Universidad Tecnológica De Panamá
Facultad de Ingeniería en Sistemas Computacionales
2do Semestre

Profesor: Ronald Ponce

Integrantes: Jean Meléndez 8-985-955

Kevin Valdés 8-1021-301

Daniel Gonzales 8-1022-1099

Martin Liao 1-757-1706

Ricardo Rose

Asignatura: Base de datos

Semestral: Trabajo Escrito Proyecto Final BD

Año Lectivo: 2024

INTRODUCCIÓN

El desarrollo e implementación de sistemas de bases de datos es un pilar fundamental en la gestión eficiente de información en el ámbito tecnológico. Este trabajo, correspondiente al proyecto final de la materia “Base de Datos 1”, tiene como objetivo aplicar los conocimientos adquiridos durante el semestre para diseñar y desarrollar una base de datos transaccional junto con un sistema funcional que permita la administración y control de datos en un entorno realista.

El proyecto, denominado "Proyecto_Final_BD", está enfocado en gestionar una librería ficticia que incluye la venta y compra de libros. Para ello, se diseñaron e implementaron tablas relacionadas que representan usuarios, libros, métodos de pago, pedidos y otros elementos esenciales. Además, se creó un sistema de autenticación que distingue entre dos tipos de roles: administrador y cliente, cada uno con funcionalidades específicas accesibles mediante menús personalizados.

El desarrollo del sistema incluye funcionalidades clave, tales como la gestión de usuarios, la ejecución de compras mediante procedimientos almacenados, y la visualización de datos en tiempo real a través de una interfaz gráfica diseñada en Visual Basic .NET. Este proyecto busca no solo demostrar competencias técnicas en diseño y manipulación de bases de datos, sino también la integración práctica de estas con herramientas de desarrollo de software.

Con este trabajo se pretende reforzar conceptos fundamentales de normalización, transacciones SQL, procedimientos almacenados, y conexiones a bases de datos, reflejando la importancia de los sistemas de información en el ámbito profesional y académico.

Descripción General del Proyecto

El proyecto semestral titulado "Sistema de Gestión de Librería", desarrollado como parte de la asignatura Base de Datos 1, integra una base de datos en SQL Server con una aplicación gráfica desarrollada en Visual Studio utilizando Visual Basic .NET. Este sistema busca proporcionar una solución integral para la gestión de una librería, facilitando tanto las operaciones administrativas como las interacciones de los clientes con el sistema.

La base de datos, denominada Proyecto_Final_BD, está diseñada siguiendo principios de normalización, consistencia referencial y transaccionalidad, mientras que la interfaz gráfica divide las funcionalidades entre administradores y clientes, asegurando un control diferenciado según los roles.

Procedimientos Almacenados:

ComprarLibroCliente2: Maneja la compra de libros con validaciones de stock, registro de detalles del pedido y asignación automática de estados de pedidos según el método de pago.

CancelarCompra: Permite cancelar compras en estado "Pendiente", asegurando la reversión del stock.

ComprarLibroProveedor: Automatiza la compra de libros a proveedores, incluyendo el cálculo de costos totales.

Interfaz Gráfica:

Form1 (Inicio):

Actúa como punto de entrada para la conexión con la base de datos, asegurando que el sistema no avance sin una conexión exitosa.

FrmLogin (Autenticación):

Verifica credenciales de SQL Server y valida roles (1 para administrador, 2 para cliente) para redirigir al menú correspondiente.

FrmMenu (Administradores):

Ofrece herramientas avanzadas para:

Gestión de usuarios y libros.

Visualización y control de proveedores.

Análisis de pedidos y compras.

Eliminación de registros.

FrmMenuCliente (Clientes):

Focalizado en la experiencia del cliente, con opciones para:

Comprar libros, consultar métodos de pago y categorías.

Actualizar su información personal.

Cancelar pedidos pendientes y recibir facturas detalladas en tiempo real.

Diseño de la Interfaz:

Centrado en la usabilidad y accesibilidad, con:

Animaciones interactivas en botones.

DataGridViews dinámicos para consultas personalizadas.

Mensajes emergentes que guían al usuario durante las operaciones.

Seguridad e Integridad:

Validación exhaustiva de entradas del usuario.

Manejo robusto de errores mediante excepciones.

Uso de transacciones SQL para asegurar que las operaciones críticas (compras, cancelaciones) se ejecuten de manera atómica.

