**Manual técnico Base de Datos “Proyecto”**

**Versiones de código utilizadas:**

Versión de IDE y código C#:Visual Studio 2022, versión **17.6**.

Versión de motor de base de datos SQL Server: SQL Server 2022, versión **16.00**.

**Requerimientos .NET:**

1. Procesador: Se recomienda un procesador con múltiples núcleos y una velocidad de reloj alta. Un procesador de gama media o alta de Intel Core i5 o i7, o su equivalente en AMD, sería una buena opción.
2. Memoria RAM: Se recomienda al menos 8 GB de RAM para un rendimiento óptimo. Sin embargo, si planeas trabajar en proyectos grandes o utilizar varias instancias de Visual Studio al mismo tiempo, es posible que desees considerar 16 GB o más.
3. Almacenamiento: Un disco duro de estado sólido (SSD) es altamente recomendado. Proporciona una mayor velocidad de lectura/escritura en comparación con los discos duros tradicionales, lo que mejora el rendimiento general del sistema y acelera los tiempos de carga de Visual Studio.
4. Tarjeta gráfica: Aunque no es un requisito estricto para Visual Studio, si también planeas utilizar Visual Studio para el desarrollo de aplicaciones gráficas intensivas o juegos, te recomendamos una tarjeta gráfica dedicada de gama media o alta. Esto ayudará en el rendimiento general del sistema y en la aceleración de tareas gráficas.
5. Sistema operativo: Visual Studio es compatible con Windows 10 y versiones posteriores. Asegúrate de tener instalada la versión más reciente de Windows y de mantenerla actualizada.
6. Resolución de pantalla: Se recomienda una resolución de pantalla de al menos 1920x1080 (Full HD) para aprovechar al máximo la interfaz de usuario y la visualización de código en Visual Studio.
7. Conectividad: Asegúrate de tener puertos USB y una conexión a Internet estable para descargar actualizaciones y acceder a recursos en línea.
8. Otros: Además de los requisitos de hardware, es importante asegurarse de tener instalada la versión más reciente de Visual Studio y de mantenerla actualizada. También se recomienda disponer de un buen antivirus y realizar copias de seguridad periódicas de tus proyectos para evitar la pérdida de datos.

Recuerda que estos son requisitos recomendados y pueden variar dependiendo de tus necesidades específicas y del tamaño y complejidad de tus proyectos en Visual Studio. Siempre es bueno consultar la documentación oficial de Microsoft para obtener información actualizada sobre los requisitos del sistema para Visual Studio.

**Requerimientos SQL Server:**

1. Procesador: Se recomienda un procesador de múltiples núcleos y alta velocidad para manejar eficientemente las consultas y cargas de trabajo de SQL Server. Un procesador Intel Core i5 o i7, o su equivalente en AMD, sería una buena opción. Si planeas trabajar con bases de datos muy grandes o realizar operaciones intensivas, es posible que desees considerar un procesador de gama alta.
2. Memoria RAM: SQL Server es intensivo en memoria y se recomienda tener al menos 8 GB de RAM. Sin embargo, para bases de datos más grandes o para un mejor rendimiento, es recomendable tener 16 GB o más de RAM.
3. Almacenamiento: SQL Server se beneficia de un almacenamiento rápido y confiable. Se recomienda utilizar discos duros de estado sólido (SSD) para el almacenamiento de bases de datos. También es útil tener un disco separado para los archivos de registro de transacciones. Considera utilizar RAID para una mayor protección de datos y rendimiento.
4. Sistema operativo: SQL Server es compatible con varias versiones de Windows Server, así como con algunas versiones de Windows 10. Verifica la documentación oficial de Microsoft para conocer las versiones compatibles.
5. Espacio en disco: Asegúrate de tener suficiente espacio en disco para almacenar tus bases de datos y archivos de registro, así como para las copias de seguridad y los archivos de registro de errores. El espacio requerido variará según el tamaño de tus bases de datos y las necesidades específicas de tu aplicación.
6. Conectividad: Asegúrate de tener una conexión de red estable para acceder a las bases de datos de SQL Server y para permitir conexiones remotas si es necesario. También debes asegurarte de que los puertos necesarios para SQL Server estén abiertos en tu firewall.
7. Seguridad: Considera implementar medidas de seguridad adecuadas, como contraseñas seguras y autenticación fuerte, para proteger tus bases de datos y prevenir accesos no autorizados.

