**DOCUMENTACIÓN DEL PROGRAMADOR**

* **Especificaciones de Programación**

**Tabla Clientes:**

1. **Encabezado:**

ABCM Clientes

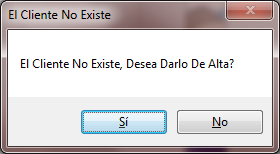
1. **Descripción:**

Programa para generar: Altas, Bajas, Consultas y Modificaciones, de la tabla/clase Clientes.

1. **Interfaz:**



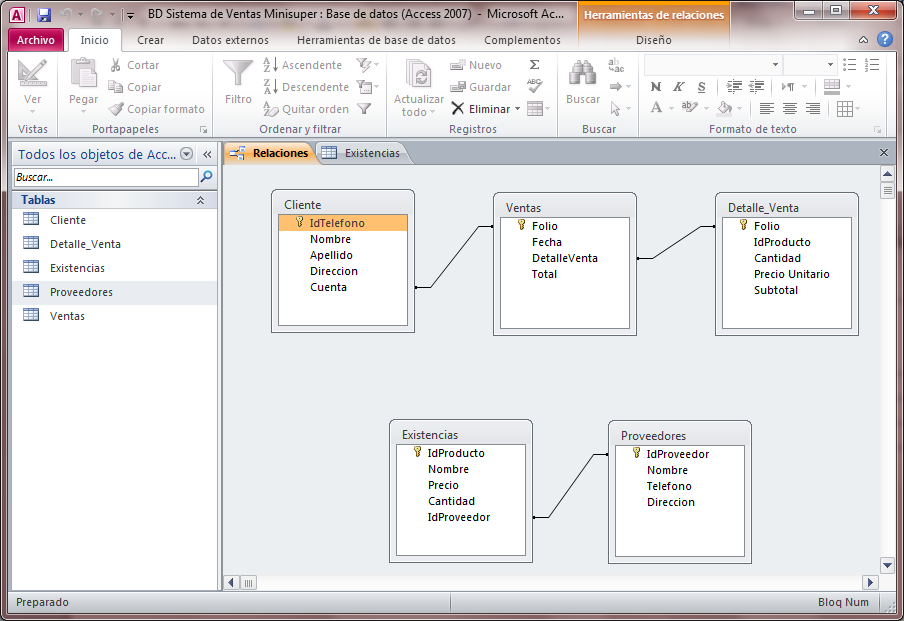
1. **Ventana Para Mensajes:**



1. **Base de Datos:**

BD Sistema de Ventas Minisuper

1. **Tabla:**



1. **Validaciones de Dominio:**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Tamaño** |
| IdTelefono | Entero | 7 o 10 dig. |
| Nombre | Cadena | 3 a 15 dig. |
| Apellido | Cadena | 3 a 15 dig. |
| Direccion | Cadena | 10 a 40 dig. |
| Cuenta | Flotante | 2 a 5 dig. 2 decimales |

1. **Validación de Integridad Referencial:**

IdTelefono Único

1. **Validación de las Reglas del Negocio:**

IdTelefono debe ser un numero existente.

1. **Secuencia del Proceso:**

Altas:

* Ingresar el teléfono del cliente.
* Oprimir botón buscar.
* Buscar el teléfono en la base de datos
* Si existe: mostrar los datos del cliente para realizar venta.
* Si no existe: abrir interfaz para ingresar los datos del nuevo cliente
* Oprimir aceptar para dar de alta al nuevo cliente.

Bajas:

* En el menú principal seleccionar Bajas
* Seleccionar el submenú Baja Cliente
* De la lista de clientes mostrada en el dialogo seleccionar el cliente que se desea dar de baja.
* Oprimir el botón baja.
* Confirmar la solicitud.

Modificaciones:

* En el menú principal seleccionar Modificaciones
* Seleccionar el submenú Modificación Cliente
* De la lista de clientes mostrada en el dialogo seleccionar el cliente que se desea modificar.
* Oprimir el botón modificar.
* Hacer las modificaciones necesarias.
* Oprimir aceptar para guardar los cambios.

Consultas:

* En el menú principal seleccionar Consultas
* Seleccionar el submenú Consulta Cliente
* En el dialogo ingresar el IdTelefono del Cliente.
* Oprimir el botón Buscar.
* Si existe se mostraran los datos del Cliente.
* Si no existe se mostrara un mensaje.

1. **Colores:**

Azul RGB(36, 14, 196) y

Blanco RGB(255, 255, 255).

1. **Tipo de Letra:**

Times New Roman 12

**Tabla Proveedores:**

1. **Encabezado:**

ABCM Proveedores

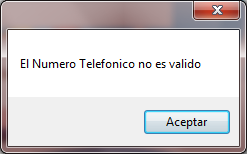
1. **Descripción:**

Programa para generar: Altas, Bajas, Consultas y Modificaciones, de la tabla/clase Proveedores.

1. **Interfaz:**



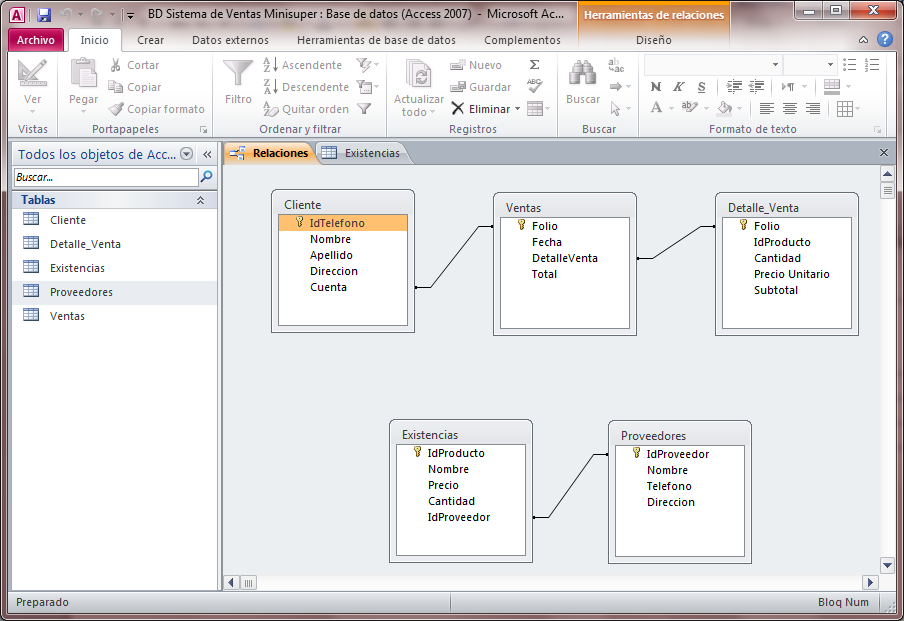
1. **Ventana Para Mensajes:**



1. **Base de Datos:**

BD Sistema de Ventas Minisuper

1. **Tabla:**



1. **Validaciones de Dominio:**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Tamaño** |
| IdProveedor | Entero | 7 o 10 dig. |
| Nombre | Cadena | 3 a 15 dig. |
| Telefono | Cadena | 7 0 10 dig. |
| Direccion | Cadena | 10 a 40 dig. |

1. **Validación de Integridad Referencial:**

IdProveedor Único

1. **Validación de las Reglas del Negocio:**

IdProveedor debe ser un número existente, que tenga registrado el Proveedor.

1. **Secuencia del Proceso:**

Altas:

* En la ventana pedidos oprimir el botón nuevo proveedor.
* En la nueva venta ingresar los datos.
* Oprimir el botón aceptar.
* Buscar el IdProveedor en la base de datos
* Si existe: mostrar un mensaje.
* Si no existe: validar los demás datos.
* Si los datos esta validados dar de alta al proveedor.

Bajas:

* En el menú principal seleccionar Bajas.
* Seleccionar el submenú Baja Proveedor.
* De la lista de proveedores mostrada en el dialogo seleccionar el proveedor que se desea dar de baja.
* Oprimir el botón baja.
* Confirmar la solicitud.

Modificaciones:

* En el menú principal seleccionar Modificaciones
* Seleccionar el submenú Modificación Proveedor
* De la lista de Proveedores mostrada en el dialogo seleccionar el Proveedor que se desea modificar.
* Oprimir el botón modificar.
* Hacer las modificaciones necesarias.
* Oprimir aceptar para guardar los cambios.

Consultas:

* En el menú principal seleccionar Consultas
* Seleccionar el submenú Consulta Proveedor
* En el dialogo ingresar el IdProveedor del Proveedor.
* Oprimir el botón Buscar.
* Si existe se mostraran los datos del Proveedor.
* Si no existe se mostrara un mensaje.