CÓDIGO DE CREACIÓN DE TABLAS UTILIZADO EN SQL SERVER

```
USE Proyecto_Final_BD;

-- CREACIÓN DE LA TABLA ROLES
CREATE TABLE ROLES (
    id_rol INT PRIMARY KEY,
    nombre_rol VARCHAR(50) NOT NULL -- insertar aquí los roles (cliente, administrador, etc.)
);

--CREACIÓN POST EJECUCIÓN: TABLA METODO_PAGO
CREATE TABLE METODO_PAGO (
id_metodo_pago INT PRIMARY KEY,
nombre_metodo VARCHAR (40) NOT NULL --metodos de pago: visa, paypal, ACH, etc.
);

-- CREACIÓN DE LA TABLA USUARIOS
CREATE TABLE USUARIOS (
    id_usuario INT IDENTITY (1,1) PRIMARY KEY,
    nombre VARCHAR(40) NOT NULL,
    apellido VARCHAR(40) NOT NULL,
    email VARCHAR(50) UNIQUE NOT NULL,
    telefono VARCHAR(15) UNIQUE NOT NULL,
    direccion NVARCHAR(100) NOT NULL,
    id_rol INT NOT NULL, -- LLAVE FORÁNEA HACIA TABLA ROLES

    CONSTRAINT FK_USUARIOS_ROLES FOREIGN KEY (id_rol) REFERENCES ROLES (id_rol)
    ON UPDATE CASCADE ON DELETE CASCADE
);

-- CREACIÓN DEL ÍNDICE PARA id_rol EN USUARIOS
```

```
CREATE INDEX idx_id_rol ON USUARIOS (id_rol);



-- CREACIÓN DE LA TABLA CATEGORIAS
CREATE TABLE CATEGORIAS (
    id_categoria INT PRIMARY KEY,
    nombre_categoria VARCHAR(50) NOT NULL,
    descripcion VARCHAR(100) NOT NULL
);

-- CREACIÓN DE LA TABLA PROVEEDORES_LB
CREATE TABLE PROVEEDORES_LB (
    id_proveedor INT PRIMARY KEY,
    nombre_proveedor VARCHAR(50) NOT NULL,
    telefono VARCHAR(15) UNIQUE NOT NULL,
    direccion VARCHAR(100) UNIQUE NOT NULL
);

-- CREACIÓN DE LA TABLA LIBROS
CREATE TABLE LIBROS (
    id_libro INT IDENTITY (1,1) PRIMARY KEY,
    titulo VARCHAR(100) NOT NULL,
    autor VARCHAR(50) NOT NULL,
    editorial VARCHAR(50) NOT NULL,
    publicacion DATE NOT NULL,
    precio DECIMAL(10, 2) CHECK (precio > 0) NOT NULL,
    stock INT CHECK (stock >= 0) NOT NULL,
    id_categoria INT NOT NULL, -- LLAVE FORÁNEA HACIA LA TABLA CATEGORIAS

    CONSTRAINT FK_LIBROS_CATEGORIAS FOREIGN KEY (id_categoria) REFERENCES CATEGORIAS (id_categoria)
    ON UPDATE CASCADE ON DELETE CASCADE
```

```
);

-- CREACIÓN DE LA TABLA PEDIDOS
CREATE TABLE PEDIDOS (
    id_pedido INT IDENTITY (1,1) PRIMARY KEY,
    id_usuario INT NOT NULL, -- LLAVE FORÁNEA HACIA LA TABLA USUARIOS
        id_metodo_pago INT NOT NULL, --LLAVE FORÁNEA HACIA LA TABLA METODO_PAGO
    fecha_pedido DATE NOT NULL,
    estado_pedido VARCHAR(20) NOT NULL, -- pendiente, enviado, cancelado, etc.

    CONSTRAINT FK_PEDIDOS_USUARIOS FOREIGN KEY (id_usuario) REFERENCES USUARIOS
(id_usuario)
    ON UPDATE CASCADE ON DELETE CASCADE,

        CONSTRAINT FK_PEDIDOS_METODO_PAGO FOREIGN KEY (id_metodo_pago)
REFERENCES METODO_PAGO (id_metodo_pago)
        ON UPDATE CASCADE ON DELETE CASCADE
);

-- CREACIÓN DE LA TABLA DETALLE_PEDIDOS
CREATE TABLE DETALLE_PEDIDOS (
    id_detalle_pedido INT IDENTITY (1,1) PRIMARY KEY,
    id_pedido INT NOT NULL, -- LLAVE FORÁNEA A LA TABLA PEDIDOS
    id_libro INT NOT NULL, -- LLAVE FORÁNEA A LA TABLA LIBROS
    cantidad INT CHECK (cantidad > 0) NOT NULL,
    precio_total DECIMAL(10, 2) NOT NULL,

    CONSTRAINT FK_DETALLE_PEDIDOS_PEDIDOS FOREIGN KEY (id_pedido) REFERENCES
PEDIDOS (id_pedido)
    ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT FK_DETALLE_PEDIDOS_LIBROS FOREIGN KEY (id_libro) REFERENCES
LIBROS (id_libro)
    ON UPDATE CASCADE ON DELETE CASCADE
```

```
);

-- CREACIÓN DE LA TABLA COMPRAS_PROVEEDOR
CREATE TABLE COMPRAS_PROVEEDOR (
    id_compra INT PRIMARY KEY,
    id_proveedor INT NOT NULL, -- LLAVE FORÁNEA CON LA TABLA PROVEEDORES_LB
    fecha_compra DATE NOT NULL,
    monto_total DECIMAL(10, 2) NOT NULL,

    CONSTRAINT FK_COMPRAS_PROVEEDOR_PROVEEDORES_LB FOREIGN KEY (id_proveedor) REFERENCES PROVEEDORES_LB (id_proveedor)
    ON UPDATE CASCADE ON DELETE CASCADE
);

-- CREACIÓN DE LA TABLA DETALLE_COMPRAS
CREATE TABLE DETALLE_COMPRAS (
    id_detalle_compra INT PRIMARY KEY,
    id_compra INT NOT NULL, -- LLAVE FORÁNEA HACIA LA TABLA COMPRAS_PROVEEDOR
    id_libro INT NOT NULL, -- LLAVE FORÁNEA HACIA LA TABLA LIBROS
    cantidad INT CHECK (cantidad > 0) NOT NULL,
    costo_unitario DECIMAL(10, 2) CHECK (costo_unitario > 0) NOT NULL,

    CONSTRAINT FK_DETALLE_COMPRAS_COMPRAS_PROVEEDOR FOREIGN KEY (id_compra) REFERENCES COMPRAS_PROVEEDOR (id_compra)
    ON DELETE CASCADE,
    CONSTRAINT FK_DETALLE_COMPRAS_LIBROS FOREIGN KEY (id_libro) REFERENCES LIBROS (id_libro)
    ON DELETE CASCADE
);
```