Estos son los requisitos básicos recomendados para ejecutar SQL Server. Ten en cuenta que los requisitos reales pueden variar dependiendo del tamaño y la complejidad de tus bases de datos, así como de las necesidades específicas de tu aplicación. Consulta la documentación oficial de Microsoft para obtener información actualizada y detallada sobre los requisitos del sistema para SQL Server.

**Código fuente con anexo de Forms .NET:**

Explorador de soluciones:En el presente se puede visualizar todos los Windows form utilizados para la creación de la parte gráfica del sistema. Así también todas las conexiones creadas con la base de datos para guardar la información y traerla a la vista del usuario.

**Texto

Descripción generada automáticamente**

Conexión a base de datos: Por medio de las presentes líneas de código establecimos la conexión a la base de datos ubicada en SQLEXPRESS edition.

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="ProyectoBDD.Properties.Settings.storeConnectionString"

connectionString="Data Source=LUIS\_HERRERA\SQLEXPRESS;Initial Catalog=store;Integrated Security=True"

providerName="System.Data.SqlClient" />

<add name="ProyectoBDD.Properties.Settings.ProyectoConnectionString"

connectionString="Data Source=LUIS\_HERRERA\SQLEXPRESS;Initial Catalog=Proyecto;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

Form1: Con el presente Windows form se creo una medida de seguridad para el sistema para que el usuario cuente con un “Usuario” y “Contraseña” los cuales han sido creados desde el código fuente el cual validará si son verídicas o no.

Cuenta además con un botón llamado “Ingresar” para validar por medio de condicionales de manera directa por medio de datos quedamos.

Cuenta con dos textbox, uno de ellos nos permite escribir el usuario el cual es “Ingeniero123” y el segundo para escritura de contraseña “Umg123”.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProyectoBDD

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string usuario = txtUsuario.Text;

string contrasena = txtContraseña.Text;

// Verificar si el usuario y la contraseña son válidos

if (usuario == "Ingeniero123" && contrasena == "Umg123")

{

// Si el inicio de sesión es exitoso, mostrar un mensaje y cerrar el formulario de inicio de sesión

MessageBox.Show(" ¡Bienvenido Ingeniero!");

FormMenú formMenú = new FormMenú();

formMenú.Show();

this.Hide();

}

else

{

// Si el inicio de sesión falla, mostrar un mensaje de error y borrar los cuadros de texto de la contraseña

MessageBox.Show(" El nombre de usuario y/o la contraseña son incorrectos.");

txtContraseña.Clear();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

Interfaz de usuario gráfica

Descripción generada automáticamente

Form comprar: Este es uno de los forms más importantes, puesto que gracias al presente podremos realizar todas las compras necesarias y poder poblar la totalidad de las tablas de base de datos. Fue creada a base de textbox los cuales guardarán información en una tabla llamada “Compras” en la base de datos maestra.

Cuenta además con un DATAGRIDVIEW el cual mostrará de manera gráfica todos los cambios realizados y los campos afectados dentro de la base de datos.

Cuenta con un botón programado el cual es el encargado de devolver a un menú en el que el usuario podrá moverse de un form a otro.