1. **Colores:**

Azul RGB(36, 14, 196) y

Blanco RGB(255, 255, 255

1. **Tipo de Letra:**

Times New Roman 12

**Tabla Existencias:**

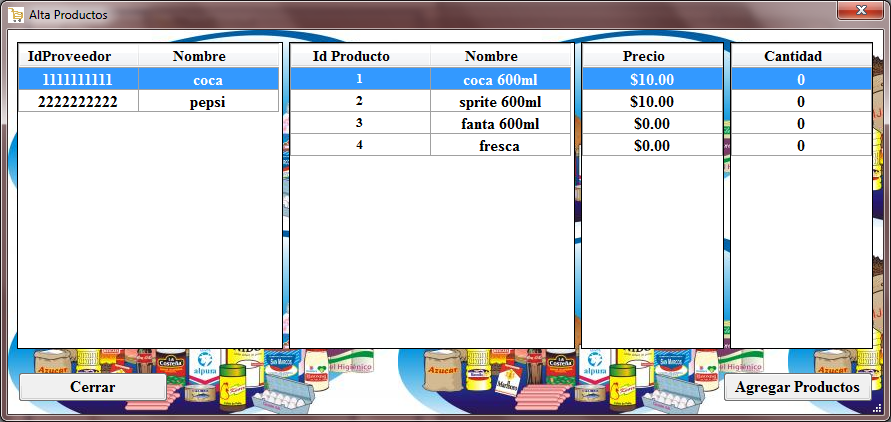
1. **Encabezado:**

ABCM Existencias

1. **Descripción:**

Programa para generar: Altas, Bajas, Consultas y Modificaciones, de la tabla/clase Existencias.

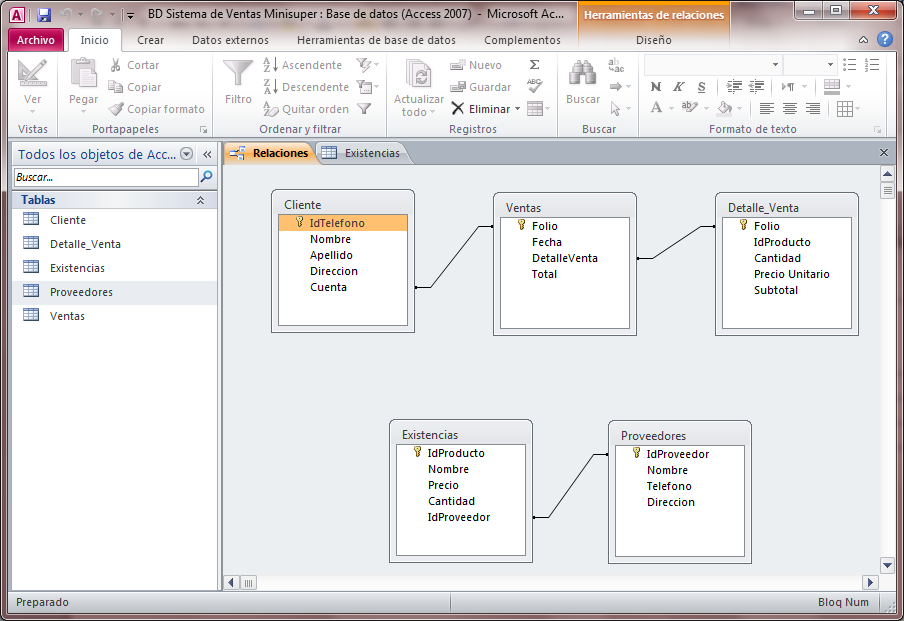
1. **Interfaz:**



1. **Base de Datos:**

BD Sistema de Ventas Minisuper

1. **Tabla:**



1. **Validaciones de Dominio:**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Tamaño** |
| IdProducto | Entero | 1 a 4 dig. |
| Nombre | Cadena | 3 a 15 dig. |
| Precio | Flotante | Decimales |
| Cantidad | Entero | 1 a 4 dig. |
| IdProveedor | Entero | 7 o 10 dig. |

1. **Validación de Integridad Referencial:**

IdProducto Único

IdProveedor debe existir en la tabla/clase Proveedores.

1. **Validación de las Reglas del Negocio:**

Precio > 0

1. **Secuencia del Proceso:**

Altas:

* En la ventana de pedidos oprimir el botón alta producto.
* En la nueva venta seleccionar el proveedor.
* Seleccionar el producto.
* Modificar el precio si es necesario.
* Ingresar la cantidad.
* Oprimir el aceptar para dar de alta el producto.

Bajas:

* Al Momento de la venta se dará de baja automáticamente.
* En el menú principal seleccionar Bajas.
* Seleccionar el submenú Baja Producto.
* De la lista de productos mostrada en el dialogo seleccionar el producto que se desea dar de baja.
* Oprimir el botón baja.
* Confirmar la solicitud.

Modificaciones:

* En el menú principal seleccionar Modificaciones
* Seleccionar el submenú Modificación Producto
* De la lista de Productos mostrada en el dialogo seleccionar el Producto que se desea modificar.
* Oprimir el botón modificar.
* Hacer las modificaciones necesarias.
* Oprimir aceptar para guardar los cambios.

Consultas:

* En el menú principal seleccionar Consultas
* Seleccionar el submenú Consulta Producto
* En el dialogo ingresar el IdProducto del Producto.
* Oprimir el botón Buscar.
* Si existe se mostraran los datos del Producto.
* Si no existe se mostrara un mensaje.

1. **Colores:**

Azul RGB(36, 14, 196) y

Blanco RGB(255, 255, 255

1. **Tipo de Letra:**

Times New Roman 12

**Programa para registrar AC de Ventas:**

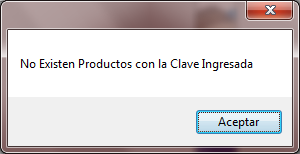
1. **Descripción:**

Programa para generar: Altas, Bajas y Consultas, de la tabla/clase Ventas.

1. **Interfaz:**



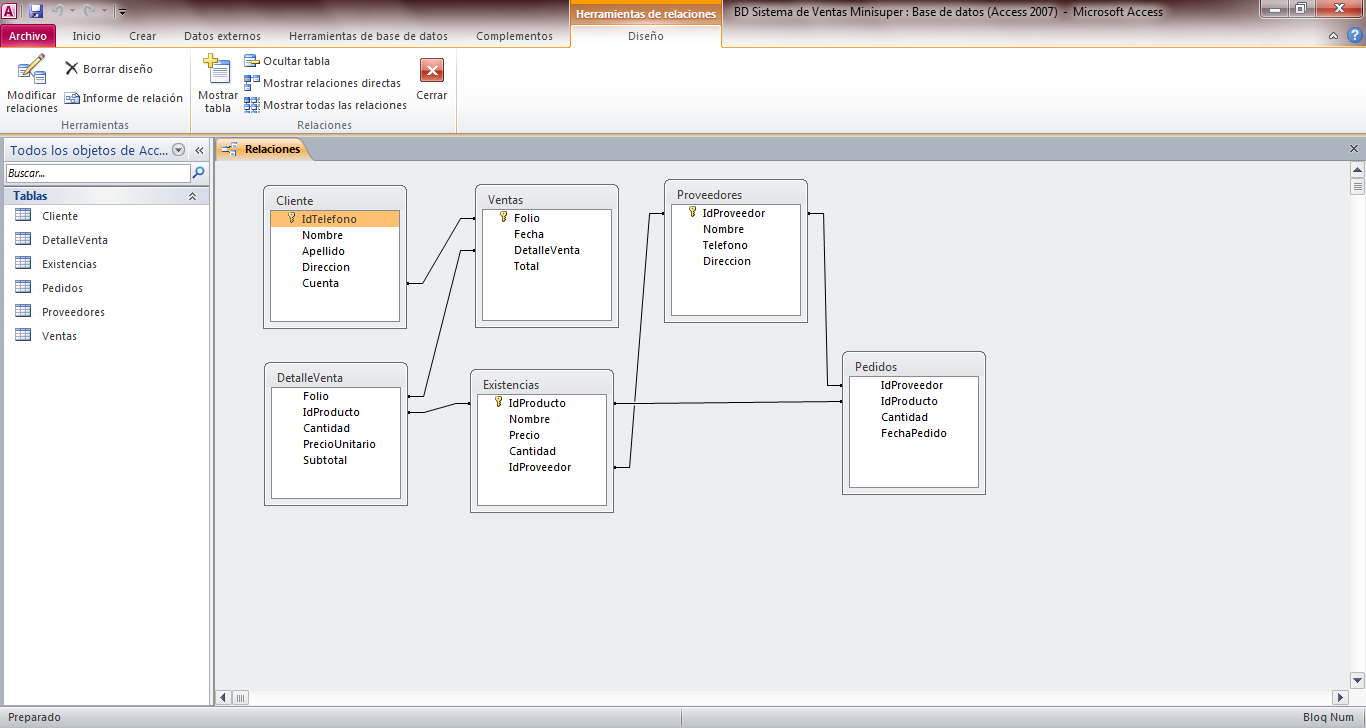
1. **Ventanas Adicionales:**



1. **Base de Datos:**

BD Sistema de Ventas Minisuper

1. **Tabla:**



1. **Validaciones de Dominio:**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Tamaño** |
| Folio | Entero | * 0 dig. |
| Fecha | Cadena | 8 dig. |
| Detalle de Venta | - | - |
| Total | Flotante | 2 a 5 dig. 2 decimales |