COMPOSICIÓN DE LA APLICACIÓN

Esta aplicación se compone de 4 formularios:

a. Form1.vb
b. FrmLogin.vb
c. FrmMenu.vb
d. FrmMenuCliente.vb

Form1.vb:

El Form1 es el punto de inicio de la aplicación y sirve como puente inicial para conectar a la base de datos Proyecto_Final_BD. Este formulario tiene una función primordial: establecer una conexión con la base de datos y redirigir al usuario al formulario de autenticación (FrmLogin), desde donde se determina el flujo de acceso a los menús principales de la aplicación.

Funciones principales:

Pantalla inicial:

Es el primer formulario que se muestra al ejecutar la aplicación.

Presenta un diseño simple y profesional, con un botón centralizado que invita al usuario a conectarse a la base de datos.

Conexión a la base de datos:

Usa una cadena de conexión predeterminada para conectarse al servidor de base de datos Proyecto_Final_BD mediante autenticación integrada de Windows.

Verifica que la conexión no esté ya abierta antes de intentar establecerla.

Redirección al formulario de autenticación:

Al hacer clic en el botón Conectar a la Base de Datos, oculta el Form1 y muestra el formulario FrmLogin.

Si el usuario cierra el FrmLogin, el Form1 se vuelve a mostrar.

Manejo de errores:

Si ocurre un error al intentar conectarse a la base de datos, muestra un mensaje de error detallado para ayudar a diagnosticar el problema.

Gestión de recursos:

Implementa un cierre adecuado de la conexión a la base de datos cuando el formulario se cierra o la aplicación finaliza.

Flujo del formulario:

El usuario ejecuta la aplicación, y se muestra el Form1.

