoStock()
    End Function
    Public Function GetDeudores()
        'iterar a través de los rut de deudores para colocarles el guion antes de mostrarlos
        Dim datos As DataTable = daoDatos.GetDeudores()
        For Each fila As DataRow In datos.Rows
            fila.Item(0) = FormatearRutParaMostrar(fila.Item(0))
        Next
        Return datos
    End Function
    Public Function GetUsuarios()
        Dim datos As DataTable = daoDatos.GetUsuarios()
        For Each fila As DataRow In datos.Rows
            fila.Item(0) = FormatearRutParaMostrar(fila.Item(0))
        Next
        Return datos
    End Function

    Public Function GetProveedores()
```

```

'iterar a través de los rut de deudores para colocarles el guion antes de mostrarlos
Dim datos As DataTable = daoDatos.GetProveedores()
For Each fila As DataRow In datos.Rows
    fila.Item(0) = FormatearRutParaMostrar(fila.Item(0))
Next
Return datos
End Function
Public Function GetAjustes()
    Return daoDatos.GetAjustes()
End Function
Public Function GetAjustesSegunProducto(cod_producto As String)
    Return daoDatos.GetAjustesSegunProducto(cod_producto)
End Function
Public Function InsertarProductos(cod_producto As String, nom_producto As String, marca As String, descripcion As String,
estado As String, precio_venta As Integer, tipo_prod As String, stock_Actual As Integer, stock_critico As Integer)
    Dim obj As New DaoDatos()
    Dim prod As New Productos()
    prod.cod_producto = cod_producto.Trim()
    prod.nom_producto = nom_producto.Trim()
    prod.marca = marca.Trim()
    prod.descripcion = descripcion.Trim()
    prod.estado = estado.Trim()
    prod.precio_venta = precio_venta
    prod.tipo_prod = tipo_prod.Trim()
    prod.stock_actual = stock_Actual
    prod.stock_critico = stock_critico
    Return obj.InsertarProductos(prod)
End Function
Public Function InsertarProductoDesdeCompra(cod_producto As String, nom_producto As String, marca As String,
descripcion As String, precio_venta As Integer, tipo_prod As String, stock_actual As Integer, stock_critico As Integer)
    'revisar si cod_producto existe
    If daoDatos.CodigoRepetido(cod_producto) Then
        MsgBox("El código de producto ya existe en el sistema")
        Return Nothing
    End If
    'agregar producto
    Return InsertarProductos(cod_producto, nom_producto, marca, descripcion, "DISPONIBLE", precio_venta, tipo_prod, 0,
stock_critico)
End Function

Public Sub ActualizarProductos(cod_producto As String, nom_producto As String, marca As String, descripcion As String,
estado As String, precio_venta As Integer, tipo_prod As String, stock_Actual As Integer, stock_critico As Integer)
    Dim prod As New Productos()
    prod.cod_producto = cod_producto.Trim()
    prod.nom_producto = nom_producto.Trim()
    prod.marca = marca.Trim()
    prod.descripcion = descripcion.Trim()
    prod.estado = estado.Trim()
    prod.precio_venta = precio_venta
    prod.tipo_prod = tipo_prod.Trim()
    prod.stock_actual = stock_Actual
    prod.stock_critico = stock_critico
    daoDatos.ActualizarProductos(prod)
End Sub
Function EliminarProductos(cod_producto As String)
    Return daoDatos.EliminarProductos(cod_producto)
End Function
***** COMPRAS *****

'Inserta la compra en la BD
'Recibe rut del proveedor y datagridview con los productos de la compra
Sub InsertarCompraProductos(rut_proveedor As String, ByRef dgvCompra As DataGridView)
    'Insertar en compras, recibe cod_compra
    Dim cod_compra As Integer = daoDatos.InsertarCompra(FormatearRutParaInsertar(rut_proveedor))

    'Insertar productos de la compra
    daoDatos.InsertarProductosCompra(cod_compra, dgvCompra)
End Sub

'Obtener compras

```

```

'periodo ->(dia, mes o año)
Public Function GetCompras(fecha As Date, periodo As String)
    If periodo = "dia" Then
        Return daoDatos.GetComprasDia(fecha)
    ElseIf periodo = "mes" Then
        Return daoDatos.GetComprasMes(fecha)
    Else
        Return daoDatos.GetComprasAño(fecha)
    End If
End Function
Public Function GetDetalleCompras(cod_compra As Integer)
    Return daoDatos.GetDetalleCompra(cod_compra)
End Function
Public Sub EliminarCompra(cod_compra As Integer)
    daoDatos.BorrarCompra(cod_compra)
End Sub
Public Function GetAñosCompras()
    Return daoDatos.GetAñosCompras()
End Function
Public Function GetComprasProveedor(rut_proveedor As String, año As Integer)
    If rut_proveedor = "0" Then
        'todos los proveedores
        Return daoDatos.GetComprasProveedores(año)
    Else
        'proveedor en específico
        Return daoDatos.GetComprasProveedores(año, FormatearRutParaInsertar(rut_proveedor))
    End If
End Function
Function Filtrar(tabla As String, filtro As String, texto As String)
    Return daoDatos.Filtrar(tabla, filtro, texto)
End Function
Friend Sub EliminarProdutos(producto As Object)
    Throw New NotImplementedException()
End Sub
Public Sub InsertarProveedor(rut_proveedor As String, nom_proveedor As String, empresa As String, celular As String)
    Dim obj As New DaoDatos()
    Dim prov As New Proveedor()
    If daoDatos.ExisteProveedor(FormatearRutParaInsertar(rut_proveedor.Trim())) Then
        MsgBox("Ya existe el proveedor en el sistema")
        Return
    End If
    prov.rut_proveedor = FormatearRutParaInsertar(rut_proveedor.Trim())
    prov.nom_proveedor = nom_proveedor.Trim()
    prov.empresa = empresa.Trim()
    prov.celular = celular.Trim()

    obj.InsertarProveedor(prov)
End Sub
Function EliminarProveedor(rut_proveedor As String)
    Return daoDatos.EliminarProveedor(FormatearRutParaInsertar(rut_proveedor))
End Function
Public Sub ActualizarProveedor(rut_proveedor As String, nom_proveedor As String, empresa As String, celular As String)
    Dim prov As New Proveedor()
    prov.rut_proveedor = FormatearRutParaInsertar(rut_proveedor.Trim())
    prov.nom_proveedor = nom_proveedor.Trim()
    prov.empresa = empresa.Trim()
    prov.celular = celular.Trim()
    daoDatos.ActualizarProveedor(prov)
End Sub
Public Sub InsertarUsuario(user As Usuario)
    Dim obj As New DaoDatos()
    user.RutUsuario = FormatearRutParaInsertar(user.RutUsuario)
    obj.InsertarUsuario(user)
End Sub
Public Sub ActualizarUsuario(RutUsuario As String, password As String, NombreUsuario As String, tipoUsuario As String, CelularUsuario As String)
    Dim user As New Usuario()
    user.RutUsuario = FormatearRutParaInsertar(RutUsuario.Trim())
    user.Password = password.Trim()
    user.NombreUsuario = NombreUsuario.Trim()

```

```

        user.TipoUsuario = tipoUsuario
        user.CelularUsuario = CelularUsuario.Trim()
        daoDatos.ActualizarUsuario(user)
    End Sub
Function EliminarUsuario(RutUsuario As String)
    Return daoDatos.EliminarUsuario(FormatearRutParaInsertar(RutUsuario))
End Function
'***** Deudor *****+++
Public Function GetDeudor(rut_deudor As String)
    Return daoDatos.GetDeudor(rut_deudor)
End Function
Public Function GetDeudas(rut_deudor As String)
    Return daoDatos.GetDeudas(rut_deudor)
End Function
Public Function GetAbonosDeudas(cod_venta As Integer)
    Return daoDatos.GetAbonosDeuda(cod_venta)
End Function
Public Function InsertarAbono(cod_venta, monto)
    daoDatos.InsertarAbono(cod_venta, monto)
End Function
Public Sub InsertarDeudor(rut_deudor As String, nom_deudor As String, direccion As String, celular As String)
    Dim obj As New DaoDatos()
    Dim deu As New Deudores()
    If daoDatos.ExisteDeudor(FormatearRutParaInsertar(rut_deudor.Trim())) Then
        MsgBox("El deudor ya existe en el sistema")
        Return
    End If
    deu.rut_deudor = FormatearRutParaInsertar(rut_deudor.Trim())
    deu.nom_deudor = nom_deudor.Trim()
    deu.direccion = direccion.Trim()
    deu.celular = celular.Trim()
    obj.InsertarDeudor(deu)
End Sub
Public Sub ActualizarDeudor(rut_deudor As String, nom_deudor As String, direccion As String, celular As String)
    Dim deud As New Deudores()
    deud.rut_deudor = FormatearRutParaInsertar(rut_deudor.Trim())
    deud.nom_deudor = nom_deudor.Trim()
    deud.direccion = direccion.Trim()
    deud.celular = celular.Trim()
    daoDatos.ActualizarDeudor(deud)
End Sub
Function EliminarDeudor(rut_deudor As String)
    Return daoDatos.EliminarDeudor(FormatearRutParaInsertar(rut_deudor))
End Function
'***** AJUSTES *****
Public Sub AgregarAjustes(cod_producto As String, tipo_ajuste As String, descripcion As String, fecha As Date, cantidad As Integer)
    Dim ajus As New Ajustes()
    ajus.cod_producto = cod_producto.Trim()
    ajus.tipo_ajuste = tipo_ajuste.Trim()
    ajus.descripcion = descripcion.Trim()
    ajus.fecha = fecha
    ajus.cantidad = cantidad
    daoDatos.AgregarAjustes(ajus)
End Sub

Public Sub ActualizarProductos(cod_producto As String, stock_actual As Integer)
    daoDatos.ActualizarProductos(cod_producto, stock_actual)
End Sub
'***** VENTAS *****
'Método que agrega una venta a la base de datos

Public Function IngresarVenta(ByVal productos As List(Of ProductoVenta))
    'obtener fecha de hoy
    Dim fecha As Date = DateTime.Now.Date

    'insertar nueva venta
    Dim cod_venta As Integer = daoDatos.AgregarVenta(fecha, LoginInfo.rutUsuario, productos)