1. **Validación de Integridad Referencial:**

Id\_cli en Clientes

Id\_Prod debe estar registrado en existencias

Subtotal = precio \* cantidad calculo automático

Total = Total + Subtotales calculo automático

1. **Validación de las Reglas del Negocio:**

Subtotal y Total > 0

1. **Secuencia del Proceso:**

Altas:

* En la Ventana ingresar el Id de cliente si es necesario.
* Se busca el cliente y se muestran su datos.
* Ingresar Id Producto.
* Si existe se muestran los datos del producto, si no existe muestra un mensaje.
* Si existe se puede ingresar la cantidad del producto.
* Se calcula es subtotal y se muestra
* Se calcula el Total y se muestra.
* Oprimir aceptar para dar de alta la venta.
* Los productos de la venta se descuentan de existencias.

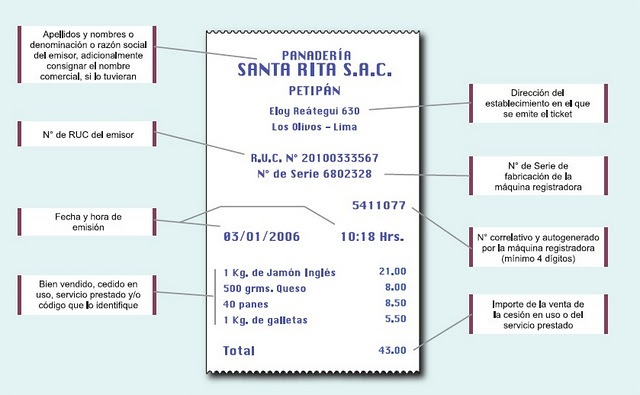
Bajas:

* En el menú principal seleccionar Bajas
* Seleccionar el submenú Baja Venta.
* De la lista de Ventas mostrada en el dialogo seleccionar el Folio de la venta que se desea dar de baja.
* Oprimir el botón baja.
* Confirmar la solicitud.
* Agregar los productos de la venta a existencias.

Consultas:

* En el menú principal seleccionar Consultas
* Seleccionar el submenú Consulta Venta
* En el dialogo ingresar el Folio de la Venta.
* Oprimir el botón Buscar.
* Si existe se mostraran los datos de la Venta.
* Si no existe se mostrara un mensaje.

1. **Diseño de Salida de Información:**



1. **Colores:**

Azul RGB(36, 14, 196) y

Blanco RGB(255, 255, 255

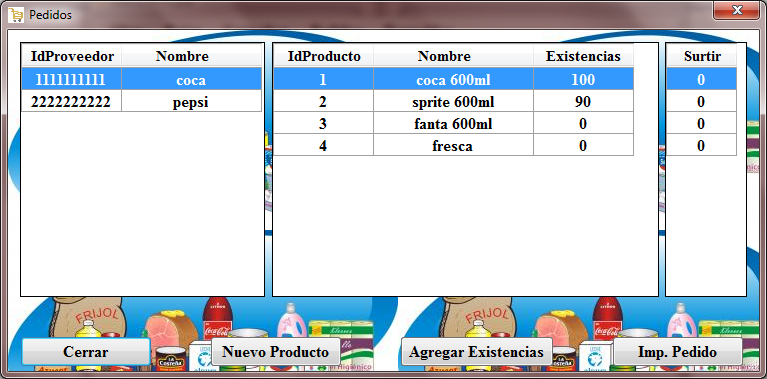
1. **Tipo de Letra:**

Times New Roman 12

**Programa para registrar AC de Pedidos:**

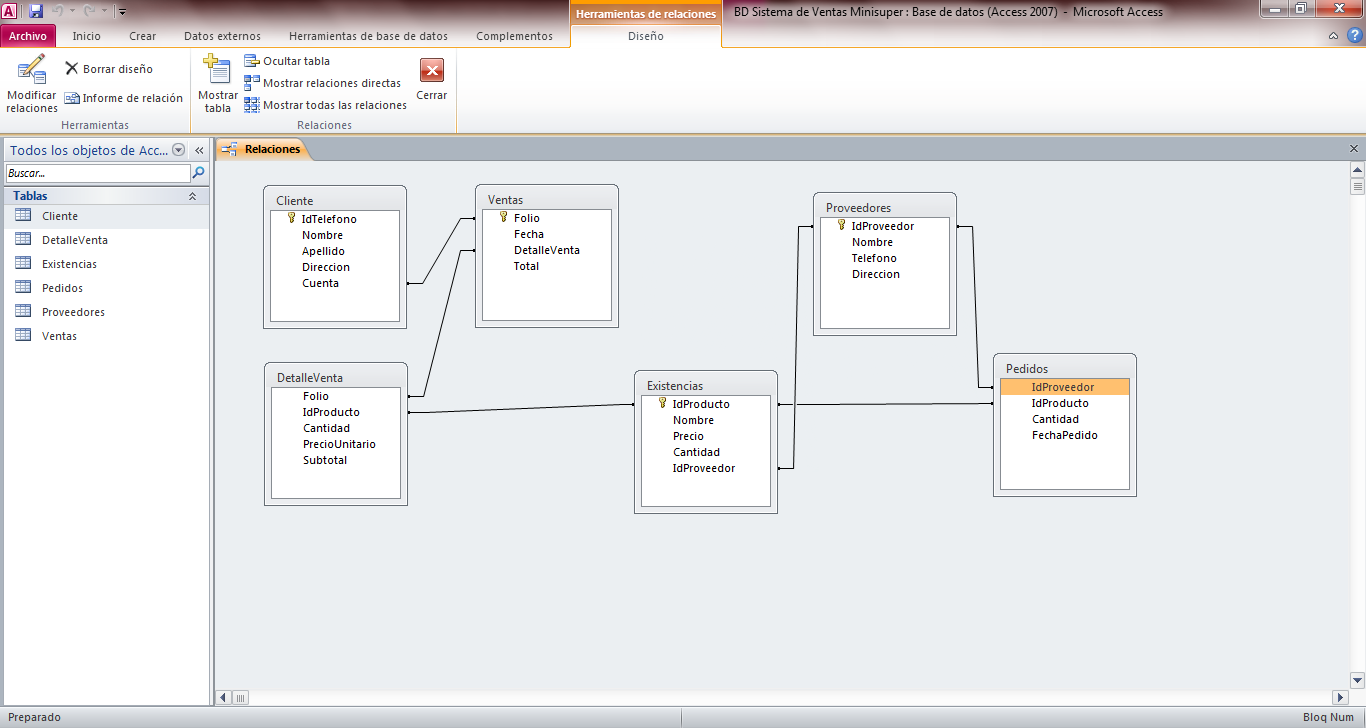
1. **Descripción:**

Programa para generar: Altas y Consultas, de la tabla/clase Pedidos.