El usuario hace clic en el botón Conectar a la Base de Datos:

Se establece la conexión con la base de datos.

El formulario se oculta, y se abre el FrmLogin.

Si el usuario cierra el FrmLogin, el Form1 vuelve a ser visible.

Al cerrar el Form1, la conexión con la base de datos se cierra para liberar recursos.

Código del Form1.vb empleado en Visual Studio:

```
Imports System.Data.SqlClient
```

```
Public Class Form1
    ' Variables globales
    Private connectionString As String = "Data Source=LAPTOP-82OAAGJ0\MSSQLSERVER_2022;Initial Catalog=Proyecto_Final_BD;Integrated Security=True;"
    Private conexion As SqlConnection

    ' Evento Load del formulario principal
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "Inicio - Proyecto_Final_BD"
        Me.StartPosition = FormStartPosition.CenterScreen ' Centrar formulario al iniciar
        Me.BackColor = Color.MidnightBlue ' Fondo azul oscuro
        Me.ForeColor = Color.White ' Texto blanco
        Me.Font = New Font("Consolas", 12, FontStyle.Bold) ' Fuente más grande
        Me.Width = 500
        Me.Height = 250

        ' Crear un botón para iniciar conexión
        Dim btnConectar As New Button With {
            .Text = "Conectar a la Base de Datos",
            .Width = 250,
            .Height = 60,
            .Top = (Me.ClientSize.Height - 60) \ 2,
            .Left = (Me.ClientSize.Width - 250) \ 2,
            .BackColor = Color.White,
            .ForeColor = Color.MidnightBlue,
            .Font = New Font("Consolas", 12, FontStyle.Bold) ' Ajustar fuente del botón
```

```
    }
    AddHandler btnConectar.Click, AddressOf BtnConectar_Click
    Me.Controls.Add(btnConectar)
End Sub

' Lógica al hacer clic en el botón "Conectar"
Private Sub BtnConectar_Click(sender As Object, e As EventArgs)
    Try
        ' Verificar si la conexión ya está abierta
        If conexion Is Nothing Then
            conexion = New SqlConnection(connectionString)
        End If

        If conexion.State <> ConnectionState.Open Then
            conexion.Open()
        End If

        ' Ocultar este formulario
        Me.Hide()

        ' Abrir el formulario de login y pasar los argumentos necesarios
        Dim loginForm As New FrmLogin(Me, conexion)
        AddHandler loginForm.FormClosed, Sub(s, args)
                                    ' Mostrar este formulario si se cierra el login
                                    Me.Show()
                                End Sub
        loginForm.ShowDialog()
```

```
        Catch ex As Exception
            MessageBox.Show("Error al conectar con la base de datos: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Método para ocultar este formulario al abrir los menús
    Public Sub Ocultar()
        Me.Hide()
    End Sub

    ' Evento al cerrar la aplicación
    Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles MyBase.FormClosing
        ' Cerrar la conexión si está abierta
        If conexion IsNot Nothing AndAlso conexion.State = ConnectionState.Open Then
            conexion.Close()
        End If
    End Sub
End Class
```

FrmLogin:

El FrmLogin es el segundo formulario de la aplicación, diseñado para autenticar a los usuarios y determinar el acceso a los diferentes menús de la aplicación según su rol.

Este formulario establece las credenciales y valida el rol para abrir el menú correspondiente (FrmMenu para administradores y FrmMenuCliente para clientes).

Funciones principales del FrmLogin:

Autenticación de usuario SQL Server:

Solicita al usuario las credenciales de SQL Server (sa como usuario y Perromuerto98 como contraseña por defecto).

Comprueba que las credenciales coincidan antes de permitir el acceso.

Validación de rol:

Solicita un rol adicional al usuario (1 para Administrador y 2 para Cliente).

Verifica que el rol ingresado exista en la tabla ROLES de la base de datos Proyecto_Final_BD.

Si el rol no es válido, muestra un mensaje de error y solicita nuevamente.

Redirección al menú correspondiente:

Si las credenciales y el rol son correctos:

Rol 1 (Administrador): Abre el formulario FrmMenu.

Rol 2 (Cliente): Abre el formulario FrmMenuCliente.

Cierra el FrmLogin después de redirigir al menú correspondiente.

Gestión de errores:

Valida la entrada del usuario para asegurarse de que no haya campos vacíos.