    'rebajar stock

```

```

For Each prod As ProductoVenta In productos
    daoDatos.ActualizarProductos(prod.cod_producto, prod.cantidad, True)
Next
'returna cod_venta para insertarlo en tabla deuda cuando se vende fiado
Return cod_venta
End Function
Public Function GetAñosVentas()
    Return daoDatos.GetAñosVentas()
End Function
Public Sub IngresarVentaConDeuda(ByVal productos As List(Of ProductoVenta), rut_deudor As String)
    Dim cod_venta = IngresarVenta(productos)
    daoDatos.InsertarDeuda(cod_venta, FormatearRutParaInsertar(rut_deudor))
End Sub
Public Sub EliminarVenta(cod_venta As Integer)
    daoDatos.BorrarVenta(cod_venta)
End Sub
'Obtener ventas
'periodo ->(día, mes o año)
Public Function GetVentas(fecha As Date, periodo As String)
    If periodo = "día" Then
        Return daoDatos.GetVentasDia(fecha)
    ElseIf periodo = "mes" Then
        Return daoDatos.GetVentasMes(fecha)
    Else
        Return daoDatos.GetVentasAño(fecha)
    End If
End Function
'Obtener ventas PARA INFORME
'periodo ->(día, mes o año)
Public Function GetVentasInforme(fecha As Date, periodo As String)
    Dim ventas As DataTable
    If periodo = "día" Then
        ventas = daoDatos.GetVentasDiaInforme(fecha)
    ElseIf periodo = "mes" Then
        ventas = daoDatos.GetVentasMesInforme(fecha)
    Else
        ventas = daoDatos.GetVentasAñoInforme(fecha)
    End If
    Return ventas
End Function
'Obtener ventas CON DEUDA PARA INFORME
'periodo ->(día, mes o año)
Public Function GetVentasDeudaInforme(fecha As Date, periodo As String)
    Dim datos As DataTable
    If periodo = "día" Then
        datos = daoDatos.GetVentasDiaDeudaInforme(fecha)
    ElseIf periodo = "mes" Then
        datos = daoDatos.GetVentasMesDeudaInforme(fecha)
    Else
        datos = daoDatos.GetVentasAñoDeudaInforme(fecha)
    End If
    For Each fila As DataRow In datos.Rows
        fila.Item(2) = FormatearRutParaMostrar(fila.Item(2))
        If fila.Item(5).ToString() = "" Then
            fila.Item(5) = 0
        End If
        fila.Item(6) = Integer.Parse(fila.Item(4)) - Integer.Parse(fila.Item(5))
        If (Integer.Parse(fila.Item(4)) - Integer.Parse(fila.Item(5))) = 0 Then
            fila.Item(7) = "SI"
        Else
            fila.Item(7) = "NO"
        End If
    Next
    Return datos
End Function
Public Function GetDetalleVentas(cod_venta As Integer)
    Return daoDatos.GetDetalleVenta(cod_venta)
End Function

```

***** OTRAS FUNCIONES *****

```

'VALIDAR RUT
Public Function ValidarRut(ByVal rut As String) As String
    Dim Resultado As String = ""
    Dim Multiplicador As Integer = 2
    Dim iNum As Integer = 0
    Dim Suma As Integer = 0
    If Len(rut) < 8 Or Len(rut) > 9 Then
        Return -1
    End If
    For i As Integer = 8 To 1 Step -1
        iNum = Mid(rut, i, 1)
        Suma += iNum * Multiplicador
        Multiplicador += 1
        If Multiplicador = 8 Then Multiplicador = 2
    Next
    Resultado = CStr(11 - (Suma Mod 11))
    If Resultado = "10" Then Resultado = "K"
    If Resultado = "11" Then Resultado = "0"
    Return Resultado
End Function
'recibe rut CON GUION y se lo saca para insertarlo en la BD
Public Function FormatearRutParaInsertar(rut As String)
    If rut(0) = "0" Then
        rut = rut.Substring(1)
    End If
    Dim largo As Integer = rut.Length
    Return rut.Substring(0, largo - 2) & rut.Substring(largo - 1, 1)
End Function
'recibe rut SIN GUION (como está en la BD) y se lo coloca
Public Function FormatearRutParaMostrar(rut As String)
    Dim largo As Integer = rut.Length
    Return rut.Substring(0, largo - 1) & "-" & rut.Substring(largo - 1, 1)
End Function
End Class

```

DaoDatos

```

Imports MySql.Data.MySqlClient

Imports TrabajoBotilleria.Biblioteca

Public Class DaoDatos

    Dim conexionMysql As New Conexion

    Function Consulta(cadena As String) As DataTable

        Dim command As New MySqlCommand

        Dim dataset As New DataSet

        command.Connection = conexionMysql.GetConexion()

        conexionMysql.AbrirConexion()

        command.CommandText = cadena

        Dim reader As New MySqlDataAdapter

        reader.SelectCommand = command

        reader.Fill(dataset)

        conexionMysql.CerrarConexion()

        Return dataset.Tables(0)

    End Function

    Function Actualizacion(aux As Byte, sentencia As String) As Integer

        Dim command As New MySqlCommand

        Dim mensaje As Boolean

        Dim retorno As Integer
    
```

```

command.Connection = conexionMysql.GetConexion()
conexionMysql.AbrirConexion()
command.CommandText = sentencia
Try
    retorno = command.ExecuteNonQuery()
    mensaje = True
Catch ex As Exception
    Select Case aux
        Case 1
            MsgBox("Error! No se ha podido agregar el registro")
        Case 2
            MsgBox("Error! No se ha podido actualizar el registro")
        Case 3
            MsgBox("Error! No se ha podido eliminar el registro")
        Case Else
            MsgBox(ex.ToString())
    End Select
    mensaje = False
End Try
conexionMysql.CerrarConexion()
Return retorno
End Function

'***** PRODUCTOS *****

Friend Function InsertarProductos(producto As Productos)

    If Not CodigoRepetido(producto.cod_producto) Then

        Return Actualizacion(1, "Insert into Productos (cod_producto, nom_producto, marca , descripcion, estado,
precio_venta, tipo_prod, stock_actual, stock_critico) value ('" & producto.cod_producto & "' , '" & producto.nom_producto &
"' ,'" & producto.marca & "' ,'" & producto.descripcion & "' ,'" & producto.estado & "' ,'" & producto.precio_venta & "' ,'" &
producto.tipo_prod & "' ,'" & producto.stock_actual & "' ,'" & producto.stock_critico & "' )"

    End If

    Return Nothing
End Function

Friend Function ActualizarProductos(producto As Productos)

    Return Actualizacion(2, "Update Productos set nom_producto = '" & producto.nom_producto & "' , marca = '" &
producto.marca & "' , descripcion = '" & producto.descripcion & "' , estado = '" & producto.estado & "' , precio_venta = '" &
producto.precio_venta & "' , tipo_prod = '" & producto.tipo_prod & "' , stock_actual = '" & producto.stock_actual & "' ,
stock_critico = '" & producto.stock_critico & "' where cod_producto = '" & producto.cod_producto & "' )"

End Function

Friend Function ActualizarProductos(cod_producto As String, stock_actual As Integer)

    Return Actualizacion(2, "Update Productos set stock_actual = '" & stock_actual & "' where cod_producto = '" &
cod_producto & "' )"

End Function

Friend Function ActualizarProductos(cod_producto As String, cantidad As Integer, esRebaja As Boolean)

    If esRebaja Then

        cantidad *= -1

    End If

```

```

Return Actualizacion(2, "Update Productos set stock_actual = stock_actual + " & cantidad & " where cod_producto = " &
cod_producto & """)
End Function

Friend Function EliminarProductos(cod_producto As String)

If ProductoConAjustes(cod_producto) Then

Return -1

End If

If ProductoConCompras(cod_producto) Then

Return -2

End If

If ProductoConVentas(cod_producto) Then

Return -3

End If

Return Actualizacion(3, "Delete from Productos where cod_producto = " & cod_producto & """)

End Function

'Obtiene productos con stock critico

Public Function GetProductosBajoStock()

Return Consulta("SELECT cod_producto,nom_producto,stock_actual,stock_critico FROM productos WHERE
stock_actual<=stock_critico")

End Function

'Revisa si producto tiene algún movimiento (ajuste de stock, venta o compra).

'Si tiene alguno de esos 3, no se puede borrar.

Friend Function ProductoConMovimiento(cod_producto As String)

Dim result = Consulta("SELECT p.COD_PRODUCTO FROM productos p JOIN ajustes a ON
p.COD_PRODUCTO=a.COD_PRODUCTO WHERE p.COD_PRODUCTO=" & cod_producto & " UNION SELECT p.COD_PRODUCTO
FROM productos p JOIN productos_ventas pv ON p.COD_PRODUCTO=pv.COD_PRODUCTO WHERE p.COD_PRODUCTO=" &
cod_producto & " UNION SELECT p.COD_PRODUCTO FROM productos p JOIN productos_compras pc ON
p.COD_PRODUCTO=pc.COD_PRODUCTO WHERE p.COD_PRODUCTO=" & cod_producto)

Return result.Rows.Count > 0

End Function

Friend Function ProductoConVentas(cod_producto As String)

Dim result = Consulta("SELECT p.COD_PRODUCTO FROM productos p JOIN productos_ventas pv ON
p.COD_PRODUCTO=pv.COD_PRODUCTO WHERE p.COD_PRODUCTO=" & cod_producto)

Return result.Rows.Count > 0

End Function

Friend Function ProductoConCompras(cod_producto As String)

Dim result = Consulta("SELECT p.COD_PRODUCTO FROM productos p JOIN productos_compras pc ON
p.COD_PRODUCTO=pc.COD_PRODUCTO WHERE p.COD_PRODUCTO=" & cod_producto)

Return result.Rows.Count > 0

End Function

Friend Function ProductoConAjustes(cod_producto As String)

Dim result = Consulta("SELECT p.COD_PRODUCTO FROM productos p JOIN ajustes a ON
p.COD_PRODUCTO=a.COD_PRODUCTO WHERE p.COD_PRODUCTO=" & cod_producto)

Return result.Rows.Count > 0

End Function

'Revisa si el codigo esta repetido.