1. **Interfaz:** 
2. **Base de Datos:**

BD Sistema de Ventas Minisuper

1. **Tabla:**



1. **Validaciones de Dominio:**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Tamaño** |
| IdProveedor | Entero | * 8 dig. |
| IdProducto | Entero | 8 dig. |
| Cantidad | Entero | 4 dig |
| Fecha | Cadena | 8 dig |

1. **Validación de Integridad Referencial:**

IdProveedor en Proveedores

Id\_Producto en Existencias

1. **Validación de las Reglas del Negocio:**

Cantidad > 0

1. **Secuencia del Proceso:**

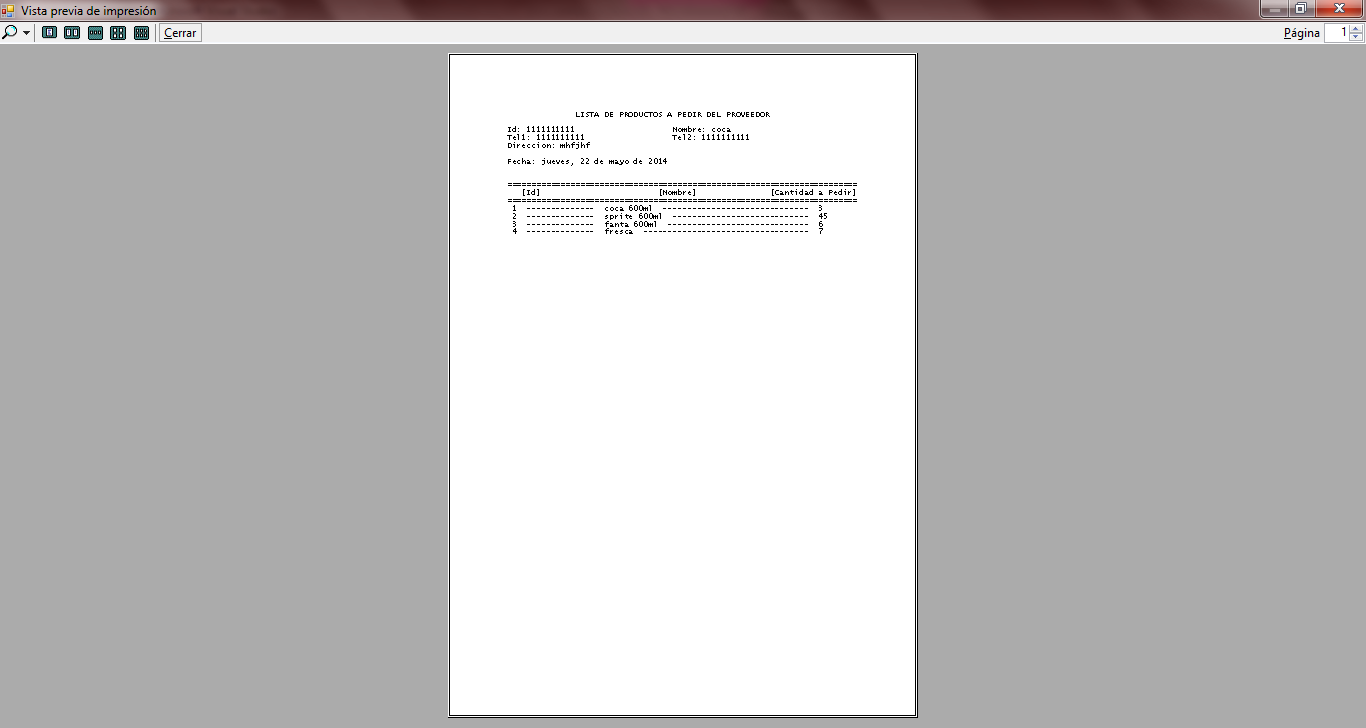
Altas:

* En la columna surtir ingresar la cantidades a pedir de cada producto del proveedor seleccionado.
* Al dar clic en imp Pedido
* Hace un recorrido por la columna
* Todos los productos que tengan cantidad mayor a cero.
* Se guardan en la tabla pedidos
* Se imprime un documento con la lista de productos a pedir

Consultas:

* En el menú principal seleccionar Reportes
* En la parte inferior de la ventana.
* Se encuentra la lista de pedidos.
* Ordenando por fecha.

1. **Diseño de Salida de Información:**



1. **Colores:**

Azul RGB(36, 14, 196) y

Blanco RGB(255, 255, 255

1. **Tipo de Letra:**

Times New Roman 12

* **Documentación de Código**
* Esta clase se encarga de mostrar la carga de la aplicación y el menú principal, además de crear y mostrar las diferentes interfaces con las que se trabajaran

public partial class FormPrincipal : Form

{

/\*==================================================

\* Cadena de Conexion que se utilizara para manipular la base de Datos

==================================================\*/

private string conexion = Properties.Settings.Default.CadenaConexion;

/\*==================================================

\* Constructor de clase FormPrincipal

\* Es el primer metodo que se ejecuta, antes de cargar la venta

\* del menu principal, crea y destruye la simulación del cargador

\* de la aplicación

\* ==================================================\*/

public FormPrincipal()

{

Thread t = new Thread(new ThreadStart(SplashScreen));

t.Start();

Thread.Sleep(12000);

t.Abort();

InitializeComponent();

}

/\*===================================================

\* Este metodo crea un direcctorio en C:

\* con que se trabajara mientras la aplicación esta en ejecución

===================================================\*/

private void FormPrincipal\_Load(object sender, EventArgs e)

{

System.IO.Directory.CreateDirectory("C:\\temporalSW");

}

/\*===================================================

\* Este metodo ejecuta la simulacion del cargador de la aplicación

====================================================\*/

public void SplashScreen()

{

Application.Run(new FormPreload());

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el botón Venta

\* Crea y muestra la venta de una nueva venta

\*==================================================\*/

private void bVenta\_Click(object sender, EventArgs e)

{

FormVenta fv = new FormVenta();

fv.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el botón Pedido

\* Crea y muestra la ventana para hacer un pedido al Proveedor

\*==================================================\*/

private void bPedido\_Click(object sender, EventArgs e)

{

FormPedidos fp = new FormPedidos();

fp.ShowDialog();

}

/\*==================================================

\* Este metodos se ejecuto cuando el usuario da clic en el botón Reportes

\* Creo y muestra la ventana de Reportes donde se muestran las ventas

\* de determinado periodo y las existencias

\*==================================================\*/

private void bReportes\_Click(object sender, EventArgs e)

{

FormReportes fr = new FormReportes();

fr.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Consulta+Clientes

\* Crea y muestra la interfaz donde se muestran los clientes

\*==================================================\*/

private void mClientes\_Click(object sender, EventArgs e)

{

DClientes dc = new DClientes();

dc.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Consulta+Proveedores

\* Crea y muestra la interfaz donde se muestran los porveedores

\*==================================================\*/

private void mProveedores\_Click(object sender, EventArgs e)

{

DCProveedores dcp = new DCProveedores();

dcp.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Alta+Clientes

\* Crea y muestra la interfaz donde se ingresaran los datos de un nuevo cliente

\*==================================================\*/

private void mACliente\_Click(object sender, EventArgs e)

{

FormAltaCliente fac = new FormAltaCliente();

fac.tbTelefono.Enabled = true;

fac.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Alta+Proveedor

\* Crea y muestra la interfaz donde se ingresaran los datos de un nuevo Proveedor

\*==================================================\*/

private void mAProveedor\_Click(object sender, EventArgs e)

{

FormAltaPP fap = new FormAltaPP();

fap.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Consulta+Productos

\* Crea y muestra la interfaz donde se muestran los productos

\*==================================================\*/

private void mProductos\_Click(object sender, EventArgs e)

{

DCProductos dcp = new DCProductos();

dcp.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Baja+Clientes

\*==================================================\*/

private void mBCliente\_Click(object sender, EventArgs e)

{

FormBajaCliente fbc = new FormBajaCliente();

fbc.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Baja+Proveedor

\*==================================================\*/

private void mBProveedor\_Click(object sender, EventArgs e)

{

DCProveedores dcp = new DCProveedores();

dcp.bElimina.Visible = true;

dcp.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando el usuario da clic en el menu Baja+Proveedor

\*==================================================\*/

private void mBProducto\_Click(object sender, EventArgs e)

{

DCProductos dcp = new DCProductos();

dcp.bElimina.Visible = true;

dcp.ShowDialog();

}

private void mPClientes\_Click(object sender, EventArgs e)

{

FormVenta fv = new FormVenta();

fv.tbTelefono.Enabled = true;

fv.bBuscarCli.Enabled = true;

fv.tbTelefono.TabIndex = 0;

fv.bImpRegVenta.Visible = false;

fv.bVCaja.Visible = false;

fv.bVDomicilio.Visible = false;

fv.bVPendiente.Visible = false;

fv.bCancelar.Visible = false;

fv.bAceptar.Visible = false;

fv.bImpPedido.Visible = true;

fv.lFolio.Visible = false;

fv.tbFolio.Visible = false;

fv.tbCliente.Text = "";

fv.Text = "Tomar Pedido de Cliente";

fv.ShowDialog();

}

/\*==================================================

\* Este metodo se ejecuta cuando se esta cerrando la aplicacion

\* elimina la carpeta creada al inicio de la aplicacion

\*==================================================\*/

private void FormPrincipal\_FormClosing(object sender, FormClosingEventArgs e)