Por medio de un .Hide creamos la limpieza del form anterior al seleccionado por medio de botones programados.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Drawing;

using System.Drawing.Printing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace ProyectoBDD

{

public partial class FormComprar : Form

{

public FormComprar()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'proyectoDataSet6.Compras' table. You can move, or remove it, as needed.

this.comprasTableAdapter.Fill(this.proyectoDataSet6.Compras);

}

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

{

FormMenú formMenú = new FormMenú();

formMenú.Show();

this.Hide();

}

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

{

SqlConnection con = new SqlConnection("Data Source=LUIS\_HERRERA\\SQLEXPRESS;Initial Catalog=Proyecto;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Compras VALUES (@id\_producto,@cantidad,@NIT,@nombre,@direccion,@telefono, @correo, @portal)", con);

cmd.Parameters.AddWithValue("@id\_producto", int.Parse(txtidproductoc.Text));

txtidproductoc.Clear();

cmd.Parameters.AddWithValue("@cantidad", int.Parse(txtcantidadc.Text));

txtcantidadc.Clear();

cmd.Parameters.AddWithValue("@NIT", int.Parse(txtnitc.Text));

txtnitc.Clear();

cmd.Parameters.AddWithValue("@nombre", comboboxcompras.Text);

comboboxcompras.Items.Clear();

cmd.Parameters.AddWithValue("@direccion", txtdirecciónc.Text);

txtdirecciónc.Clear();

cmd.Parameters.AddWithValue("@telefono", txttelefonoc.Text);

txttelefonoc.Clear();

cmd.Parameters.AddWithValue("@correo", txtcorreoc.Text);

txtcorreoc.Clear();

cmd.Parameters.AddWithValue("@portal", comboboxcompras.Text);

comboboxcompras.Items.Clear();

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Datos guardados con éxito");

}

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

{

SqlConnection con = new SqlConnection("Data Source=LUIS\_HERRERA\\SQLEXPRESS;Initial Catalog=Super;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("SELECT \* FROM Compras", con);

con.Close();

}

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

{

try

{

// Obtener el texto del TextBox

string contenido = txtidproductoc.Text;

string contenido1 = txtcantidadc.Text;

string contenido2 = txtnitc.Text;

string contenido3 = comboboxcompras.Text;

string contenido4 = txtdirecciónc.Text;

string contenido5 = txttelefonoc.Text;

string contenido6 = txtcorreoc.Text;

string contenido7 = comboBoxportal.Text;

// Guardar el contenido en un archivo de texto temporal

string tempFilePath = Path.GetTempFileName();

File.WriteAllText(tempFilePath, contenido);

// Configurar el proceso para imprimir con PDFCreator

ProcessStartInfo psi = new ProcessStartInfo

{

FileName = "PDFCreator.exe",

Arguments = String.Format("/PrintFile=\"{0}\"", tempFilePath),

WindowStyle = ProcessWindowStyle.Hidden

};

// Iniciar el proceso de impresión

Process.Start(psi);

// Eliminar el archivo temporal

File.Delete(tempFilePath);

}

catch (Exception ex)

{

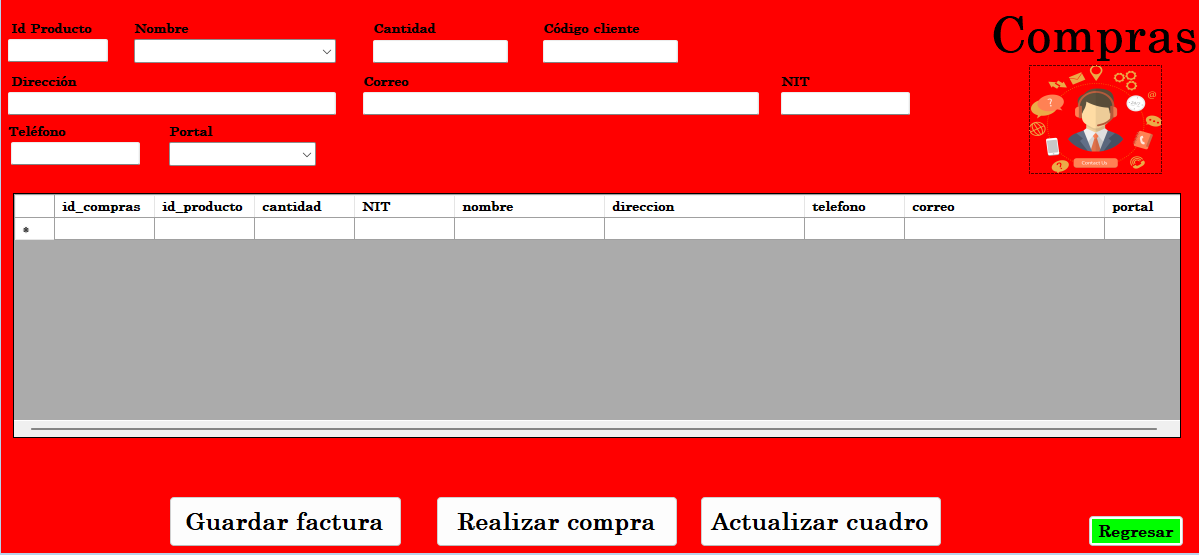
MessageBox.Show("Error al imprimir con PDFCreator: " + ex.Message);