Friend FunctionCodigoRepetido(cod_producto As String)

```

```

Dim result = Consulta("SELECT COD_PRODUCTO FROM productos WHERE COD_PRODUCTO = " & cod_producto & " ")

Return result.Rows.Count > 0

End Function

'***** COMPRAS *****

'Crear compra nueva. Inserta en tabla compras

'Returna cod_compra

Friend Function InsertarCompra(rut_proveedor As String)

    Actualizacion(1, "INSERT INTO compras (FECHA,RUT_PROVEEDOR) values ('" & DateTime.Now.ToString("yyyy-MM-dd") &
    "','" & rut_proveedor & "')")

    Dim result As DataTable = Consulta("SELECT cod_compra FROM compras ORDER BY cod_compra DESC LIMIT 0,1")

    Return Integer.Parse(result.Rows(0).Item(0))

End Function

'Inserta productos de la compra

Friend Sub InsertarProductosCompra(cod_compra As Integer, ByRef dgvCompra As DataGridView)

    For Each producto_compra As DataGridViewRow In dgvCompra.Rows

        Dim cod_producto As Integer = Integer.Parse(producto_compra.Cells(0).Value)

        Dim cantidad As Integer = Integer.Parse(producto_compra.Cells(2).Value)

        Dim precio_compra As Integer = Integer.Parse(producto_compra.Cells(3).Value)

        Actualizacion(1, "INSERT INTO productos_compras(COD_PRODUCTO,COD_COMPRA,CANTIDAD,PRECIO_COMPRA)
VALUES(" & cod_producto & "," & cod_compra & "," & cantidad & "," & precio_compra & ")")

        Actualizacion(2, "UPDATE productos SET STOCK_ACTUAL = STOCK_ACTUAL+" & cantidad & " WHERE COD_PRODUCTO="
& cod_producto)

    Next

End Sub

'Returna las compras realizadas en una fecha en particular

Public Function GetComprasDia(fecha As Date)

    Dim fecha_compra = fecha.ToString("yyyy-MM-dd")

    Return Consulta("SELECT c.COD_COMPRA,p.RUT_PROVEEDOR,p.NOM_PROVEEDOR,DATE_FORMAT(c.FECHA,'%d-%m-%Y')
as fecha_compra,SUM((pc.PRECIO_COMPRA*pc.CANTIDAD)) as total_compra FROM proveedores p JOIN compras c ON
p.RUT_PROVEEDOR=c.RUT_PROVEEDOR JOIN productos_compras pc ON c.cod_compra=pc.cod_compra WHERE c.FECHA = '"
& fecha_compra & "' GROUP BY pc.COD_COMPRA")

End Function

'Returna las compras realizadas en un mes (y año) en particular

Public Function GetComprasMes(fecha As Date)

    Return Consulta("SELECT c.COD_COMPRA,p.RUT_PROVEEDOR,p.NOM_PROVEEDOR,DATE_FORMAT(c.FECHA,'%d-%m-%Y')
as fecha_compra,SUM((pc.PRECIO_COMPRA*pc.CANTIDAD)) as total_compra FROM proveedores p JOIN compras c ON
p.RUT_PROVEEDOR=c.RUT_PROVEEDOR JOIN productos_compras pc ON c.cod_compra=pc.cod_compra WHERE
MONTH(c.FECHA) = '" & fecha.Month & "' AND YEAR(c.FECHA) = '" & fecha.Year & "' GROUP BY pc.COD_COMPRA")

End Function

'Returna las compras realizadas en un año en particular

Public Function GetComprasAño(fecha As Date)

    Return Consulta("SELECT c.COD_COMPRA,p.RUT_PROVEEDOR,p.NOM_PROVEEDOR,DATE_FORMAT(c.FECHA,'%d-%m-%Y')
as fecha_compra,SUM((pc.PRECIO_COMPRA*pc.CANTIDAD)) as total_compra FROM proveedores p JOIN compras c ON
p.RUT_PROVEEDOR=c.RUT_PROVEEDOR JOIN productos_compras pc ON c.cod_compra=pc.cod_compra WHERE
YEAR(c.FECHA) = '" & fecha.Year & "' GROUP BY pc.COD_COMPRA")

End Functio

'Returna detalle de compra

Public Function GetDetalleCompra(cod_compra As Integer)

```

```

Return Consultas("SELECT
p.COD_PRODUCTO,p.NOM_PRODUCTO,pc.PRECIO_COMPRA,pc.CANTIDAD,(pc.PRECIO_COMPRA*pc.CANTIDAD) as subtotal
FROM productos_compras pc JOIN productos p ON pc.COD_PRODUCTO=p.COD_PRODUCTO WHERE pc.COD_COMPRA = " &
cod_compra)

End Function

Public Sub BorrarCompra(cod_compra)

'Borrar registros en tabla productos_compras

Actualizacion(3, "DELETE FROM productos_compras WHERE cod_compra=" & cod_compra)

'Borrar registros en tabla compras

Actualizacion(3, "DELETE FROM compras WHERE cod_compra=" & cod_compra)

End Sub

'Returna lista de años en que hay compras

Public Function GetAñosCompras()

Return Consultas("SELECT DISTINCT(YEAR(fecha)) as anio FROM compras ORDER BY FECHA DESC")

End Function

'Obtener las compras de TODOS los proveedores en un año

Public Function GetComprasProveedores(año As Integer)

Return Consultas("SELECT c.COD_COMPRA,p.RUT_PROVEEDOR,p.NOM_PROVEEDOR,c.FECHA as
fecha_compra,SUM(pc.PRECIO_COMPRA*pc.CANTIDAD) as total_compra FROM proveedores p JOIN compras c ON
p.RUT_PROVEEDOR=c.RUT_PROVEEDOR JOIN productos_compras pc ON c.cod_compra=pc.cod_compra WHERE
YEAR(c.FECHA)=" & año & " GROUP BY pc.COD_COMPRA")

End Function

'Obtener las compras de UN PROVEEDOR en un año

Public Function GetComprasProveedores(año As Integer, rut_proveedor As String)

Return Consultas("SELECT c.COD_COMPRA,p.RUT_PROVEEDOR,p.NOM_PROVEEDOR,c.FECHA as
fecha_compra,SUM(pc.PRECIO_COMPRA*pc.CANTIDAD) as total_compra FROM proveedores p JOIN compras c ON
p.RUT_PROVEEDOR=c.RUT_PROVEEDOR JOIN productos_compras pc ON c.cod_compra=pc.cod_compra WHERE
YEAR(c.FECHA)=" & año & " AND p.RUT_PROVEEDOR=" & rut_proveedor & " GROUP BY pc.COD_COMPRA")

End Function

Function GetProductos()

Return Consultas("Select * from productos")

End Function

Function GetDeudores()

Return Consultas("Select * from Deudores")

End Function

Function GetUsuarios()

Return Consultas("Select * from Usuarios ")

End Function

Function GetProveedores()

Return Consultas("Select * from Proveedores")

End Function

Function Filtrar(tabla As String, filtro As String, texto As String)

Return Consultas("Select * from " & tabla & " Where " & filtro & " Like '%" & texto & "%'")

End Function

! ***** PROVEEDORES *****

Friend Function InsertarProveedor(Proveedor As Proveedor)

```

```

Return Actualizacion(1, "Insert into Proveedores(rut_proveedor, nom_proveedor, empresa, celular) values ('" &
Proveedor.rut_proveedor & "', '" & Proveedor.nom_proveedor & "', '" & Proveedor.empresa & "', '" & Proveedor.celular & "')")
End Function

Friend Function EliminarProveedor(rut_proveedor As String)

If Not ProveedorTieneVenta(rut_proveedor) Then

Return Actualizacion(3, "Delete from Proveedores where rut_proveedor = '" & rut_proveedor & "'")

End If

Return Nothing

End Function

Friend Function ActualizarProveedor(proveedor As Proveedor)

Return Actualizacion(2, "Update Proveedores set nom_proveedor = '" & proveedor.nom_proveedor & "', empresa = '" &
proveedor.empresa & "', celular = '" & proveedor.celular & "' where rut_proveedor = '" & proveedor.rut_proveedor & "'")

End Function

'Revisa si proveedor tiene compras. Si NO tiene, entonces se puede borrar.
'Returna TRUE si proveedor tiene compras (no se puede borrar) y FALSE si no tiene compras.

Friend Function ProveedorTieneVenta(rut_proveedor As String)

Dim result = Consulta("SELECT * FROM proveedores p JOIN compras c ON p.RUT_PROVEEDOR=c.RUT_PROVEEDOR
WHERE p.RUT_PROVEEDOR='" & rut_proveedor & "'")

Return result.Rows.Count > 0

End Function

Friend Function ExisteProveedor(rut_proveedor As String)

Dim result As DataTable = Consulta("SELECT * FROM proveedores WHERE RUT_PROVEEDOR='" & rut_proveedor & "'")

Return result.Rows.Count > 0

End Function

'***** USUARIOS *****

Friend Function InsertarUsuario(usuario As Usuario)

Return Actualizacion(1, "Insert into Usuarios(rut_usuario,Password,Nombre_Usuario,Tipo_Usu,celular) values ('" &
usuario.RutUsuario & "', '" & usuario.Password & "', '" & usuario.NombreUsuario & "', '" & usuario.TipoUsuario & "', '" &
usuario.CelularUsuario & "')")

End Function

Friend Function EliminarUsuario(RutUsuario As String)

If Not CheckUsuarioConVenta(RutUsuario) Then

Return Actualizacion(3, "Delete from Usuarios where rut_usuario = '" & RutUsuario & "'")

End If

Return Nothing

End Function

Friend Function ActualizarUsuario(usuario As Usuario)

Return Actualizacion(2, "Update Usuarios set nombre_usuario = '" & usuario.NombreUsuario & "', Password = '" &
usuario.Password & "', Tipo_Usu = '" & usuario.TipoUsuario & "', Celular = '" & usuario.CelularUsuario & "' where rut_usuario =
'" & usuario.RutUsuario & "'")

End Function

'Revisa si usuario tiene ventas. Si NO tiene, entonces se puede borrar.
'Returna TRUE si usuario tiene ventas (no se puede borrar) y FALSE si no tiene ventas.

Friend Function CheckUsuarioConVenta(rut_usuario As String)

Dim result = Consulta("SELECT * FROM usuarios u JOIN ventas v ON u.RUT_USUARIO=v.RUT_USUARIO WHERE

```

```

u.RUT_USUARIO="" & rut_usuario & "")

Return result.Rows.Count > 0

End Function

'***** DEUDORES Y DEUDAS *****

Friend Function GetDeudor(rut_deudor As String)

Return Consulta("SELECT * FROM deudores WHERE rut_deudor=" & rut_deudor & "")

End Function

'Obtiene los abonos hechos a una venta con deuda

Friend Function GetAbonosDeuda(cod_venta As Integer)

Return Consulta("SELECT DATE_FORMAT(a.FECHA_PAGO,'%d-%m-%Y') as fecha_pago,a.MONTO FROM ventas v JOIN
deudas d ON v.COD_VENTA=d.COD_VENTA JOIN abonos a ON d.COD_DEUDA=a.COD_DEUDA WHERE v.COD_VENTA=" &
cod_venta)

End Function

Friend Function InsertarAbono(cod_venta As Integer, monto As Integer)

Dim result As DataTable = Consulta("SELECT cod_deuda FROM deudas WHERE cod_venta=" & cod_venta)

Dim cod_deuda As Integer = Integer.Parse(result.Rows(0).Item(0).ToString())

Dim fecha As String = DateTime.Now.ToString("yyyy-MM-dd")

Return Actualizacion(1, "INSERT INTO abonos(fecha_pago,monto,cod_deuda) VALUES(" & fecha & "," & monto & "," &
cod_deuda & ")")

End Function

Friend Function InsertarDeudor(deudores As Deudores)

Return Actualizacion(1, "Insert into deudores (rut_deudor,nombre_deudor,direccion,celular) values (" &
deudores.rut_deudor & "," & deudores.nom_deudor & "," & deudores.direccion & "," & deudores.celular & ")")

End Function

Friend Function EliminarDeudor(rut_deudor As String)

If DeudorconDeuda(rut_deudor) Then

MsgBox("No se puede eliminar. Esta persona registra deudas")

Return Nothing

Else

Return Actualizacion(3, "Delete from Deudores where rut_deudor = " & rut_deudor & "")

MsgBox("Deudor eliminado")

End If

End Function

Friend Function ActualizarDeudor(Deudores As Deudores)

Return Actualizacion(2, "Update Deudores set nombre_deudor = " & Deudores.nom_deudor & ", direccion = " &
Deudores.direccion & ", celular = " & Deudores.celular & " where rut_deudor = " & Deudores.rut_deudor & ")

End Function

'Revisa si Deudor tiene deudas (registros en tabla deudas). Si NO tiene, entonces se puede borrar.