{

System.IO.Directory.Delete("C:\\temporalSW");

}

/\*==================================================

\* Este metodo se encarga de generar los dialgos

\* y respaldar la base de datos con la que se esta trabajando

\*==================================================\*/

private void menuRespaldar\_Click(object sender, EventArgs e)

{

string miConexion;

miConexion = conexion.Substring(conexion.LastIndexOf("\\") + 1);

miConexion = miConexion.Substring(0, miConexion.Length - 1);

dRespaldar.Filter = "ficheros .accdb (\*.accdb) | \*.accdb";

if (dRespaldar.ShowDialog() == DialogResult.OK)

{

if (File.Exists(dRespaldar.FileName))

MessageBox.Show("No Se Puedo Sobreescribir un respaldo");

else

{

File.Copy(miConexion, dRespaldar.FileName);

MessageBox.Show("El Respaldo Se Ha Realizado Exitosamente");

}

}

}

/\*==================================================

\* Este metodo se encarga de recuperar y cargar una base de datos

\* antes respaldada por el usuario

\*==================================================\*/

private void menuImportar\_Click(object sender, EventArgs e)

{

dImportar.Filter = "ficheros .accdb (\*.accdb) | \*.accdb";

if (dImportar.ShowDialog() == DialogResult.OK)

{

#region Existencias

try

{

OleDbConnection MyConnection = new OleDbConnection(conexion);

string query = "DELETE FROM Existencias";

OleDbCommand cmd = new OleDbCommand(query, MyConnection);

OleDbDataAdapter delete = new OleDbDataAdapter();

MyConnection.Open();

delete.DeleteCommand = cmd;

delete.DeleteCommand.ExecuteNonQuery();

MyConnection.Close();

}

catch { MessageBox.Show("No se Puede"); }

using (OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source= "+dImportar.FileName))

{

string query = "SELECT \* FROM Existencias";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into Existencias(IdProducto, Nombre, Precio, Cantidad, IdProveedor)values (@iprod, @nprod, @pprod, @cprod, @iprov)";

OleDbCommand cmd2 = new OleDbCommand(insert, cnn);

cmd2.Parameters.AddWithValue("@iprod", reader.GetString(0));

cmd2.Parameters.AddWithValue("@nprod", reader.GetString(1));

cmd2.Parameters.AddWithValue("@pprod", reader.GetFloat(2));

cmd2.Parameters.AddWithValue("@cprod", reader.GetInt16(3));

cmd2.Parameters.AddWithValue("@iprov", reader.GetString(4));

cnn.Open();

int exe = cmd2.ExecuteNonQuery();

cnn.Close();

}

}

reader.Close();

}

#endregion

#region Clientes

try

{

OleDbConnection MyConnection = new OleDbConnection(conexion);

string query = "DELETE FROM Cliente";

OleDbCommand cmd = new OleDbCommand(query, MyConnection);

OleDbDataAdapter delete = new OleDbDataAdapter();

MyConnection.Open();

delete.DeleteCommand = cmd;

delete.DeleteCommand.ExecuteNonQuery();

MyConnection.Close();

}

catch { MessageBox.Show("No se Puede"); }

using (OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source= " + dImportar.FileName))

{

string query = "SELECT \* FROM Cliente";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into Cliente(IdTelefono, Nombre, Apellido, Direccion, Cuenta)values (@iprod, @nprod, @pprod, @cprod, @iprov)";

OleDbCommand cmd2 = new OleDbCommand(insert, cnn);

cmd2.Parameters.AddWithValue("@iprod", reader.GetString(0));

cmd2.Parameters.AddWithValue("@nprod", reader.GetString(1));

cmd2.Parameters.AddWithValue("@pprod", reader.GetString(2));

cmd2.Parameters.AddWithValue("@cprod", reader.GetString(3));

cmd2.Parameters.AddWithValue("@iprov", "0");

cnn.Open();

int exe = cmd2.ExecuteNonQuery();

cnn.Close();

}

}

reader.Close();

}

#endregion

#region Proveedores

try

{

OleDbConnection MyConnection = new OleDbConnection(conexion);

string query = "DELETE FROM Proveedores";

OleDbCommand cmd = new OleDbCommand(query, MyConnection);

OleDbDataAdapter delete = new OleDbDataAdapter();

MyConnection.Open();

delete.DeleteCommand = cmd;

delete.DeleteCommand.ExecuteNonQuery();

MyConnection.Close();

}

catch { MessageBox.Show("No se Puede"); }

using (OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source= " + dImportar.FileName))

{

string query = "SELECT \* FROM Proveedores";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into Proveedores(IdProveedor, Nombre, Telefono, Direccion)values (@iprod, @nprod, @pprod, @cprod)";

OleDbCommand cmd2 = new OleDbCommand(insert, cnn);

cmd2.Parameters.AddWithValue("@iprod", reader.GetString(0));

cmd2.Parameters.AddWithValue("@nprod", reader.GetString(1));

cmd2.Parameters.AddWithValue("@pprod", reader.GetString(2));

cmd2.Parameters.AddWithValue("@cprod", reader.GetString(3));

cnn.Open();

int exe = cmd2.ExecuteNonQuery();

cnn.Close();

}

}

reader.Close();

}

#endregion

#region Ventas

try

{

OleDbConnection MyConnection = new OleDbConnection(conexion);

string query = "DELETE FROM Ventas";

OleDbCommand cmd = new OleDbCommand(query, MyConnection);

OleDbDataAdapter delete = new OleDbDataAdapter();

MyConnection.Open();

delete.DeleteCommand = cmd;

delete.DeleteCommand.ExecuteNonQuery();

MyConnection.Close();

}

catch { MessageBox.Show("No se Puede"); }

using (OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source= " + dImportar.FileName))

{

string query = "SELECT \* FROM Ventas";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into Ventas(Folio, Fecha, DetalleVenta, Total)values (@iprod, @nprod, @pprod, @cprod)";

OleDbCommand cmd2 = new OleDbCommand(insert, cnn);

cmd2.Parameters.AddWithValue("@iprod", reader.GetInt32(0));

cmd2.Parameters.AddWithValue("@nprod", reader.GetString(1));

cmd2.Parameters.AddWithValue("@pprod", reader.GetInt32(2));

cmd2.Parameters.AddWithValue("@cprod", reader.GetFloat(3));

cnn.Open();

int exe = cmd2.ExecuteNonQuery();

cnn.Close();

}

}

reader.Close();

}

#endregion

#region DetalleVenta

try

{

OleDbConnection MyConnection = new OleDbConnection(conexion);

string query = "DELETE FROM DetalleVenta";

OleDbCommand cmd = new OleDbCommand(query, MyConnection);

OleDbDataAdapter delete = new OleDbDataAdapter();

MyConnection.Open();

delete.DeleteCommand = cmd;

delete.DeleteCommand.ExecuteNonQuery();

MyConnection.Close();

}

catch { MessageBox.Show("No se Puede"); }

using (OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source= " + dImportar.FileName))

{

string query = "SELECT \* FROM DetalleVenta";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into DetalleVenta(Folio, IdProducto, Cantidad, PrecioUnitario, Subtotal)values (@iprod, @nprod, @pprod, @cprod, @iprov)";

OleDbCommand cmd2 = new OleDbCommand(insert, cnn);

cmd2.Parameters.AddWithValue("@iprod", reader.GetInt32(0));

cmd2.Parameters.AddWithValue("@nprod", reader.GetString(1));

cmd2.Parameters.AddWithValue("@pprod", reader.GetInt16(2));

cmd2.Parameters.AddWithValue("@cprod", reader.GetFloat(3));

cmd2.Parameters.AddWithValue("@iprov", reader.GetFloat(4));

cnn.Open();

int exe = cmd2.ExecuteNonQuery();

cnn.Close();

}

}

reader.Close();

}

#endregion

MessageBox.Show("La Importacion Se Ha Realizado Exitosamente");

}

}

}

* Esta clase se encarga de generar y registrar las ventas y generar un ticket, además de actualizar las existencias.

public partial class FormVenta : Form

{

/\*=======================================================================

Instancias de la clase

\*=======================================================================\*/

private string conexion = Properties.Settings.Default.CadenaConexion;

private CExistencias producto;

private CVentas venta;

private List<int> lfolios = new List<int>();

private int indC;

private Font Fuente;

private StreamReader streamParaImp;

/\*=======================================================================

Constructor de la clase

\*=======================================================================\*/

public FormVenta()