Muestra mensajes de error específicos si las credenciales o el rol son incorrectos.

Captura excepciones relacionadas con la base de datos, como problemas al ejecutar consultas SQL.

Gestión de la conexión:

Usa la conexión establecida por el Form1.

Garantiza que la conexión a la base de datos esté abierta solo durante la validación y se cierre inmediatamente después de ser utilizada.

Interfaz centrada y personalizada:

La ventana está centrada en la pantalla y cuenta con un diseño profesional, con colores azul oscuro (fondo) y blanco (texto).

Los campos de texto y el botón están bien organizados y estilizados para una experiencia de usuario fluida.

Flujo del formulario:

Inicio:

El formulario se abre después de que el Form1 establece la conexión a la base de datos.

Ingreso de datos:

Solicita al usuario ingresar las credenciales y el rol.

Valida la autenticidad de los datos.

Redirección:

Si los datos son correctos, abre el menú correspondiente según el rol del usuario.

Cierra el FrmLogin al abrir el menú.

Error handling:

Si hay errores en las credenciales o rol, muestra un mensaje y permite intentarlo de nuevo.

FrmMenu:

El FrmMenu es el menú principal destinado a los administradores del sistema. Este formulario proporciona acceso completo a las funciones administrativas para gestionar la base de datos Proyecto_Final_BD, incluyendo la administración de usuarios, libros, proveedores, compras, pedidos y detalles de compras.

Funciones principales del FrmMenu:

Interfaz principal para administradores:

Permite a los administradores acceder y gestionar los datos clave de la base de datos.

Incluye un conjunto de botones organizados horizontalmente en la parte superior del formulario para facilitar la navegación.

Visualización de datos en un DataGridView:

Los datos de las tablas seleccionadas se cargan y se muestran dinámicamente en un DataGridView, que está oculto por defecto y se hace visible al ejecutar una consulta.

Gestión de usuarios:

Mostrar Usuarios: Permite visualizar la lista completa de usuarios registrados en la base de datos.

Eliminar Usuario: Solicita un ID de usuario, valida su existencia y elimina el registro si el usuario confirma la acción.

Gestión de libros y proveedores:

Libros Disponibles: Muestra todos los libros disponibles en el inventario.

Mostrar Proveedores: Visualiza la lista de proveedores registrados.

Gestión de compras y pedidos:

Detalles Compras: Muestra información detallada sobre las compras realizadas a los proveedores.

Compra Proveedor: Ejecuta un procedimiento almacenado (ComprarLibroProveedor) para registrar la compra de libros a un proveedor. Solicita al administrador el ID del proveedor, ID del libro, cantidad y costo unitario de los libros.

Mostrar Pedidos: Presenta una lista de todos los pedidos realizados por los clientes.

Configuración del diseño:

Panel superior para botones:

Los botones están organizados horizontalmente y cuentan con animaciones visuales que resaltan al pasar el cursor sobre ellos.

DataGridView dinámico:

Se ajusta automáticamente al ancho del formulario y está diseñado para una fácil lectura con colores claros y sin encabezados de fila visibles.

Funciones adicionales:

Validaciones: Verifica si los IDs proporcionados existen en la base de datos antes de realizar acciones críticas, como eliminar usuarios.

Ejecución de comandos SQL: Proporciona métodos genéricos para ejecutar consultas y comandos SQL con parámetros.

Conexión a la base de datos: Asegura que la conexión esté abierta antes de ejecutar cualquier operación y la cierra automáticamente en caso de error.

Flujo lógico:

Al seleccionar una acción (como mostrar usuarios, libros o pedidos), el formulario consulta la base de datos y muestra los resultados en el DataGridView.

Para acciones que requieren entrada de datos (como eliminar usuarios o comprar libros), se solicita información adicional al usuario mediante cuadros de diálogo.

Resumen del flujo:

Inicio:

El formulario se abre después de que un usuario administrador inicia sesión desde el FrmLogin.

Acceso a funciones:

Los botones permiten al administrador acceder rápidamente a diversas funciones administrativas, desde visualizar datos hasta realizar transacciones complejas.

Interacción dinámica:

El DataGridView se actualiza automáticamente para reflejar los datos relevantes según la acción seleccionada.