'Returna TRUE si deudor tiene ventas (no se puede borrar) y FALSE si no tiene ventas.

Friend Function DeudorconDeuda(rut_deudor As String)

Dim result = Consulta("SELECT * FROM deudores d JOIN deudas de ON d.RUT_DEUDOR =de.RUT_DEUDOR WHERE
d.RUT_DEUDOR=" & rut_deudor & "")

Return result.Rows.Count > 0

End Function

Friend Function InsertarDeuda(cod_venta As Integer, rut_deudor As String)

Dim total_venta As Integer = ObtenerTotalVenta(cod_venta)

```

```

    Actualizacion(1, "INSERT INTO deudas (total_deuda,rut_deudor,cod_venta) VALUES(" & total_venta & "," & rut_deudor &
    "," & cod_venta & ")")

End function

Friend Function GetDeudas(rut_deudor As String)

    Return          Consulta("SELECT                                v.COD_VENTA,u.NOMBRE_USUARIO                as
vendedor,v.fecha_venta,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta,(SELECT SUM(monto) FROM
abonos WHERE COD_DEUDA=d.COD_DEUDA) as abonos,(SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO)-(SELECT
SUM(monto) FROM abonos WHERE COD_DEUDA=d.COD_DEUDA)) as deuda FROM deudas d JOIN ventas v ON
d.COD_VENTA=v.COD_VENTA JOIN productos_ventas pv ON v.COD_VENTA=pv.COD_VENTA JOIN productos p on
pv.COD_PRODUCTO=p.COD_PRODUCTO JOIN usuarios u ON v.RUT_USUARIO=u.RUT_USUARIO WHERE d.RUT_DEUDOR=" &
rut_deudor & " GROUP BY pv.COD_VENTA")

End Function

Friend Function ObtenerTotalVenta(cod_venta As Integer)

    Dim result = Consulta("SELECT SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM productos p
JOIN productos_ventas pv ON p.COD_PRODUCTO=pv.COD_PRODUCTO WHERE pv.COD_VENTA=" & cod_venta)

    Dim total_venta = result.Rows(0).Item(0)

    Return total_venta

End Function

Friend Function ExisteDeudor(rut_deudor As String)

    Dim result As DataTable = Consulta("SELECT * FROM deudores WHERE RUT_DEUDOR=" & rut_deudor & "")

    Return result.Rows.Count > 0

End Function

'***** AJUSTES *****

Function GetAjustes()

    Return Consulta("Select * from Ajustes")

End Function

Friend Function AgregarAjustes(ajustes As Ajustes)

    Return Actualizacion(1, "Insert into Ajustes (cod_producto, tipo_ajuste,descripcion,fecha,cantidad) values (" &
ajustes.cod_producto & "," & ajustes.tipo_ajuste & "," & ajustes.descripcion & "," &
FormateaFechaParaInsert(ajustes.fecha.Date) & "," & ajustes.cantidad & ")")

End Function

'Obtiene los ajustes de un producto

Friend Function GetAjustesSegunProducto(cod_producto As String)

    Return Consulta("SELECT a.tipo_ajuste,a.cantidad,a.fecha,a.descripcion FROM productos p JOIN ajustes a ON
p.cod_producto = a.cod_producto WHERE p.cod_producto=" & cod_producto & "")

End Function

Friend Function FormateaFechaParaInsert(fecha As Date)

    Return fecha.Year & "-" & fecha.Month & "-" & fecha.Day

End Function

'***** VENTAS *****

'inserta en tabla ventas y luego en tabla productos_ventas

Friend Function AgregarVenta(fecha As Date, rut_usuario As String, ByVal productos As List(Of ProductoVenta))

    'insertar en venta

    Actualizacion(1, "Insert into ventas (fecha_venta,rut_usuario) values (" & FormateaFechaParaInsert(fecha) & "," &
rut_usuario & ")")

    'obtener codigo_venta

    Dim result = Consulta("SELECT cod_venta FROM ventas ORDER BY cod_venta DESC LIMIT 0,1")

    Dim cod_venta = Integer.Parse(result.Rows(0).Item(0))

```

```

'insertar en tabla productos_ventas

For Each prod As ProductoVenta In productos
    Actualizacion(1, "Insert into productos_ventas (cod_producto,cod_venta,cantidad,descuento) values ('" &
prod.cod_producto & "', '" & cod_venta & "', " & prod.cantidad & ", " & prod.descuento & "')"
Next

'returna codigo de venta para usarlo en ventas con deudor
Return cod_venta

End Function

Public Sub BorrarVenta(cod_venta)

    If VentaTieneDeuda(cod_venta) Then

        Dim respuesta As Integer = MessageBox.Show("Esta venta tiene deuda pendiente. ¿Desea continuar de todas formas?
(se eliminará la deuda)", "Confirmación", MessageBoxButtons.YesNo)

        If respuesta = DialogResult.No Then

            Return

        End If

    End If

    'Borrar registros en tabla producto_venta
    Actualizacion(3, "DELETE FROM productos_ventas WHERE cod_venta=" & cod_venta)

    'Borrar registros en tabla abonos
    Dim cod_deuda As Integer = GetCodigoDeuda(cod_venta)
    Actualizacion(3, "DELETE FROM abonos WHERE cod_deuda=" & cod_deuda)

    'Borrar registros en tabla deudas
    Actualizacion(3, "DELETE FROM deudas WHERE cod_deuda=" & cod_deuda)

    'Borrar registros en tabla ventas
    Actualizacion(3, "DELETE FROM ventas WHERE cod_venta=" & cod_venta)

End Sub

'Ver si venta tiene deuda. Si tiene deuda asociada retorna su codigo
Public Function GetCodigoDeuda(cod_venta As Integer)

    Dim result As DataTable = Consulta("SELECT cod_deuda FROM deudas WHERE cod_venta=" & cod_venta)

    Try

        Return Integer.Parse(result.Rows(0).Item(0).ToString())

    Catch

        Return 0

    End Try

End Function

'Ver si venta tiene deuda pendiente (abonos no suman deuda total)
Public Function VentaTieneDeuda(cod_venta)

    Dim cod_deuda As Integer = GetCodigoDeuda(cod_venta)

    Dim result As DataTable = Consulta("SELECT SUM((p.precio_venta*cantidad)-descuento) as total_venta FROM productos p
JOIN productos_ventas pv ON p.cod_producto=pv.cod_producto WHERE pv.cod_venta=" & cod_venta)

    Dim total_venta As Integer = Integer.Parse(result.Rows(0).Item(0).ToString())

    Dim total_abonos As Integer

    If cod_deuda = 0 Then

        total_abonos = total_venta

    End If

End Function

```

```

Else
    result = Consulta("SELECT SUM(monto) as total_abonos FROM abonos WHERE cod_deuda=" & cod_deuda)
Try
    total_abonos = Integer.Parse(result.Rows(0).Item(0).ToString())
Catch
    total_abonos = 0
End Try
End If
Return total_venta > total_abonos
End Function

'Returna lista de años en que hay ventas
Public Function GetAñosVentas()
    Return Consulta("SELECT DISTINCT(YEAR(venta_fecha)) as año FROM ventas ORDER BY FECHA_VENTA DESC")
End Function

'Returna las ventas realizadas en una fecha en particular
Public Function GetVentasDia(venta_fecha As Date)
    Dim venta_fecha = venta_fecha.ToString("yyyy-MM-dd")

    Return Consulta("SELECT v.COD_VENTA,u.NOMBRE_USUARIO,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y') as
venta_fecha,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM usuarios u JOIN ventas v ON
u.RUT_USUARIO=v.RUT_USUARIO JOIN productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON
pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE v.FECHA_VENTA = '" & venta_fecha & "' GROUP BY pv.COD_VENTA")
End Function

'Returna las ventas realizadas en un mes (y año) en particular
Public Function GetVentasMes(venta_fecha As Date)
    Return Consulta("SELECT v.COD_VENTA,u.NOMBRE_USUARIO,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y') as
venta_fecha,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM usuarios u JOIN ventas v ON
u.RUT_USUARIO=v.RUT_USUARIO JOIN productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON
pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE MONTH(v.FECHA_VENTA) = '" & venta_fecha.Month & "' AND YEAR(v.FECHA_VENTA)
= '" & venta_fecha.Year & "' GROUP BY pv.COD_VENTA")
End Function

'Returna las ventas realizadas en un año en particular
Public Function GetVentasAño(venta_fecha As Date)
    Return Consulta("SELECT v.COD_VENTA,u.NOMBRE_USUARIO,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y') as
venta_fecha,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM usuarios u JOIN ventas v ON
u.RUT_USUARIO=v.RUT_USUARIO JOIN productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON
pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE YEAR(v.FECHA_VENTA) = '" & venta_fecha.Year & "' GROUP BY pv.COD_VENTA")
End Function

'Returna detalle de venta
Public Function GetDetalleVenta(cod_venta As Integer)
    Return
    Consulta("SELECT
p.COD_PRODUCTO,p.NOM_PRODUCTO,p.PRECIO_VENTA,pv.CANTIDAD,pv.DESCUENTO,(p.PRECIO_VENTA*pv.CANTIDAD)-
pv.DESCUENTO as subtotal FROM productos_ventas pv JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
pv.COD_VENTA = '" & cod_venta)
End Function

'***** PARA INFORME VENTAS

'Returna las ventas realizadas en una fecha en particular
Public Function GetVentasDiaInforme(venta_fecha As Date)
    Dim venta_fecha = venta_fecha.ToString("yyyy-MM-dd")

```

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM ventas v JOIN productos_ventas pv
ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE v.FECHA_VENTA = '"
& fecha_venta & "' GROUP BY pv.COD_VENTA")

```

End Function

'Retorna las ventas realizadas en un mes (y año) en particular

Public Function GetVentasMesInforme(fecha As Date)

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM ventas v JOIN productos_ventas pv
ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
MONTH(v.FECHA_VENTA) = '" & fecha.Month & "' AND YEAR(v.FECHA_VENTA) = '" & fecha.Year & "' GROUP BY
pv.COD_VENTA")

```

End Function

'Retorna las ventas realizadas en un año en particular

Public Function GetVentasAñoInforme(fecha As Date)

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta FROM ventas v JOIN productos_ventas pv
ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
YEAR(v.FECHA_VENTA) = '" & fecha.Year & "' GROUP BY pv.COD_VENTA")

```

End Function

'Retorna las ventas CON DEUDA realizadas en una fecha en particular

Public Function GetVentasDiaDeudaInforme(fecha As Date)

Dim fecha_venta = fecha.ToString("yyyy-MM-dd")