}

}

}

}



Form Menú: Un menú de opciones es importante dentro de cualquier sistema a nivel mundial, puesto que nos proporciona un ambiente amigable para poder movernos según nuestras necesidades como usuarios. Cuenta con tres botones los cuales nos redirigen a los otros form explicados con su funcionalidad.

Por medio de un .Hide creamos la limpieza del form anterior al seleccionado por medio de botones programados.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProyectoBDD

{

public partial class FormMenú : Form

{

public FormMenú()

{

InitializeComponent();

}

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

{

FormResumen formInventario = new FormResumen();

formInventario.Show();

this.Hide();

}

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

{

FormVender formVender = new FormVender();

formVender.Show();

this.Hide();

}

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

{

FormComprar formComprar = new FormComprar();

formComprar.Show();

this.Hide();

}

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

{

}

}

}

Texto

Descripción generada automáticamente

Form Resumen: El área de reportería y control es crucial en cualquier empresa o sistema que se encargue del diseño y desarrollo de soluciones de tecnología. Por tal motivo gracias a cuatro data grid view hemos creado un apartado de control en el cual podremos observar desde el disponible en el form “Inventario” o stock, así como el nombre de todos los proveedores que han surtido nuestro negocio, y los dos últimos siendo uno de ellos de “Compras” y “Ventas” por medio de los cuales contaremos con un control de la cantidad de cada una así como los campos para la verificación de la información.

Cuenta con un botón programado el cual es el encargado de devolver a un menú en el que el usuario podrá moverse de un form a otro.

Por medio de un .Hide creamos la limpieza del form anterior al seleccionado por medio de botones programados.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProyectoBDD

{

public partial class FormResumen : Form

{

public FormResumen()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'proyectoDataSet10.Compras' table. You can move, or remove it, as needed.

this.comprasTableAdapter1.Fill(this.proyectoDataSet10.Compras);

// TODO: This line of code loads data into the 'proyectoDataSet9.Ventas' table. You can move, or remove it, as needed.

this.ventasTableAdapter1.Fill(this.proyectoDataSet9.Ventas);

// TODO: This line of code loads data into the 'proyectoDataSet8.proveedores' table. You can move, or remove it, as needed.

this.proveedoresTableAdapter1.Fill(this.proyectoDataSet8.proveedores);

// TODO: This line of code loads data into the 'proyectoDataSet7.productos' table. You can move, or remove it, as needed.

this.productosTableAdapter1.Fill(this.proyectoDataSet7.productos);

}

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

{

FormMenú formMenú = new FormMenú();

formMenú.Show();

this.Hide();

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void dataGridView4\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

Imagen que contiene Calendario

Descripción generada automáticamente

Form Ventas: Este es uno de los forms más importantes, puesto que gracias al presente podremos realizar todas las ventas necesarias y poder interactuar y afectar a nuestra base de datos. Fue creada a base de textbox los cuales guardarán información en una tabla llamada “Ventas” en la base de datos maestra.