{

InitializeComponent();

leerRegistroVentas();

ponerFolio();

}

private void FormVenta\_Load(object sender, EventArgs e)

{

}

/\*=======================================================================

Este metodo activa o desactiva controles en el formulario

\*=======================================================================\*/

private void bVCaja\_Click(object sender, EventArgs e)

{

tbTelefono.Enabled = false;

bBuscarCli.Enabled = false;

tbDireccion.Enabled = false;

tbCliente.Text = "Venta de Caja";

tbTelefono.Text = "";

tbDireccion.Text = "";

dgvProductos.Focus();

}

/\*=======================================================================

Este metodo activa o desactiva controles en el formulario

\*=======================================================================\*/

private void bVDomicilio\_Click(object sender, EventArgs e)

{

tbTelefono.Enabled = true;

bBuscarCli.Enabled = true;

tbDireccion.Enabled = true;

tbCliente.Text = "";

tbTelefono.Focus();

}

/\*=======================================================================

Este metodo se encarga de buscar un cliente en la base de datos

si lo encutra muestra sus datos, sino da la opción de agregarlo

\*=======================================================================\*/

private void bBuscarCli\_Click(object sender, EventArgs e)

{

dgvProductos.Focus();

bool b=false;

if (tbTelefono.TextLength < 7 || (tbTelefono.TextLength > 7 && tbTelefono.TextLength < 10) || tbTelefono.TextLength > 10)

{

MessageBox.Show("El Numero Telefonico No Es Valido", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

tbTelefono.Focus();

}

else

{

b = buscarRegistro(tbTelefono.Text);

if (!b)

{

if (MessageBox.Show("El Cliente No Existe, Desea Darlo De Alta?", "El Cliente No Existe", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)

{

FormAltaCliente fac = new FormAltaCliente();

fac.tbTelefono.Text = tbTelefono.Text;

if (fac.ShowDialog() == DialogResult.OK)

{

tbTelefono.Text = fac.tbTelefono.Text;

tbCliente.Text = fac.tbNombre.Text + " " + fac.tbApellido.Text;

tbDireccion.Text = fac.tbDireccion.Text;

}

}

else

{

tbTelefono.Text = "";

tbTelefono.Focus();

}

}

else

{

leerRegistro(tbTelefono.Text);

}

}

}

/\*=======================================================================

Este metodo hace la consulta para la busqueda en la base de datos

\*=======================================================================\*/

private bool buscarRegistro(string id)

{

using (OleDbConnection cn = new OleDbConnection(conexion))

{

string query = "SELECT COUNT(\*) FROM Cliente WHERE IdTelefono=@Id";

OleDbCommand cmd = new OleDbCommand(query, cn);

cmd.Parameters.AddWithValue("Id", id);

cn.Open();

int count = Convert.ToInt32(cmd.ExecuteScalar());

if (count == 0)

return false;

else

return true;

}

}

/\*=======================================================================

Este metodo busca los datos del cliente

\*=======================================================================\*/

private void leerRegistro(string id)

{

using (OleDbConnection cn = new OleDbConnection(conexion))

{

string query = "SELECT IdTelefono, Nombre, Apellido, Direccion FROM Cliente WHERE IdTelefono=@Id";

OleDbCommand cmd = new OleDbCommand(query, cn);

cmd.Parameters.AddWithValue("Id", id);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

tbTelefono.Text = reader.GetString(0);

tbCliente.Text = reader.GetString(1);

tbCliente.Text +=" "+ reader.GetString(2);

tbDireccion.Text = reader.GetString(3);

}

cn.Close();

}

}

/\*=======================================================================

Este metodo actualiza el datagrid de los productos

\*=======================================================================\*/

private void dgvProductos\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

Point ind = dgvProductos.CurrentCellAddress;

if(productoRegistrado(ind) == false)

if (ind.X == 0 && dgvProductos[ind.X, ind.Y].Value != null)

{

string idp = dgvProductos[ind.X, ind.Y].Value.ToString();

leerRegistroExistencias(idp);

if (producto != null)

{

dgvDescripcion.Rows.Add();

dgvDescripcion[0, ind.Y].Value = producto.nombre;

dgvDescripcion[1, ind.Y].Value = producto.precio.ToString("C2");

dgvCantidad.Rows.Add();

dgvCantidad[0, ind.Y].Value = "1";

float precio = Convert.ToSingle(dgvDescripcion[1, ind.Y].Value.ToString().Substring(1));

int cantidad = Convert.ToInt32(dgvCantidad[0, ind.Y].Value.ToString());

dgvSubtotales[0, ind.Y].Value = (precio \* cantidad).ToString("C2");

}

else

{

MessageBox.Show("No Existen Productos Con La Clave Ingresada", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

dgvProductos.Rows.RemoveAt(ind.Y);

if (dgvDescripcion.RowCount > ind.Y)

{

dgvDescripcion.Rows.RemoveAt(ind.Y);

dgvCantidad.Rows.RemoveAt(ind.Y);

dgvSubtotales.Rows.RemoveAt(ind.Y);

}

dgvProductos.CurrentCell = dgvProductos[0, dgvProductos.RowCount - 1];

}

}

}

/\*=======================================================================

Este metodo busca la clave del producto ingresado

\*=======================================================================\*/

private void leerRegistroExistencias(string id)

{

using (OleDbConnection cn = new OleDbConnection(conexion))

{

string query = "SELECT IdProducto, Nombre, Precio, Cantidad, IdProveedor FROM Existencias WHERE IdProducto=@Id";

OleDbCommand cmd = new OleDbCommand(query, cn);

cmd.Parameters.AddWithValue("Id", id);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

producto = new CExistencias();

producto.idProducto = Convert.ToInt64(reader.GetString(0));

producto.nombre = reader.GetString(1);

producto.precio = reader.GetFloat(2);

producto.cantidad = reader.GetInt16(3);

producto.idProveedor = Convert.ToInt64(reader.GetString(4));

//lE.Add(producto);

}

}

else

{

producto = null;

}

reader.Close();

}

}

/\*=======================================================================

Este metodo actualiza los subtotales segun los datos de la

columna cantidad

\*=======================================================================\*/

private void dgvCantidad\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

dgvCantidad\_CellEndEdit(sender, e);

Point ind = dgvCantidad.CurrentCellAddress;

if (ind != new Point(-1, -1) && dgvProductos.RowCount - 1 > ind.Y)

{

verificaExistencias(e.RowIndex);

if (dgvCantidad.RowCount /\*-1\*/> dgvSubtotales.RowCount)

{

float precio = Convert.ToSingle(dgvDescripcion[1, ind.Y].Value.ToString().Substring(1));

int cantidad = Convert.ToInt32(dgvCantidad[0, ind.Y].Value.ToString());

dgvSubtotales.Rows.Add((precio \* cantidad).ToString("C2"));

}

else

{

int i = indC;//dgvCantidad.CurrentRow.Index;

float precio = Convert.ToSingle(dgvDescripcion[1, i].Value.ToString().Substring(1));

int cantidad = Convert.ToInt32(dgvCantidad[0, i].Value.ToString());

dgvSubtotales[0, i].Value = (precio \* cantidad).ToString("C2");

}

}

else

{

if (dgvProductos.RowCount - 1 <= ind.Y)

dgvCantidad.Rows.RemoveAt(dgvCantidad.RowCount - 2);

}

}

/\*=======================================================================

Este metodo actualiza el total cuando se modifica la columna subtotales

\*=======================================================================\*/

private void dgvSubtotales\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

Point ind = dgvCantidad.CurrentCellAddress;

int i=0;

float total=0;

if (ind != new Point(-1, -1))

{

for(i=0;i<dgvSubtotales.RowCount;i++)

total+=Convert.ToSingle(dgvSubtotales[0,i].Value.ToString().Substring(1));

}

tbTotal.Text = total.ToString("C2");

}

/\*=======================================================================

Este evento se ejecuta cuando se agrega una fila al datagrid

subtotales

\*=======================================================================\*/

private void dgvSubtotales\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

Point ind = dgvCantidad.CurrentCellAddress;

int i = 0;

float total = 0;

if (ind != new Point(-1, -1))

{

for (i = 0; i < dgvSubtotales.RowCount; i++)

total += Convert.ToSingle(dgvSubtotales[0, i].Value.ToString().Substring(1));

}

tbTotal.Text = Convert.ToString(total);

}

/\*=======================================================================

Este evento se ejecuto cuando se eliman una fila del datagrid

subtotales

\*=======================================================================\*/

private void dgvSubtotales\_RowsRemoved(object sender, DataGridViewRowsRemovedEventArgs e)