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,de.RUT_DEUDOR,de.NOMBRE_DEUDOR,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta,
(SELECT SUM(monto) FROM abonos WHERE cod_deuda=d.cod_deuda) as abonos, 0 as saldo, 'Si' as deuda_pagada FROM
deudores de JOIN deudas d ON de.rut_deudor=d.rut_deudor JOIN ventas v ON d.cod_venta=v.cod_venta JOIN
productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
v.FECHA_VENTA = '" & fecha_venta & "' GROUP BY pv.COD_VENTA")

```

End Function

'Retorna las ventas CON DEUDA realizadas en un mes (y año) en particular

Public Function GetVentasMesDeudaInforme(fecha As Date)

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,de.RUT_DEUDOR,de.NOMBRE_DEUDOR,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta,
(SELECT SUM(monto) FROM abonos WHERE cod_deuda=d.cod_deuda) as abonos, 0 as saldo, 'Si' as deuda_pagada FROM
deudores de JOIN deudas d ON de.rut_deudor=d.rut_deudor JOIN ventas v ON d.cod_venta=v.cod_venta JOIN
productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
MONTH(v.FECHA_VENTA) = '" & fecha.Month & "' AND YEAR(v.FECHA_VENTA) = '" & fecha.Year & "' GROUP BY
pv.COD_VENTA")

```

End Function

'Retorna las ventas CON DEUDA realizadas en un año en particular

Public Function GetVentasAñoDeudaInforme(fecha As Date)

```

Return          Consulta("SELECT          v.COD_VENTA,DATE_FORMAT(v.FECHA_VENTA,'%d-%m-%Y')          as
fecha_venta,de.RUT_DEUDOR,de.NOMBRE_DEUDOR,SUM((p.PRECIO_VENTA*pv.CANTIDAD)-pv.DESCUENTO) as total_venta,
(SELECT SUM(monto) FROM abonos WHERE cod_deuda=d.cod_deuda) as abonos, 0 as saldo, 'Si' as deuda_pagada FROM
deudores de JOIN deudas d ON de.rut_deudor=d.rut_deudor JOIN ventas v ON d.cod_venta=v.cod_venta JOIN
productos_ventas pv ON v.COD_VENTA = pv.COD_VENTA JOIN productos p ON pv.COD_PRODUCTO=p.COD_PRODUCTO WHERE
YEAR(v.FECHA_VENTA) = '" & fecha.Year & "' GROUP BY pv.COD_VENTA")

```

End Function

End Class

Ajustes

```
Imports MySql.Data.MySqlClient
Public Class FrmAjustes
    Dim bsnNegocio As New BsnNegocio
    Dim primero As Byte = 0
    Sub CargarTextbox()
        If Not (dgvProductos.Rows.Count = 0) Then
            txtCodigo.Text = dgvProductos.SelectedRows.Item(0).Cells(0).Value
            txtNombre.Text = dgvProductos.SelectedRows.Item(0).Cells(1).Value
            txtActual.Text = dgvProductos.SelectedRows.Item(0).Cells(7).Value
        End If
    End Sub
    Sub CargarCb()
        CbTipo.DropDownStyle = ComboBoxStyle.DropDownList
        CbTipo.DisplayMember = "Text"
        CbTipo.ValueMember = "Value"
        Dim tb As New DataTable
        tb.Columns.Add("Text", GetType(String))
        tb.Columns.Add("Value", GetType(Integer))
        tb.Rows.Add("MERMA (-)", 0)
        tb.Rows.Add("CONSUMO INTERNO (-)", 1)
        tb.Rows.Add("INCREMENTO DE PRODUCTOS (+)", 2)
        tb.Rows.Add("OTROS (-)", 3)
        CbTipo.DataSource = tb
    End Sub
    Private Sub FrmAjustes_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.WindowState = 2
        dgvProductos.DataSource = bsnNegocio.GetProductos()
        dgvProductos.Columns(2).Visible = False
        dgvProductos.Columns(3).Visible = False
        dgvProductos.Columns(4).Visible = False
        dgvProductos.Columns(5).Visible = False
        dgvProductos.Columns(6).Visible = False
        dgvProductos.Columns(8).Visible = False
        CargarTextbox()
        CargarCb()
        txtCodigo.Enabled = False
        txtActual.Enabled = False
        txtNombre.Enabled = False
        BtnActualizar.Visible = True
    End Sub
    Private Sub FrmAjustes_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing
        If e.CloseReason = CloseReason.UserClosing Then
            e.Cancel = True
            Me.Dispose()
            FrmMantenedores.Show()
        End If
    End Sub
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles BtnSalir.Click
        Me.Hide()
        FrmMantenedores.Show()
    End Sub

    Private Sub dgvAjustes_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvAjustes.CellClick
        CargarTextbox()
    End Sub
    Private Sub rbMerma_CheckedChanged(sender As Object, e As EventArgs)
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles BtnActualizar.Click
        'BtnAceptar.Visible = False
        BtnActualizar.Visible = True
        Dim prod As New Productos()
        Dim valor As Integer
        If txtAjuste.Text = "" Or TxtDescripcion.Text = "" Then
            MsgBox("Debe rellenar todos los campos")
        Else
            If CbTipo.Text = "MERMA (-)" Or CbTipo.Text = "CONSUMO INTERNO (-)" Or CbTipo.Text = "OTROS (-)" Then
                valor = Integer.Parse(txtActual.Text) - Integer.Parse(txtAjustar.Text)
            Else

```

```

        valor = Integer.Parse(txtActual.Text) + Integer.Parse(txtAjustar.Text)
    End If
    bsnNegocio.AgregarAjustes(
        txtCodigo.Text,
        CbTipo.Text,
        UCase(TxtDescripcion.Text),
        DateTimePicker1.Value,
        Integer.Parse(txtAjustar.Text.Trim())
    )
    bsnNegocio.ActualizarProductos(
        txtCodigo.Text,
        valor
    )
    MsgBox("Ingreso exitoso")
    dgvAjustes.DataSource = bsnNegocio.GetAjustes()
    dgvProductos.DataSource = bsnNegocio.GetProductos()
    BtnActualizar.Visible = True
    txtAjustar.Text = 0
    TxtDescripcion.Text = ""
End If
End Sub
Private Sub dgvProductos_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvProductos.CellClick
    CargarTextbox()
End Sub
Private Sub TxtDescripcion_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtDescripcion.KeyPress
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
End Sub
Private Sub TextBox4_TextChanged(sender As Object, e As EventArgs) Handles txtAjustar.TextChanged
End Sub
Private Sub TextBox4_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtAjustar.KeyPress
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
    If Char.IsWhiteSpace(e.KeyChar) Then
        e.Handled = True
        MsgBox("No se permiten espacios")
    End If
End Sub
Private Sub TxtDescripcion_TextChanged(sender As Object, e As EventArgs) Handles TxtDescripcion.TextChanged
End Sub
Private Sub dgvProductos_SelectionChanged(sender As Object, e As EventArgs) Handles dgvProductos.SelectionChanged
    Try
        Dim cod_producto As String = dgvProductos.SelectedRows(0).Cells(0).Value
        dgvAjustes.DataSource = bsnNegocio.GetAjustesSegunProducto(cod_producto)
    Catch ex As Exception
    End Try
End Sub
End Class

```

Compras Proveedor

```
Imports MySql.Data.MySqlClient
Public Class FrmComprasProveedor
    Dim bsnNegocio As New BsnNegocio
    Dim primero As Byte = 0
    Sub CargarTextBox()
        If Not (dgvProveedoresVentas.Rows.Count = 0) Then
            TxtRut.Text = dgvProveedoresVentas.SelectedRows.Item(0).Cells(0).Value
            TxtNombre.Text = dgvProveedoresVentas.SelectedRows.Item(0).Cells(1).Value
            TxtEmpresa.Text = dgvProveedoresVentas.SelectedRows.Item(0).Cells(2).Value
        End If
    End Sub
    Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles TxtNombre.TextChanged
    End Sub
    Private Sub FrmComprasProveedor_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        dgvProveedoresVentas.DataSource = bsnNegocio.GetProveedores()
        Me.WindowState = 2
        If primero = 0 Then
            primero = 1
        End If
        CargarTextBox()
        dgvProductos.DataSource = bsnNegocio.GetProductos()
        dgvProductos.Rows(0).Selected = True
        CargarDatosProducto()
        CargarCbTipo()
        CalcularTotalCompra()
    End Sub
    Private Sub FrmComprasProveedor_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
MyBase.FormClosing
        If e.CloseReason = CloseReason.UserClosing Then
            e.Cancel = True
            Me.Dispose()
            FrmPrincipal.Show()
        End If
    End Sub
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
        Me.Hide()
        FrmPrincipal.Show()
    End Sub
    Private Sub dgvProveedoresVentas_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles
dgvProveedoresVentas.CellClick
        CargarTextBox()
    End Sub
    Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
        Me.Hide()
        FrmPrincipal.Show()
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs)
        Me.Hide()
        FrmProveedores.Show()
    End Sub
    Private Sub TxtRut_TextChanged(sender As Object, e As EventArgs) Handles TxtRut.TextChanged
    End Sub
    Private Sub dgvProductos_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles
dgvProductos.CellClick
        CargarDatosProducto()
    End Sub
    'Carga datos de producto seleccionado en textbox de formulario
    Public Sub CargarDatosProducto()
        txtCodProducto.Enabled = False
        txtNomProducto.Enabled = False
        txtMarca.Enabled = False
        txtDescripcion.Enabled = False
        cbTipo.Enabled = False
        txtPrecioVenta.Enabled = False
        txtStockCritico.Enabled = False
        chkProductoNuevo.Checked = False
        txtCodProducto.Text = dgvProductos.SelectedRows(0).Cells(0).Value
        txtNomProducto.Text = dgvProductos.SelectedRows(0).Cells(1).Value
        txtMarca.Text = dgvProductos.SelectedRows(0).Cells(2).Value
    End Sub
End Class
```

```

txtDescripcion.Text = dgvProductos.SelectedRows(0).Cells(3).Value
cbTipo.SelectedValue = dgvProductos.SelectedRows(0).Cells(6).Value.ToString()
txtPrecioVenta.Text = dgvProductos.SelectedRows(0).Cells(5).Value
txtStockCritico.Text = dgvProductos.SelectedRows(0).Cells(8).Value
txtCantidad.Clear()
txtPrecioCompra.Clear()
End Sub
Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles btnAgregar.Click
    Dim precioCompra, cantidad As Integer
    Try
        precioCompra = Integer.Parse(txtPrecioCompra.Text.Trim())
        cantidad = Integer.Parse(txtCantidad.Text.Trim())
    Catch ex As Exception
        MsgBox("Indique precio de compra y cantidad del producto")
        Return
    End Try
    If precioCompra <= 0 Or cantidad <= 0 Then
        MsgBox("Precio de compra y cantidad deben ser números positivos")
        Return
    End If
    If txtCodProducto.Text = "" Or txtNomProducto.Text = "" Or txtMarca.Text = "" Or txtDescripcion.Text = "" Then
        MsgBox("Indique todos los datos del producto")
        Return
    End If
    Dim precioVenta, stockCritico As Integer
    Try
        precioVenta = Integer.Parse(txtPrecioVenta.Text.Trim())
        stockCritico = Integer.Parse(txtStockCritico.Text.Trim())
    Catch ex As Exception
        MsgBox("Precio venta y stock crítico deben ser números.")
        Return
    End Try
    If precioVenta <= 0 Then
        MsgBox("Precio de venta debe ser mayor a cero")
        Return
    End If
    If stockCritico < 0 Then
        MsgBox("Stock crítico debe ser mayor o igual a cero")
        Return
    End If
    If precioVenta <= precioCompra Then
        Dim respuesta As Integer = MessageBox.Show("El precio de venta es menor o igual al de compra. ¿Desea continuar?",
"Advertencia", MessageBoxButtons.YesNo)
        If respuesta = DialogResult.No Then
            Return
        End If
    End If
    If cantidad <= stockCritico Then
        Dim respuesta As Integer = MessageBox.Show("La cantidad comprada es menor o igual al stock crítico. ¿Desea
continuar?", "Advertencia", MessageBoxButtons.YesNo)
        If respuesta = DialogResult.No Then
            Return
        End If
    End If
    Dim cod_producto = txtCodProducto.Text.Trim()
    If RevisarProductoEnCompra(cod_producto) Then
        MsgBox("El producto ya está en la compra")
        Return
    End If
    'agregar producto si es nuevo
    If chkProductoNuevo.Checked Then
        cod_producto = bsnNegocio.InsertarProductoDesdeCompra(
            txtCodProducto.Text.Trim(),
            txtNomProducto.Text.Trim(),
            txtMarca.Text.Trim(),
            txtDescripcion.Text.Trim(),
            txtPrecioVenta.Text.Trim(),
            cbTipo.SelectedValue,
            txtCantidad.Text.Trim(),
            txtStockCritico.Text.Trim()