Cuenta además con un DATAGRIDVIEW el cual mostrará de manera gráfica todos los cambios realizados y los campos afectados dentro de la base de datos.

Cuenta con un botón programado el cual es el encargado de devolver a un menú en el que el usuario podrá moverse de un form a otro.

Por medio de un .Hide creamos la limpieza del form anterior al seleccionado por medio de botones programados.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Drawing;

using System.Drawing.Printing;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace ProyectoBDD

{

public partial class FormVender : Form

{

public FormVender()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'proyectoDataSet5.Ventas' table. You can move, or remove it, as needed.

this.ventasTableAdapter.Fill(this.proyectoDataSet5.Ventas);

// TODO: This line of code loads data into the 'proyectoDataSet4.productos' table. You can move, or remove it, as needed.

this.productosTableAdapter.Fill(this.proyectoDataSet4.productos);

}

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

{

SqlConnection con = new SqlConnection("Data Source=LUIS\_HERRERA\\SQLEXPRESS;Initial Catalog=Proyecto;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Ventas VALUES (@id\_producto,@cantidad,@codigo\_cliente,@nombre,@direccion,@correo, @nit)", con);

cmd.Parameters.AddWithValue("@id\_producto", int.Parse(txtidproductov.Text));

txtidproductov.Clear();

cmd.Parameters.AddWithValue("@cantidad", int.Parse(txtcantidadv.Text));

txtcantidadv.Clear();

cmd.Parameters.AddWithValue("@codigo\_cliente", int.Parse(txtcodigoclientev.Text));

txtcodigoclientev.Clear();

cmd.Parameters.AddWithValue("@nombre", comboboxventas.Text);

comboboxventas.Items.Clear();

cmd.Parameters.AddWithValue("@direccion", txtdirecciónv.Text);

txtdirecciónv.Clear();

cmd.Parameters.AddWithValue("@correo", txtcorreov.Text);

txtcorreov.Clear();

cmd.Parameters.AddWithValue("@nit", int.Parse(txtnitv.Text));

txtnitv.Clear();

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Datos guardados con éxito");

}

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

{

FormMenú formMenú = new FormMenú();

formMenú.Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection("Data Source=LUIS\_HERRERA\\SQLEXPRESS;Initial Catalog=Super;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("SELECT \* FROM Ventas", con);

con.Close();

}

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

{

try

{

// Obtener el texto del TextBox

string contenido = txtidproductov.Text;

string contenido1 = txtcantidadv.Text;

string contenido2 = txtcodigoclientev.Text;

string contenido3 = comboboxventas.Text;

string contenido4 = txtdirecciónv.Text;

string contenido5 = txtcorreov.Text;

string contenido6 = txtnitv.Text;

// Configurar la impresora y el documento

PrintDocument printDocument = new PrintDocument();

printDocument.PrintPage += (s, ev) =>

{

// Definir el formato y el tamaño del documento

Font font = new Font("Arial", 12);

RectangleF bounds = ev.MarginBounds;

// Imprimir el contenido en la página

ev.Graphics.DrawString(contenido, font, Brushes.Black, bounds, StringFormat.GenericTypographic);

};

// Imprimir directamente a la impresora predeterminada

printDocument.Print();

}

catch (Exception ex)

{

MessageBox.Show("Error al imprimir: " + ex.Message);

}

}

}

}

Imagen que contiene Diagrama

Descripción generada automáticamente

**Script Base de Datos Proyecto:**

Explicación de la función de la base de datos: La importancia para la conexión con la solución en .Net es crucial, ya que este va a ser el almacén de nuestro sistemas con el cual poblaremos nuestra base de datos desde cero.

Por otro lado vamos a contar con tablas que serán utilizadas con diferente propósito.