{

Point ind = dgvCantidad.CurrentCellAddress;

int i = 0;

float total = 0;

if (ind != new Point(-1, -1))

{

for (i = 0; i < dgvSubtotales.RowCount; i++)

total += Convert.ToSingle(dgvSubtotales[0, i].Value.ToString().Substring(1));

}

tbTotal.Text = total.ToString("C2");

}

/\*=======================================================================

Este metodo muestra los datos del produto ingresado

\*=======================================================================\*/

public bool productoRegistrado(Point ind)

{

int i, cantidad=0;

for (i = 0; i < dgvProductos.RowCount - 1; i++)

{

if (dgvProductos[0, i].Value != null && dgvProductos[0, ind.Y].Value != null && dgvProductos[0, i].Value.ToString() == dgvProductos[0, ind.Y].Value.ToString() && i != ind.Y)

{

indC = i;

cantidad = Convert.ToInt32(dgvCantidad[0, i].Value.ToString());

cantidad++;

dgvCantidad[0, i].Value = cantidad;

dgvProductos.Rows.RemoveAt(ind.Y);

if (dgvDescripcion.RowCount > ind.Y)

{

dgvDescripcion.Rows.RemoveAt(ind.Y);

dgvCantidad.Rows.RemoveAt(ind.Y);

dgvSubtotales.Rows.RemoveAt(ind.Y);

}

dgvProductos.CurrentCell = dgvProductos[0, dgvProductos.RowCount - 1];

return true;

}

}

return false;

}

/\*=======================================================================

Este metodo lee las ventas para saber cual folio sigue

\*=======================================================================\*/

private void leerRegistroVentas()

{

using (OleDbConnection cn = new OleDbConnection(conexion))

{

string query = "SELECT \* FROM Ventas";

OleDbCommand cmd = new OleDbCommand(query, cn);

cn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

venta = new CVentas();

venta.folio = reader.GetInt32(0);

venta.fecha = reader.GetString(1);

venta.detalleVenta = reader.GetInt32(2);

venta.total = reader.GetFloat(3);

lfolios.Add(venta.folio);

}

}

else

{

venta = null;

}

reader.Close();

}

}

/\*=======================================================================

Este valida los datos de la interfaz y si son correctos registra la venta

\*=======================================================================\*/

private void bAceptar\_Click(object sender, EventArgs e)

{

if (dgvProductos.RowCount > 1)

{

int i;

RegistraVenta();

actualizaExistencias();

for (i = 0; i < dgvProductos.RowCount - 1; i++)

RegistraDetalleVenta(i);

Mensaje();

}

else

MessageBox.Show("No hay Productos para Registrar la Venta ", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

/\*=======================================================================

Este metodo registra la venta y actualiza las existencias

en la base de datos

\*=======================================================================\*/

private void RegistraVenta()

{

try

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into Ventas(Folio, Fecha, DetalleVenta, Total)values (@iventa, @fventa, @dventa, @tventa)";

OleDbCommand cmd = new OleDbCommand(insert, cnn);

cmd.Parameters.AddWithValue("@iventa", tbFolio.Text);

cmd.Parameters.AddWithValue("@fventa", DateTime.Today.ToString("d") + " " + DateTime.Now.ToLongTimeString());

cmd.Parameters.AddWithValue("@dventa", tbFolio.Text);

cmd.Parameters.AddWithValue("@tventa", tbTotal.Text);

cnn.Open();

int exe = cmd.ExecuteNonQuery();

cnn.Close();

}

catch (OleDbException)

{

MessageBox.Show("Ocurrió un error al insertar: ", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

/\*=======================================================================

Este metod actualiza las existencias en la base de datos

\*=======================================================================\*/

private void actualizaExistencias()

{

int i;

List<CExistencias> lEAux = new List<CExistencias>();

List<CExistencias> lE = new List<CExistencias>();

for (i = 0; i < dgvCantidad.RowCount; i++ )

{

leerRegistroExistencias(dgvProductos[0, i].Value.ToString());

using (OleDbConnection cn = new OleDbConnection(conexion))

{

string query = "UPDATE Existencias SET [Cantidad] = @cantidad WHERE IdProducto=@Id";

OleDbCommand cmd = new OleDbCommand(query, cn);

cmd.Parameters.AddWithValue("@cantidad", producto.cantidad - Convert.ToInt32(dgvCantidad[0, i].Value.ToString()));

cmd.Parameters.AddWithValue("@Id", dgvProductos[0, i].Value.ToString());

cn.Open();

cmd.ExecuteNonQuery();

cn.Close();

}

}

}

/\*=======================================================================

Este metodo registra el Detalle de Venta en la base de datos

\*=======================================================================\*/

private void RegistraDetalleVenta(int i)

{

try

{

OleDbConnection cnn = new OleDbConnection(conexion);

const string insert = @"insert into DetalleVenta(Folio, idProducto, Cantidad, PrecioUnitario, Subtotal)values (@fo, @ip, @ca, @pu, @st)";

OleDbCommand cmd = new OleDbCommand(insert, cnn);

cmd.Parameters.AddWithValue("@fo", tbFolio.Text);

cmd.Parameters.AddWithValue("@ip", dgvProductos[0, i].Value.ToString());

cmd.Parameters.AddWithValue("@ca", Convert.ToInt32(dgvCantidad[0, i].Value.ToString()));

cmd.Parameters.AddWithValue("@pu", Convert.ToSingle(dgvDescripcion[1, i].Value.ToString().Substring(1)));

cmd.Parameters.AddWithValue("@st", Convert.ToSingle(dgvSubtotales[0, i].Value.ToString().Substring(1)));

cnn.Open();

int exe = cmd.ExecuteNonQuery();

cnn.Close();

}

catch (OleDbException)

{

MessageBox.Show("Ocurrió un error al insertar: ", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void bVPendiente\_Click(object sender, EventArgs e)

{

FormVenta nfv = new FormVenta();

if (lfolios[lfolios.Count - 1] + 1 == Convert.ToInt32(tbFolio.Text))

nfv.tbFolio.Text = (Convert.ToInt32(tbFolio.Text) + 1).ToString();

else

nfv.tbFolio.Text = Convert.ToString(lfolios[lfolios.Count - 1] + 1);

nfv.bVPendiente.Visible = false;

nfv.bImpPedido.Visible = false;

nfv.ShowDialog();

}

private void ponerFolio()

{

lfolios.Sort();

if (venta == null)

tbFolio.Text = "1";

else

{

if (lfolios[lfolios.Count - 1] == lfolios.Count)

tbFolio.Text = Convert.ToString(lfolios[lfolios.Count - 1] + 1);

else

{

int cont = 1;

foreach (int i in lfolios)

{

if (i != cont)

{

tbFolio.Text = Convert.ToString(cont.ToString());

break;

}

cont++;

}

}

}

}

/\*=======================================================================

Este evento se ejecuta cuando se termina de editar un celda

en el datagrid Cantidad

\*=======================================================================\*/

private void dgvCantidad\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

int i;

if (dgvCantidad.RowCount > 0)

{

if (dgvCantidad[0, e.RowIndex].Value == null)

dgvCantidad[0, e.RowIndex].Value = 0;

string s = dgvCantidad[0, e.RowIndex].Value.ToString();

for (i = 0; i < s.Length; i++)

{

if (Char.IsNumber(s.ElementAt(i)) == false)

{

MessageBox.Show("La Celda\nSolo Puede Contener Valores Numericos Enteros", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

dgvCantidad[0, e.RowIndex].Value = "1";

break;

}

}

//verificaExistencias(e.RowIndex);

}

}

/\*=======================================================================

Este metodo se encarga de verificar las existencias

\*=======================================================================\*/

private void verificaExistencias(int indCantidad)

{

int i = dgvProductos.CurrentRow.Index;

string idP = dgvProductos[0, indCantidad].Value.ToString();

string cantidad = dgvCantidad[0, indCantidad].Value.ToString();

leerRegistroExistencias(idP);

if (Convert.ToInt32(cantidad) > producto.cantidad)

{

MessageBox.Show("No hay Existencias Suficientes\nSe Modificara la Cantidad", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

dgvCantidad[0, indCantidad].Value = producto.cantidad;

}

}

/\*=======================================================================

Este metodo se encarga de eliminar el producto seleccionado

de la lista

\*=======================================================================\*/

private void bEliminarProducto\_Click(object sender, EventArgs e)

{

if (dgvProductos.CurrentRow.Index < dgvProductos.RowCount-1)

{

dgvDescripcion.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvCantidad.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvSubtotales.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvProductos.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvProductos.Focus();

dgvProductos.CurrentCell = dgvProductos[0, dgvProductos.RowCount-1];