Gestión y control:

El administrador puede realizar tareas críticas, como registrar compras a proveedores, eliminar usuarios o revisar pedidos, con validaciones y mensajes de confirmación.

Propósito:

El FrmMenu es el núcleo del sistema administrativo, diseñado para facilitar la gestión integral de la base de datos mediante una interfaz intuitiva, validaciones robustas y opciones de personalización.

Código del FrmMenu:

```
Imports System.Data.SqlClient

Public Class FrmMenu
    Inherits Form

    Private conexion As SqlConnection
    Private dgvResultados As DataGridView ' DataGridView como variable global
    Private panelBotones As Panel ' Panel para contener los botones

    Public Sub New(conn As SqlConnection)
        MyBase.New()
        Me.conexion = conn

        ' Configuración del formulario
        Me.Text = "Menú de "
        Me.StartPosition = FormStartPosition.CenterScreen ' Centrado
        Me.BackColor = Color.MidnightBlue ' Fondo azul oscuro
        Me.ForeColor = Color.White ' Texto blanco
        Me.Font = New Font("Consolas", 10, FontStyle.Regular) ' Fuente estilo Python
        Me.Width = 800
        Me.Height = 600

        ' Inicializar controles
        InicializarControles()
    End Sub

    ' Método para inicializar controles
    Private Sub InicializarControles()
        ' Crear panel para botones (barra horizontal)
```

```
panelBotones = New Panel With {
    .Top = 0,
    .Left = 0,
    .Width = Me.ClientSize.Width,
    .Height = 50,
    .BackColor = Color.WhiteSmoke,
    .Dock = DockStyle.Top ' Se adhiere a la parte superior del formulario
}
Me.Controls.Add(panelBotones)

' Crear botones en el panel
Dim botones = New List(Of Button) From {
    CrearBoton("Usuarios", AddressOf MostrarUsuarios),
    CrearBoton("Libros Disponibles", AddressOf MostrarLibros),
    CrearBoton("Mostrar Proveedores", AddressOf MostrarProv),
    CrearBoton("Detalles Compras", AddressOf MostrarDetalles),
    CrearBoton("Compra Proveedor", AddressOf ComprarLibroProveedor),
    CrearBoton("Mostrar Pedidos", AddressOf MostrarPedidos),
    CrearBoton("Eliminar Usuario", AddressOf EliminarUsuario)
}

' Posicionar botones horizontalmente dentro del panel
Dim botonAncho = panelBotones.Width \ botones.Count
For i = 0 To botones.Count - 1
    botones(i).Width = botonAncho
    botones(i).Height = panelBotones.Height
    botones(i).Top = 0
    botones(i).Left = i * botonAncho
    panelBotones.Controls.Add(botones(i))
Next
```

```
    ' Crear DataGridView (oculto inicialmente)
    dgvResultados = New DataGridView With {
        .Name = "dgvResultados",
        .Width = Me.ClientSize.Width - 40,
        .Height = 400,
        .Top = panelBotones.Bottom + 20,
        .Left = 20,
        .Visible = False,
        .BackgroundColor = Color.WhiteSmoke,
        .ForeColor = Color.Black,
        .Font = New Font("Consolas", 10, FontStyle.Regular),
        .RowHeadersVisible = False,
        .SelectionMode = DataGridViewSelectionMode.FullRowSelect
    }
    Me.Controls.Add(dgvResultados)
End Sub

' Crear un botón personalizado con animación al pasar el cursor
Private Function CrearBoton(texto As String, metodo As EventHandler) As Button
    Dim boton As New Button With {
        .Text = texto,
        .BackColor = Color.MidnightBlue,
        .ForeColor = Color.White,
        .FlatStyle = FlatStyle.Flat,
        .Font = New Font("Consolas", 10, FontStyle.Regular),
        .Dock = DockStyle.None ' No se adhiere automáticamente
    }
    AddHandler boton.Click, metodo

    ' Animación al pasar el cursor
    AddHandler boton.MouseEnter, Sub(sender As Object, e As EventArgs)
```

```
                        boton.BackColor = Color.DarkOrange
                        boton.ForeColor = Color.Black
                    End Sub
    AddHandler boton.MouseLeave, Sub(sender As Object, e As EventArgs)
                        boton.BackColor = Color.MidnightBlue
                        boton.ForeColor = Color.White
                    End Sub

    Return boton
End Function

' Mostrar usuarios en el DataGridView
Private Sub MostrarUsuarios(sender As Object, e As EventArgs)
    dgvResultados.Visible = True
    EjecutarConsulta("SELECT * FROM USUARIOS", dgvResultados)
End Sub

' Mostrar los libros disponibles en el DataGridView
Private Sub MostrarLibros(sender As Object, e As EventArgs)
    dgvResultados.Visible = True
    EjecutarConsulta("SELECT * FROM LIBROS", dgvResultados)
End Sub

'Mostrar la tabla de PROVEEDORES_LB para hacer compras
Private Sub MostrarProv(sender As Object, e As EventArgs)
    dgvResultados.Visible = True
    EjecutarConsulta("SELECT * FROM PROVEEDORES_LB", dgvResultados)
End Sub

'Mostrar la tabla DETALLE_COMPRAS para verificar las compras a proveedores y sus detalles
Private Sub MostrarDetalles(sender As Object, e As EventArgs)
```

```
        dgvResultados.Visible = True
        EjecutarConsulta("SELECT * FROM DETALLE_COMPRAS", dgvResultados)