CREATE DATABASE Proyecto;

USE Proyecto;

CREATE TABLE Ventas (

id\_venta INT IDENTITY(1,1) PRIMARY KEY,

id\_producto INT,

cantidad INT,

codigo\_cliente int,

nombre varchar(80),

direccion varchar(120),

correo varchar(150),

nit int

);

CREATE TABLE Compras (

id\_compras INT IDENTITY(1,1) PRIMARY KEY,

id\_producto INT,

cantidad INT,

NIT int,

nombre varchar(100),

direccion varchar(100),

telefono int,

correo varchar(120),

portal varchar(50)

);

CREATE TABLE productos (

id\_producto INT NOT NULL PRIMARY KEY,

descripción VARCHAR(80),

id\_categoria INT NOT NULL,

precio INT NOT NULL,

disponible INT NOT NULL,

detalles VARCHAR(80)

);

INSERT INTO productos(id\_producto, descripción, id\_categoria, precio, disponible, detalles) VALUES

(7, 'Lapicero Bic Rosado', 1, 3, 8,'Lapicero especial para escritura en carta'),

(8, 'Lapicero BIC azul', 1, 2, 9,'Lapicero especial para escritura en matemática'),

(9, 'Lapicero BIC Rojo', 1, 3, 11,'Lapicero especial para escritura en caligrafía'),

(10, 'Bolson Totto Mx-352', 3, 350, 15, 'Especial para traslado de computadoras');

CREATE TABLE categorías (

id INT PRIMARY KEY,

descripcion VARCHAR(45) NOT NULL

);

INSERT INTO categorías(id, descripcion) VALUES

(4, 'Marcador permanente'),

(3, 'Lapiceros'),

(1, 'libros'),

(2, 'Lapices'),

(5, 'Crayones'),

(6, 'Bolsones');

CREATE TABLE customer (codigo\_cliente int IDENTITY(1,1) not null primary key,

nombre varchar(80) not null,

direccion varchar(120) not null,

correo varchar(150) not null,

nit int not null);

INSERT INTO customer VALUES ('José Carlos de León Juárez', 'Aldea Estanzuelas', 'jose02@gmail.com', 19558889);

CREATE TABLE proveedores (NIT int primary key not null,

nombre varchar(100) not null,

direccion varchar(100) not null,

telefono int not null,

correo varchar(120) not null,

portal varchar(50) not null);

INSERT INTO proveedores VALUES (105192933, 'Luis Felipe Herrera Donis', 'Aldea La Casita', 49888936, 'luisherrera.lh925@gmail.com', 'Físico');

CREATE TABLE factura (numero int identity (1,1),

NIT int not null,

serie varchar(10) not null,

nombre varchar(80) not null);

INSERT INTO factura VALUES (105991929, 'A-2', 'La Republicana'),

(19557899, 'A-3', 'La Democrática'),

(20788899, 'A-7', 'El Porvenir'),

(48778997, 'A-4', 'Caballeros S.A.');

Triggers: La base de datos interactúa entre sí por medio de dos disparadores, los cuales invocan tablas específicas, seleccionando los campos a afectar, la lógica será que disponible va a ser igual a disponible restado la cantidad de artículos comprados o vendidos.

CREATE TRIGGER realizar\_ventas

ON Ventas FOR INSERT AS

BEGIN

DECLARE @producto AS INT

DECLARE @cantidad AS INT

SET @producto = (SELECT id\_producto FROM inserted)

SET @cantidad = (SELECT cantidad FROM inserted)

UPDATE productos

SET disponible = disponible - @cantidad

WHERE id\_producto LIKE @producto

END

CREATE TRIGGER realizar\_compras

ON Compras FOR INSERT AS

BEGIN

DECLARE @producto AS INT

DECLARE @cantidad AS INT

SET @producto = (SELECT id\_producto FROM inserted)

SET @cantidad = (SELECT cantidad FROM inserted)

UPDATE productos

SET disponible = disponible + @cantidad

WHERE id\_producto = @producto

END