}

else

MessageBox.Show("Seleccione un Producto", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

private void tbTelefono\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == Convert.ToChar(Keys.Enter))

bBuscarCli\_Click(null, null);

}

/\*=======================================================================

Este metod imprime la lista de productos

\*=======================================================================\*/

private void bImpPedido\_Click(object sender, EventArgs e)

{

PrintPreviewDialog vistaPrevia = new PrintPreviewDialog();

PrintDocument pd = new PrintDocument();

crearArchivo();

try

{

streamParaImp = new StreamReader("C:\\temporalSW\\ejemplo.txt");

try

{

Fuente = new Font("Lucida Console", 10);

pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(this.pd\_PrintPage);

vistaPrevia.Document = pd;

Form f = vistaPrevia as Form;

Control[] ts = vistaPrevia.Controls.Find("toolStrip1", true);

ToolStrip to = ts[0] as ToolStrip;

to.Items.RemoveAt(0);

f.WindowState = FormWindowState.Maximized;

f.ShowDialog();

streamParaImp.Close();

streamParaImp = new StreamReader("C:\\temporalSW\\ejemplo.txt");

pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(this.pd\_PrintPage);

pd.Print();

}

finally

{

streamParaImp.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("No se Puede Imprimir\n" + ex.Message, "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

File.Delete("C:\\temporalSW\\ejemplo.txt");

}

}

private void crearArchivo()

{

int x = 0;

//int ins = dgvProveedores.CurrentRow.Index;

try

{

StreamWriter sw = new StreamWriter("C:\\temporalSW\\ejemplo.txt", true, Encoding.ASCII);

sw.WriteLine(" Pedido Del Cliente");

sw.WriteLine("");

sw.WriteLine("Id: " + tbTelefono.Text);

sw.WriteLine("Cliente: " + tbCliente.Text);

sw.WriteLine("Tel: " + tbTelefono.Text);

sw.WriteLine("");

sw.WriteLine("Fecha: " + DateTime.Now.ToShortDateString() + " " + DateTime.Now.ToShortTimeString());

sw.WriteLine("");

sw.WriteLine("");

sw.WriteLine("========================================================================");

sw.WriteLine("[Producto] " + " [Cantidad]");

sw.WriteLine("========================================================================");

for (x = 0; x < dgvCantidad.RowCount; x++ )

{

sw.WriteLine(dgvDescripcion[0, x].Value.ToString() + " " + " " + calculaEspacio(dgvDescripcion[0, x].Value.ToString(), 40) + " " + " " +

dgvCantidad[0, x].Value.ToString());

}

sw.Close();

}

catch (Exception e)

{

MessageBox.Show("No se Puede Imprimir\n" + e.Message, "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private string calculaEspacio(string c, int t)

{

int x;

int s = c.Length;

int es = t - s;

string cad = "";

for (x = 0; x < es; x++)

cad += "-";

return cad;

}

private void pd\_PrintPage(object sender, PrintPageEventArgs ev)

{

float lineasPorPagina = 0;

float yPos = 0;

int count = 0;

float margenIzquierda = ev.MarginBounds.Left;

float margenArriba = ev.MarginBounds.Top;

string linea = null;

lineasPorPagina = ev.MarginBounds.Height / Fuente.GetHeight(ev.Graphics);

while (count < lineasPorPagina && ((linea = streamParaImp.ReadLine()) != null))

{

yPos = margenArriba + (count \* Fuente.GetHeight(ev.Graphics));

ev.Graphics.DrawString(linea, Fuente, Brushes.Black, margenIzquierda, yPos, new StringFormat());

count++;

}

if (linea != null)

ev.HasMorePages = true;

else

ev.HasMorePages = false;

}

private void bImpRegVenta\_Click(object sender, EventArgs e)

{

PrintPreviewDialog vistaPrevia = new PrintPreviewDialog();

PrintDocument pd = new PrintDocument();

crearArchivoVenta();

try

{

streamParaImp = new StreamReader("C:\\temporalSW\\ejemplo.txt");

try

{

Fuente = new Font("Lucida Console", 9);

pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(this.pd\_PrintPage);

vistaPrevia.Document = pd;

Form f = vistaPrevia as Form;

Control[] ts = vistaPrevia.Controls.Find("toolStrip1", true);

ToolStrip to = ts[0] as ToolStrip;

to.Items.RemoveAt(0);

f.WindowState = FormWindowState.Maximized;

f.ShowDialog();

streamParaImp.Close();

streamParaImp = new StreamReader("C:\\temporalSW\\ejemplo.txt");

pd = new PrintDocument();

pd.PrintPage += new PrintPageEventHandler(this.pd\_PrintPage);

pd.Print();

}

finally

{

streamParaImp.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("No se Puede Imprimir\n" + ex.Message, "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

File.Delete("C:\\temporalSW\\ejemplo.txt");

}

bAceptar\_Click(null, null);

}

private void crearArchivoVenta()

{

int x = 0;

try

{

StreamWriter sw = new StreamWriter("C:\\temporalSW\\ejemplo.txt", true, Encoding.ASCII);

sw.WriteLine("Folio: "+tbFolio.Text);

sw.WriteLine("Id: " + tbTelefono.Text);

sw.WriteLine("Cliente: " + tbCliente.Text);

sw.WriteLine("Telefono: " + tbTelefono.Text);

sw.WriteLine("Direccion: " + tbDireccion.Text);

sw.WriteLine("");

sw.WriteLine("Fecha: " + DateTime.Now.ToShortDateString() + " " + DateTime.Now.ToShortTimeString());

sw.WriteLine("");

sw.WriteLine("");

sw.WriteLine("==================================================================================");

sw.WriteLine(" [Id] [Producto] [P. Unitario] [Cantidad] [Subtotal]");

sw.WriteLine("==================================================================================");

for (x = 0; x < dgvCantidad.RowCount; x++)

{

sw.WriteLine(dgvProductos[0, x].Value.ToString() + " " + " " + calculaEspacio(dgvProductos[0, x].Value.ToString(), 10) + " " + " " +

dgvDescripcion[0, x].Value.ToString() + " " + " " + calculaEspacio(dgvDescripcion[0, x].Value.ToString(), 30) + " " + " " +

dgvDescripcion[1, x].Value.ToString() + " " + " " + calculaEspacio(dgvDescripcion[1, x].Value.ToString(), 10) + " " + " " +

dgvCantidad[0, x].Value.ToString() + " " + " " + calculaEspacio(dgvCantidad[0, x].Value.ToString(), 10) + " " + " " +

dgvSubtotales[0, x].Value.ToString());

}

sw.WriteLine("");

sw.WriteLine(" TOTAL = "+tbTotal.Text);

sw.Close();

}

catch (Exception e)

{

MessageBox.Show("No se Puede Imprimir\n" + e.Message, "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dgvDescripcion\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex > -1 && dgvDescripcion[1, e.RowIndex].Value != null)

{

string s = dgvDescripcion[1, e.RowIndex].Value.ToString();

char[] separador = { '.' };

string[] valores = s.Split(separador);

if (valores.LongLength == 2 && valores[1].Length > 0)

{

dgvDescripcion[1, e.RowIndex].Value = valores[0] + "." + valores[1].Substring(0, 1) + "0";

}

else

{

dgvDescripcion[1, e.RowIndex].Value = valores[0] + "." + "00";

}

}

}

private void tbTotal\_TextChanged(object sender, EventArgs e)

{

string s = tbTotal.Text;

char[] separador = { '.' };

string[] valores = s.Split(separador);

if (valores.LongLength == 2 && valores[1].Length > 0)

{

tbTotal.Text = valores[0] + "." + valores[1].Substring(0, 1) + "0";

}

else

{

tbTotal.Text = valores[0] + "." + "00";

}

}

private void dgvProductos\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if(dgvProductos[0, e.RowIndex].Value == null)

if (dgvProductos.CurrentRow.Index < dgvProductos.RowCount - 1)

{

if (dgvProductos.CurrentRow.Index < dgvDescripcion.RowCount)

{

dgvDescripcion.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvCantidad.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvSubtotales.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

}

dgvProductos.Rows.RemoveAt(dgvProductos.CurrentRow.Index);

dgvProductos.Focus();

}

}

public void Mensaje()

{

System.Threading.Thread t = new System.Threading.Thread(new System.Threading.ThreadStart(SplashScreen));

t.Start();

System.Threading.Thread.Sleep(1000);

t.Abort();

}

public void SplashScreen()

{

Application.Run(new FormMensaje());

}

}