```

```

    )
    If cod_producto Is Nothing Then
        Return
    End If
End If
Dim valores(4) As String
valores(0) = txtCodProducto.Text.Trim()
valores(1) = txtNomProducto.Text.Trim()
valores(2) = txtCantidad.Text.Trim()
valores(3) = txtPrecioCompra.Text.Trim()
valores(4) = Integer.Parse(txtCantidad.Text.Trim()) * Integer.Parse(txtPrecioCompra.Text.Trim())
dgvProductosCompra.Rows.Add(valores)
dgvProductos.DataSource = bsnNegocio.GetProductos()
dgvProductos.Rows(0).Selected = True
CargarDatosProducto()
chkProductoNuevo.Checked = False
txtCantidad.Clear()
txtPrecioCompra.Clear()
CalcularTotalCompra()
End Sub
Private Sub chkProductoNuevo_CheckedChanged(sender As Object, e As EventArgs) Handles
chkProductoNuevo.CheckedChanged
    txtCodProducto.Enabled = chkProductoNuevo.Checked
    txtNomProducto.Enabled = chkProductoNuevo.Checked
    txtMarca.Enabled = chkProductoNuevo.Checked
    txtDescripcion.Enabled = chkProductoNuevo.Checked
    cbTipo.Enabled = chkProductoNuevo.Checked
    txtPrecioVenta.Enabled = chkProductoNuevo.Checked
    txtStockCritico.Enabled = chkProductoNuevo.Checked
    If chkProductoNuevo.Checked Then
        txtCodProducto.Clear()
        txtNomProducto.Clear()
        txtMarca.Clear()
        txtDescripcion.Clear()
        cbTipo.SelectedIndex = 0
        txtPrecioVenta.Clear()
        txtStockCritico.Clear()
        txtPrecioCompra.Clear()
        txtCantidad.Clear()
    Else
        CargarDatosProducto()
    End If
End Sub
Sub CargarCbTipo()
    cbTipo.DropDownStyle = ComboBoxStyle.DropDownList
    cbTipo.DisplayMember = "Text"
    cbTipo.ValueMember = "Value"
    Dim tb As New DataTable
    tb.Columns.Add("Text", GetType(String))
    tb.Columns.Add("Value", GetType(String))
    tb.Rows.Add("BEBIDAS", "BEBIDAS")
    tb.Rows.Add("LICORES", "LICORES")
    tb.Rows.Add("ABARROTES", "ABARROTES")
    tb.Rows.Add("CIGARROS", "CIGARROS")
    cbTipo.DataSource = tb
End Sub
Private Sub btnEliminarProductoCompra_Click(sender As Object, e As EventArgs) Handles btnEliminarProductoCompra.Click
    dgvProductosCompra.Rows.Remove(dgvProductosCompra.SelectedRows(0))
    CalcularTotalCompra()
End Sub
'Determina el total de la compra y lo escribe en el textbox correspondiente
Private Sub CalcularTotalCompra()
    Dim totalcompra As Integer = 0
    For Each venta As DataGridViewRow In dgvProductosCompra.Rows
        totalcompra += Integer.Parse(venta.Cells(4).Value)
    Next
    txtTotalCompra.Text = totalcompra
End Sub

```

```

'Revisa si producto ya está en la compra
Private Function RevisarProductoEnCompra(cod_producto As String)
    For Each compra As DataGridViewRow In dgvProductosCompra.Rows
        If compra.Cells(0).Value.ToString().Equals(cod_producto) Then
            Return True
        End If
    Next
    Return False
End Function
Private Sub btnComprar_Click(sender As Object, e As EventArgs) Handles btnComprar.Click
    If dgvProductosCompra.Rows.Count <= 0 Then
        MsgBox("No ha incluido productos en la compra")
        Return
    End If
    bsnNegocio.InsertarCompraProductos(TxtRut.Text.Trim(), dgvProductosCompra)
    CargarTextBox()
    dgvProductos.DataSource = bsnNegocio.GetProductos()
    dgvProductos.Rows(0).Selected = True
    CargarDatosProducto()
    dgvProductosCompra.Rows.Clear()
    CalcularTotalCompra()
    MsgBox("Compra ingresada")
End Sub
Private Sub btnVerCompras_Click(sender As Object, e As EventArgs) Handles btnVerCompras.Click
    Dim ventana As New FrmComprasListar(Me)
    ventana.Show()
End Sub
End Class

```

Login y Principal

```

Public Class FrmPrincipal
    Dim usuario As Usuario
    Dim formInicial As FrmLogin
    Dim bsnNegocio As New BsnNegocio
    Public Sub New(ByRef formInicial As FrmLogin, ByRef usuario As Usuario)
        InitializeComponent()
        Me.usuario = usuario
        Me.formInicial = formInicial
        'CargarStockCritico()
    End Sub
    Public Sub New()
        InitializeComponent()
        'CargarStockCritico()
    End Sub
    Private Sub Button5_Click(sender As Object, e As EventArgs)
        formInicial.Show()
        Me.Dispose()
    End Sub
    Private Sub FrmPrincipal_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.WindowState = 2
    End Sub
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        Me.Hide()
        FrmMantenedores.Show()
    End Sub
    Private Sub Label3_Click(sender As Object, e As EventArgs) Handles hora.Click
    End Sub
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        hora.Text = TimeOfDay
    End Sub
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
        Me.Hide()
        FrmInformes.Show()
    End Sub
    Private Sub FrmPrincipal_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing

        If e.CloseReason = CloseReason.UserClosing Then
            e.Cancel = True

```

```

        Me.Dispose()
        FrmLogin.Show()
    End If
End Sub
Private Sub Button5_Click_1(sender As Object, e As EventArgs) Handles Button5.Click
    FrmLogin.Show()
    Me.Dispose()
End Sub
Private Sub Panel1_Paint(sender As Object, e As PaintEventArgs) Handles Panel1.Paint
End Sub
Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles txtNomUser.TextChanged
    'Dim usr As New Usuario
    'txtNomUser.Text = usr.NombreUsuario
End Sub
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    Me.Hide()
    FrmComprasProveedor.Show()
End Sub
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Me.Hide()
    FrmVentas.Show()
End Sub
Private Sub TextBox1_VisibleChanged(sender As Object, e As EventArgs) Handles txtNomUser.VisibleChanged
End Sub
Private Sub Label2_Click(sender As Object, e As EventArgs) Handles Label2.Click
End Sub
Private Sub CargarStockCritico()
    dgvAlertaStock.DataSource = bsnNegocio.GetProductosBajoStock()
End Sub
Private Sub FrmPrincipal_Shown(sender As Object, e As EventArgs) Handles MyBase.Shown
    CargarStockCritico()
End Sub
End Class

```

Mantenedor Productos

```

Imports MySql.Data.MySqlClient

Public Class FrmProductos

    Dim bsnNegocio As New BsnNegocio

    Dim tipo As New Integer

    Dim primera As Byte = 0

    Dim modo As Integer = 0

    Dim opc As String

    Public Sub New()
        ' Esta llamada es exigida por el diseñador.
        InitializeComponent()
        ' Agregue cualquier inicialización después de la llamada a InitializeComponent().
    End Sub

    Sub Recarga()

        Me.FrmProductos_Load(Me, Nothing)

        'TxtFiltro.Text = ""

    End Sub

    'Sub CargarCb1()

    '    cbFiltro.DropDownStyle = ComboBoxStyle.DropDownList

    '    cbFiltro.DisplayMember = "Text"

    '    cbFiltro.ValueMember = "Value"

```

```

' Dim tb As New DataTable
' tb.Columns.Add("Text", GetType(String))
' tb.Columns.Add("Value", GetType(Integer))
' tb.Rows.Add("Sin Filtro", 0)
' tb.Rows.Add("Codigo", 1)
' tb.Rows.Add("Nombre", 2)
' tb.Rows.Add("Marca", 3)
' cbFiltro.DataSource = tb
'End Sub

Sub CargarCb2()
    cbEstado.DropDownStyle = ComboBoxStyle.DropDownList
    cbEstado.DisplayMember = "Text"
    cbEstado.ValueMember = "Value"
    Dim tb As New DataTable
    tb.Columns.Add("Text", GetType(String))
    tb.Columns.Add("Value", GetType(Integer))
    tb.Rows.Add("DISPONIBLE", 0)
    tb.Rows.Add("NO DISPONIBLE", 1)
    cbEstado.DataSource = tb
End Sub

Sub CargarCb3()
    cbTipo.DropDownStyle = ComboBoxStyle.DropDownList
    cbTipo.DisplayMember = "Text"
    cbTipo.ValueMember = "Value"
    Dim tb As New DataTable
    tb.Columns.Add("Text", GetType(String))
    tb.Columns.Add("Value", GetType(Integer))
    tb.Rows.Add("BEBIDAS", 0)
    tb.Rows.Add("LICORES", 1)
    tb.Rows.Add("ABARROTES", 2)
    tb.Rows.Add("CIGARROS", 3)
    cbTipo.DataSource = tb
End Sub

Sub CargarTextBox()
    If Not (dgvProductos.Rows.Count = 0) Then
        TxtCodigo.Text = dgvProductos.SelectedRows.Item(0).Cells(0).Value
        TxtNombre.Text = dgvProductos.SelectedRows.Item(0).Cells(1).Value
        TxtMarca.Text = dgvProductos.SelectedRows.Item(0).Cells(2).Value
        TxtDescripcion.Text = dgvProductos.SelectedRows.Item(0).Cells(3).Value
        cbEstado.Text = dgvProductos.SelectedRows.Item(0).Cells(4).Value
        TxtPrecioVenta.Text = dgvProductos.SelectedRows.Item(0).Cells(5).Value
        cbTipo.Text = dgvProductos.SelectedRows.Item(0).Cells(6).Value
        TxtStockActual.Text = dgvProductos.SelectedRows.Item(0).Cells(7).Value
    End If
End Sub