    End Sub



    ' Método genérico para ejecutar consultas SQL y mostrar resultados
    Private Sub EjecutarConsulta(query As String, dgv As DataGridView)
        Try
            VerificarConexion()
            Dim comando As New SqlCommand(query, conexion)
            Dim adaptador As New SqlDataAdapter(comando)
            Dim dataTable As New DataTable()
            adaptador.Fill(dataTable)
            dgv.DataSource = dataTable
        Catch ex As Exception
            MessageBox.Show("Error al ejecutar consulta: " & ex.Message)
        End Try
    End Sub




    'Método: Compra a Proveedor
    Private Sub ComprarLibroProveedor()
        Try
            Dim idProveedor As Integer = CInt(InputBox("Ingrese el ID del proveedor:", "Compra a Proveedor"))
            Dim idLibro As Integer = CInt(InputBox("Ingrese el ID del libro:", "Compra a Proveedor"))
            Dim cantidad As Integer = CInt(InputBox("Ingrese la cantidad a comprar:", "Compra a Proveedor"))
            Dim costoUnitario As Decimal = CDec(InputBox("Ingrese el costo unitario del libro:", "Compra a Proveedor"))
```

```
        Dim comando As New SqlCommand("ComprarLibroProveedor", conexion)
        comando.CommandType = CommandType.StoredProcedure

        comando.Parameters.AddWithValue("@id_proveedor", idProveedor)
        comando.Parameters.AddWithValue("@id_libro", idLibro)
        comando.Parameters.AddWithValue("@cantidad", cantidad)
        comando.Parameters.AddWithValue("@costo_unitario", costoUnitario)

        VerificarConexion()
        comando.ExecuteNonQuery()
        MessageBox.Show("Compra al proveedor realizada exitosamente.")
    Catch ex As Exception
        MessageBox.Show("Error al realizar la compra al proveedor: " & ex.Message)
    End Try
End Sub



' Eliminar un usuario
Private Sub EliminarUsuario(sender As Object, e As EventArgs)
    Dim idUsuario = InputBox("Ingrese el ID del usuario a eliminar:", "Eliminar Usuario")
    If Not ValidarID(idUsuario) Then Return

    Dim confirmar As DialogResult = MessageBox.Show("¿Está seguro de que desea eliminar este usuario?", "Confirmar Eliminación", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
    If confirmar = DialogResult.No Then Return

    Dim query As String = "DELETE FROM USUARIOS WHERE id_usuario = @id_usuario"
    EjecutarComando(query, New Dictionary(Of String, String) From {{"@id_usuario", idUsuario}})
End Sub

"MOSTRAR LA TABLA PEDIDOS PARA ADMINISTRAR LAS COMPRAS HECHAS POR CLIENTES
```

```
Private Sub MostrarPedidos(sender As Object, e As EventArgs)
    dgvResultados.Visible = True
    EjecutarConsulta("SELECT * FROM PEDIDOS", dgvResultados)
End Sub




' Ejecutar comandos SQL con parámetros
Private Sub EjecutarComando(query As String, parametros As Dictionary(Of String, String))
    Try
        VerificarConexion()
        Dim comando As New SqlCommand(query, conexion)
        For Each kvp In parametros
            comando.Parameters.AddWithValue(kvp.Key, kvp.Value)
        Next
        Dim filasAfectadas = comando.ExecuteNonQuery()
        MessageBox.Show($"Operación completada. Filas afectadas: {filasAfectadas}")
    Catch ex As Exception
        MessageBox.Show("Error al ejecutar comando: " & ex.Message)
    End Try
End Sub

' Solicitar datos con múltiples entradas
Private Function SolicitarDatos(campos As String()) As Dictionary(Of String, String)
    Dim datos As New Dictionary(Of String, String)
    For Each campo In campos
        Dim valor = InputBox($"Ingrese {campo}:", "Registrar/Actualizar Usuario")
        If String.IsNullOrEmpty(valor) Then
            MessageBox.Show("Debe ingresar todos los datos.")
            Return Nothing
        End If
```

```
            datos($"@{campo.ToLower()}") = valor
        Next
        Return datos
    End Function



    ' Validar si un ID es válido y existe en la base de datos
    Private Function ValidarID(idUsuario As String) As Boolean
        If String.IsNullOrEmpty(idUsuario) Then
            MessageBox.Show("Debe ingresar un ID válido.")
            Return False
        End If

        Dim queryValidar As String = "SELECT COUNT(*) FROM USUARIOS WHERE id_usuario =
@id_usuario"
        Try
            VerificarConexion()
            Dim comando As New SqlCommand(queryValidar, conexion)
            comando.Parameters.AddWithValue("@id_usuario", idUsuario)
            Dim existe As Integer = Convert.ToInt32(comando.ExecuteScalar())
            If existe = 0 Then
                MessageBox.Show("El ID del usuario no existe.")
                Return False
            End If
            Return True
        Catch ex As Exception
            MessageBox.Show("Error al validar ID: " & ex.Message)
            Return False
        End Try
    End Function

    ' Verificar y abrir la conexión
```

```
    Private Sub VerificarConexion()
        If conexion.State <> ConnectionState.Open Then conexion.Open()
    End Sub
End Class
```