```

```

        TxtStockCritico.Text = dgvProductos.SelectedRows.Item(0).Cells(8).Value
    End If
End Sub

Private Sub FrmProductos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    dgvProductos.DataSource = bsnNegocio.GetProductos()

    Me.WindowState = 2
    If primera = 0 Then
        'CargarCb1()
        CargarCb2()
        CargarCb3()
        primera = 1
    End If
    'Filtrar()
    CargarTextBox()
    TxtCodigo.Enabled = False
    TxtNombre.Enabled = False
    TxtMarca.Enabled = False
    TxtDescripcion.Enabled = False
    TxtPrecioVenta.Enabled = False
    TxtStockActual.Enabled = False
    TxtStockCritico.Enabled = False
    cbEstado.Enabled = False
    cbTipo.Enabled = False
End Sub

Private Sub FrmProductos_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing
    If e.CloseReason = CloseReason.UserClosing Then
        e.Cancel = True
        Me.Dispose()
        FrmMantenedores.Show()
    End If
End Sub

Private Sub Button4_Click(sender As Object, e As EventArgs)
    Me.Close()
End Sub

Private Sub dgvProductos_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvProductos.CellClick
    CargarTextBox()
End Sub

Private Sub cbFiltro_SelectedIndexChanged(sender As Object, e As EventArgs)
End Sub

'Private Sub TxtFiltro_TextChanged(sender As Object, e As EventArgs) Handles TxtFiltro.TextChanged

Private Sub BtnSalir_Click(sender As Object, e As EventArgs) Handles BtnSalir.Click
    FrmMantenedores.Show()
    Me.Dispose()

```

```

End Sub

Private Sub BtnAgregar_Click(sender As Object, e As EventArgs) Handles BtnAgregar.Click
    opc = "Agregar Producto"
    BtnActualizar.Visible = False
    BtnEliminar.Visible = False
    BtnAgregar.Visible = False
    BtnSalir.Visible = False
    dgvProductos.Enabled = False
    BtnAceptar.Visible = True
    BtnCancelar.Visible = True
    TxtCodigo.Enabled = True
    TxtNombre.Enabled = True
    TxtMarca.Enabled = True
    TxtDescripcion.Enabled = True
    TxtPrecioVenta.Enabled = True
    TxtStockActual.Enabled = True
    TxtStockCritico.Enabled = True
    cbEstado.Enabled = True
    cbTipo.Enabled = True
    TxtCodigo.Text = ""
    TxtNombre.Text = ""
    TxtMarca.Text = ""
    TxtDescripcion.Text = ""
    TxtPrecioVenta.Text = ""
    TxtStockActual.Text = ""
    TxtStockCritico.Text = ""
End Sub

Private Sub BtnAceptar_Click(sender As Object, e As EventArgs) Handles BtnAceptar.Click
    Dim bsnNegocio As New BsnNegocio
    Dim obj As New DaoDatos
    Dim prod As New Productos
    If opc = "Agregar Producto" Then
        If TxtCodigo.Text = "" Or TxtNombre.Text = "" Or TxtMarca.Text = "" Or TxtDescripcion.Text = "" Or TxtPrecioVenta.Text = "" Or TxtStockActual.Text = "" Or TxtStockCritico.Text = "" Then
            MsgBox("Por favor, debe rellenar todos los campos", MsgBoxStyle.Information, "Alerta")
            Return
        End If
        If TxtStockActual.Text < TxtStockCritico.Text Then
            MsgBox("El Stock Actual es menor que el Stock Crítico")
        End If
        Dim respuesta = bsnNegocio.InsertarProductos(
            UCase(TxtCodigo.Text),
            UCase(TxtNombre.Text),
            UCase(TxtMarca.Text),

```

```

        UCase(TxtDescripcion.Text),
        cbEstado.Text,
        Integer.Parse(TxtPrecioVenta.Text),
        cbTipo.Text,
        Integer.Parse(TxtStockActual.Text),
        Integer.Parse(TxtStockCritico.Text))

If respuesta Is Nothing Then
    MsgBox("Ingreso fallido, producto ya existe")
    Return
End If
MsgBox("Ingreso Exitoso")
dgvProductos.DataSource = bsnNegocio.GetProductos()
BtnActualizar.Visible = True
BtnEliminar.Visible = True
BtnAgregar.Visible = True
BtnSalir.Visible = True
dgvProductos.Enabled = True
BtnAceptar.Visible = False
BtnCancelar.Visible = False
TxtCodigo.Enabled = False
TxtNombre.Enabled = False
cbTipo.Enabled = False
TxtMarca.Enabled = False
TxtDescripcion.Enabled = False
TxtPrecioVenta.Enabled = False
TxtStockActual.Enabled = False
TxtStockCritico.Enabled = False
cbEstado.Enabled = False
cbTipo.Enabled = False
End If

If opc = "Actualizar Productos" Then
    If TxtMarca.Text = "" Or TxtDescripcion.Text = "" Or TxtPrecioVenta.Text = "" Or TxtStockActual.Text = "" Or
    TxtStockCritico.Text = "" Then
        MsgBox("Por favor, debe rellenar todos los campos", MsgBoxStyle.Information, "Alerta")
        Return
    End If
    If Int(TxtStockActual.Text) < Int(TxtStockCritico.Text) Then
        MsgBox("El stock actual es menor que el crítico")
    End If
    bsnNegocio.ActualizarProductos(
        TxtCodigo.Text,
        UCase(TxtNombre.Text),
        UCase(TxtMarca.Text),

```

```

        UCase(TxtDescripcion.Text),
        cbEstado.Text,
        Integer.Parse(TxtPrecioVenta.Text),
        cbTipo.Text,
        Integer.Parse(TxtStockActual.Text),
        Integer.Parse(TxtStockCritico.Text))
MsgBox("Actualizacion Exitosa")
dgvProductos.DataSource = bsnNegocio.GetProductos()

BtnActualizar.Visible = True
BtnEliminar.Visible = True
BtnAgregar.Visible = True
BtnSalir.Visible = True
dgvProductos.Enabled = True
BtnAceptar.Visible = False
BtnCancelar.Visible = False
TxtCodigo.Enabled = False
TxtNombre.Enabled = False
cbTipo.Enabled = False
TxtMarca.Enabled = False
TxtDescripcion.Enabled = False
TxtPrecioVenta.Enabled = False
TxtStockActual.Enabled = False
TxtStockCritico.Enabled = False
cbEstado.Enabled = False
cbTipo.Enabled = False

End If
End Sub

Private Sub BtnCancelar_Click(sender As Object, e As EventArgs) Handles BtnCancelar.Click
    CargarTextBox()
    BtnActualizar.Visible = True
    BtnEliminar.Visible = True
    BtnAgregar.Visible = True
    BtnSalir.Visible = True
    dgvProductos.Enabled = True
    BtnAceptar.Visible = False
    BtnCancelar.Visible = False
    TxtCodigo.Enabled = False
    TxtNombre.Enabled = False
    cbTipo.Enabled = False
    TxtMarca.Enabled = False
    TxtDescripcion.Enabled = False
    TxtPrecioVenta.Enabled = False
    TxtStockActual.Enabled = False

```

```

    TxtStockCritico.Enabled = False

    cbEstado.Enabled = False

    cbTipo.Enabled = False

End Sub

Private Sub dgvProductos_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles
dgvProductos.CellContentClic
End Sub

Private Sub BtnActualizar_Click(sender As Object, e As EventArgs) Handles BtnActualizar.Click

    Dim negocio As New BsnNegocio

    opc = "Actualizar Productos"

    BtnActualizar.Visible = False

    BtnEliminar.Visible = False

    BtnAgregar.Visible = False

    BtnSalir.Visible = False

    BtnAceptar.Visible = True

    BtnCancelar.Visible = True


    TxtCodigo.Enabled = False

    TxtNombre.Enabled = True

    cbTipo.Enabled = True

    TxtMarca.Enabled = True

    TxtDescripcion.Enabled = True

    TxtPrecioVenta.Enabled = True

    TxtStockActual.Enabled = True

    TxtStockCritico.Enabled = True

    cbEstado.Enabled = True

    cbTipo.Enabled = True

End Sub

Private Sub BtnEliminar_Click(sender As Object, e As EventArgs) Handles BtnEliminar.Click

    opc = "Eliminar Productos"

    BtnActualizar.Visible = True

    BtnEliminar.Visible = True

    BtnAgregar.Visible = True

    BtnSalir.Visible = True

    BtnAceptar.Visible = False 'ACEPTAR

    BtnCancelar.Visible = False 'CANCELAR

    TxtCodigo.Enabled = False

    TxtNombre.Enabled = False

    cbTipo.Enabled = False

    TxtMarca.Enabled = False

    TxtDescripcion.Enabled = False

    TxtPrecioVenta.Enabled = False

    TxtStockActual.Enabled = False

    TxtStockCritico.Enabled = False

```

```

cbEstado.Enabled = False
cbTipo.Enabled = False
BtnActualizar.Enabled = False
BtnEliminar.Enabled = True
BtnAgregar.Enabled = False
BtnSalir.Enabled = False
If opc = "Eliminar Productos" Then
    If dgvProductos.CurrentRow Is Nothing Then
        MsgBox("Error! Debe seleccionar un registro de la tabla.", MsgBoxStyle.Critical, "Alerta")
    Else
        If MessageBox.Show("¿Está seguro que desea eliminar el producto?", "Alerta", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Question) = Windows.Forms.DialogResult.OK Then
            Dim respuesta = bsnNegocio.EliminarProductos(TxtCodigo.Text.Trim())
            Select Case respuesta
                Case -1 : MsgBox("El producto no se puede eliminar porque tiene Ajustes")
                Case -2 : MsgBox("El producto no se puede eliminar porque tiene Compras")
                Case -3 : MsgBox("El producto no se puede eliminar porque tiene Ventas")
                Case Else
                    MsgBox("Producto eliminado correctamente", MsgBoxStyle.Information, "Alerta")
                    Recarga()
            End Select
        End If
    End If
End If
BtnActualizar.Enabled = True
BtnEliminar.Enabled = True
BtnAgregar.Enabled = True
BtnSalir.Enabled = True
End Sub
Private Sub TxtMarca_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtMarca.KeyPress
    If Char.IsLetter(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If

    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If

```

```

End Sub

Private Sub TxtStockActual_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtStockActual.KeyPress
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
    If Char.IsWhiteSpace(e.KeyChar) Then
        e.Handled = True
        MsgBox("No se permiten espacios")
    End If
End Sub

Private Sub TxtStockCritico_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtStockCritico.KeyPress
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
    If Char.IsWhiteSpace(e.KeyChar) Then
        e.Handled = True
        MsgBox("No se permiten espacios")
    End If
End Sub

Private Sub TxtPrecioCompra_KeyPress(sender As Object, e As KeyPressEventArgs)
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then

```

```

        e.Handled = False
    Else
        e.Handled = True
    End If

    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If

    If Char.IsWhiteSpace(e.KeyChar) Then
        e.Handled = True
        MsgBox("No se permiten espacios")
    End If
End Sub

Private Sub TxtPrecioVenta_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtPrecioVenta.KeyPress
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If

    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If

    If Char.IsWhiteSpace(e.KeyChar) Then
        e.Handled = True
        MsgBox("No se permiten espacios")
    End If
End Sub

Private Sub TxtCodigo_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtCodigo.KeyPress
    If Char.IsNumber(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsControl(e.KeyChar) Then
        e.Handled = False
    ElseIf Char.IsSeparator(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If

    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If

```

```

End If
If Char.IsWhiteSpace(e.KeyChar) Then
    e.Handled = True
    MsgBox("No se permiten espacios")
End If
End Sub
Private Sub TxtNombre_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtNombre.KeyPress
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
End Sub
Private Sub TxtNombre_TextChanged(sender As Object, e As EventArgs) Handles TxtNombre.TextChanged
End Sub
Private Sub TxtDescripcion_TextChanged(sender As Object, e As EventArgs) Handles TxtDescripcion.TextChanged
End Sub
Private Sub TxtDescripcion_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TxtDescripcion.KeyPress
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
End Sub
Private Sub TxtMarca_TextChanged(sender As Object, e As EventArgs) Handles TxtMarca.TextChanged
End Sub
Private Sub TxtStockActual_TextChanged(sender As Object, e As EventArgs) Handles TxtStockActual.TextChanged
End Sub
Private Sub TxtStockCritico_TextChanged(sender As Object, e As EventArgs) Handles TxtStockCritico.TextChanged
End Sub
Private Sub TxtPrecioCompra_TextChanged(sender As Object, e As EventArgs)
End Sub
Private Sub TxtPrecioVenta_TextChanged(sender As Object, e As EventArgs) Handles TxtPrecioVenta.TextChanged
End Sub
Private Sub Button5_Click(sender As Object, e As EventArgs)
End Sub
Private Sub TxtFiltro_KeyPress(sender As Object, e As KeyPressEventArgs)
    If InStr(1, "" & Chr(8), e.KeyChar) = 1 Then
        e.KeyChar = ""
    End If
End Sub
End Sub
End Class.

```

Ventas.

```
Imports System.ComponentModel
Imports TrabajoBotilleria.FrmDeudores
Public Class FrmVentas
    Dim bsnNegocio As New BsnNegocio
    Sub Recarga()
        dgvProdVenta.DataSource = bsnNegocio.GetProductos()
    End Sub
    Sub CargarTextBox()
        If Not (dgvProdVenta.Rows.Count = 0) Then
            TxtCodigo.Text = dgvProdVenta.SelectedRows.Item(0).Cells(0).Value
            TxtNombre.Text = dgvProdVenta.SelectedRows.Item(0).Cells(1).Value
            TxtPrecio.Text = dgvProdVenta.SelectedRows.Item(0).Cells(5).Value
        End If
    End Sub
    Public Sub CargarCbDeudor()
        Dim bsn As New BsnNegocio()
        CbRut.DropDownStyle = ComboBoxStyle.DropDownList
        CbRut.DataSource = bsn.GetDeudores()
        CbRut.DisplayMember = "NOMBRE_DEUDOR"
        CbRut.ValueMember = "RUT_DEUDOR"
    End Sub
    Private Sub GroupBox1_Enter(sender As Object, e As EventArgs) Handles GroupBox1.Enter
    End Sub
    Private Sub FrmVentas_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.WindowState = 2
        dgvProdVenta.DataSource = bsnNegocio.GetProductos()
        dgvProdVenta.Columns(6).Visible = False
        dgvProdVenta.Columns(8).Visible = False
        CargarTextBox()
        CargarCbDeudor()
        TxtCodigo.enabled = False
        TxtNombre.enabled = False
        TxtPrecio.Enabled = False
    End Sub
    Private Sub FrmVentas_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing
        If e.CloseReason = CloseReason.UserClosing Then
            e.Cancel = True
            Me.Dispose()
            FrmPrincipal.Show()
        End If
    End Sub
    Private Sub Button8_Click(sender As Object, e As EventArgs) Handles btnSalir.Click
        Me.Hide()
        FrmPrincipal.Show()
    End Sub
    Private Sub dtgProdVenta_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles
dgvProdVenta.CellContentClick
    End Sub
    Private Sub dgvProdVenta_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvProdVenta.CellClick
        CargarTextBox()
    End Sub
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles btnAgregar.Click
        If txtCantidad.Text = "" Or TxtDcto.Text = "" Then
            MsgBox("Debe indicar cantidad y descuento. Si no hay descuento escriba 0")
            Return
        End If
        Dim cantidad, descuento As Integer
        Try
            cantidad = Integer.Parse(txtCantidad.Text.Trim())
            descuento = Integer.Parse(TxtDcto.Text.Trim())
        Catch ex As Exception
            MsgBox("Cantidad y descuento deben ser números")
            Return
        End Try
        If cantidad <= 0 Or descuento < 0 Then
            MsgBox("Cantidad debe ser mayor a cero y descuento mayor o igual a cero")
            Return
        End If
    End Sub
```

```

If RevisarProductoEnVenta(TxtCodigo.Text.Trim()) Then
    MsgBox("El producto ya está en la venta")
    Return
End If
Dim valores(6) As String
valores(0) = TxtCodigo.Text
valores(1) = TxtNombre.Text
valores(2) = TxtPrecio.Text
valores(3) = txtCantidad.Text
valores(4) = TxtDcto.Text
valores(5) = (Integer.Parse(TxtPrecio.Text) * Integer.Parse(txtCantidad.Text.Trim())) - Integer.Parse(TxtDcto.Text)
dgvProductosVentas.Rows.Add(valores)
CalcularTotalVenta()
txtCantidad.Text = 1
TxtDcto.Text = 0
End Sub
Private Sub btnAgregarDeu_Click(sender As Object, e As EventArgs) Handles btnAgregarDeu.Click
    Dim ventana As New FrmDeudores(Me)
    ventana.Show()
End Sub
Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles btnEliminar.Click
    dgvProductosVentas.Rows.Remove(dgvProductosVentas.SelectedRows(0))
    CalcularTotalVenta()
End Sub
Private Sub btnCancelar_Click(sender As Object, e As EventArgs) Handles btnCancelar.Click
    dgvProductosVentas.Rows.Clear()
    CalcularTotalVenta()
End Sub
Private Sub btnNuevaVenta_Click(sender As Object, e As EventArgs) Handles btnNuevaVenta.Click
    dgvProductosVentas.Rows.Clear()
    CalcularTotalVenta()
    txtPagoEfectivo.Text = 0
End Sub
'Calcula total de venta a medida que se agregar/quitan items
Private Sub CalcularTotalVenta()
    Dim totalventa As Integer = 0
    For Each venta As DataGridViewRow In dgvProductosVentas.Rows
        totalventa += Integer.Parse(venta.Cells(5).Value)
    Next
    txtTotalVenta.Text = totalventa
End Sub
'Revisa si producto ya está en la venta
Private Function RevisarProductoEnVenta(cod_producto As String)
    For Each venta As DataGridViewRow In dgvProductosVentas.Rows
        If venta.Cells(0).Value.ToString().Equals(cod_producto) Then
            Return True
        End If
    Next
    Return False
End Function
Private Sub TextBox3_TextChanged(sender As Object, e As EventArgs)
End Sub
Private Sub txtPagoEfectivo_TextChanged(sender As Object, e As EventArgs) Handles txtPagoEfectivo.TextChanged
End Sub

Private Sub CbRut_SelectedIndexChanged(sender As Object, e As EventArgs) Handles CbRut.SelectedIndexChanged
End Sub

Private Sub btnPagar_Click(sender As Object, e As EventArgs) Handles btnPagar.Click
    Dim totalventa As Integer = Integer.Parse(txtTotalVenta.Text)
    Dim pagoEfectivo As Integer = Integer.Parse(txtPagoEfectivo.Text)
    Dim vuelto As Integer = pagoEfectivo - totalventa

    If (vuelto < 0) Or pagoEfectivo <= 0 Then
        MsgBox("El monto de pago no es suficiente")
        Return
    End If

```

```

If totalventa = 0 Then
    MsgBox("Debe incluir productos en la venta")
    Return
End If

Dim lista_productos As New List(Of ProductoVenta)

For Each producto_venta As DataGridViewRow In dgvProductosVentas.Rows
    Dim prod As New ProductoVenta()
    prod.cod_producto = producto_venta.Cells(0).Value
    prod.cantidad = Integer.Parse(producto_venta.Cells(3).Value)
    prod.descuento = Integer.Parse(producto_venta.Cells(4).Value)
    lista_productos.Add(prod)
Next
bsnNegocio.IngresarVenta(lista_productos)

'resetea form venta
dgvProductosVentas.Rows.Clear()
CalcularTotalVenta()
dgvProdVenta.DataSource = bsnNegocio.GetProductos()
txtPagoEfectivo.Text = 0
MsgBox("Venta Ingresada. Vuelto: $" & vuelto)
End Sub

Private Sub btnDeuda_Click(sender As Object, e As EventArgs) Handles btnDeuda.Click
    'ingresar la venta normalmente
    Dim totalventa As Integer = Integer.Parse(txtTotalVenta.Text)

    If totalventa = 0 Then
        MsgBox("Debe incluir productos en la venta")
        Return
    End If

    Dim lista_productos As New List(Of ProductoVenta)

    For Each producto_venta As DataGridViewRow In dgvProductosVentas.Rows
        Dim prod As New ProductoVenta()
        prod.cod_producto = producto_venta.Cells(0).Value
        prod.cantidad = Integer.Parse(producto_venta.Cells(3).Value)
        prod.descuento = Integer.Parse(producto_venta.Cells(4).Value)
        lista_productos.Add(prod)
    Next
    Dim rut_deudor As String = CbRut.SelectedValue.ToString()
    bsnNegocio.IngresarVentaConDeuda(lista_productos, rut_deudor)

    dgvProductosVentas.Rows.Clear()
    CalcularTotalVenta()
    dgvProdVenta.DataSource = bsnNegocio.GetProductos()
    CbRut.SelectedIndex = 0
    MsgBox("Venta con deuda ingresada")
End Sub

Private Sub btnVerVentas_Click(sender As Object, e As EventArgs) Handles btnVerVentas.Click
    Dim ventana As New FrmVentasListar(Me)
    ventana.Show()
End Sub
End Class

```