FrmMenuCliente.vb:

El formulario FrmMenuCliente está diseñado como una interfaz gráfica en Visual Basic .NET que permite a los clientes interactuar con la base de datos Proyecto_Final_BD. Este formulario se centra en funcionalidades específicas relacionadas con los clientes y sus acciones dentro del sistema, como explorar libros, realizar compras y gestionar usuarios. Aquí está un desglose de su función principal y componentes:

Función principal

Proveer un menú interactivo y fácil de usar para que los clientes puedan:

Consultar libros disponibles.

Realizar compras de libros y ver detalles de sus compras (factura).

Consultar métodos de pago y categorías de libros.

Administrar su información personal mediante la actualización de datos.

Registrar nuevos usuarios como clientes (en escenarios particulares).

Cancelar compras si están en estado "Pendiente".

Componentes principales

1. Controles visuales:

Barra de botones (Panel):

Incluye botones como:

Mostrar Libros: Consulta y despliega todos los libros disponibles en un DataGridView.

Mostrar Clientes: Lista usuarios con rol de cliente (id_rol = 2).

Mostrar Categorías: Lista categorías disponibles de libros con sus descripciones.

Comprar Libro: Permite a los clientes realizar compras de libros utilizando un procedimiento almacenado ComprarLibroCliente2.

Cancelar Compra: Cancela un pedido si su estado es "Pendiente" ejecutando el procedimiento almacenado CancelarCompra.

Registrar Usuario y Actualizar Usuario: Gestionan la información de los clientes.

Los botones están diseñados con un estilo personalizado y animaciones al pasar el cursor.

DataGridViews (DGVs):

dgvLibros: Muestra la lista de libros disponibles.

dgvClientes: Despliega los datos de usuarios con rol de cliente.

dgvMetodosPago: Lista los métodos de pago disponibles.

Estas tablas están ocultas inicialmente y se activan al seleccionar las opciones correspondientes.

2. Funcionalidades:

Compra de libros:

Solicita datos como ID del usuario, ID del libro, cantidad y ID del método de pago.

Ejecuta el procedimiento almacenado ComprarLibroCliente2 que actualiza el stock de libros, registra el pedido y genera una factura con los detalles de la compra.

Cancelación de compras:

Permite cancelar un pedido en estado "Pendiente" utilizando el procedimiento almacenado CancelarCompra.

Visualización de categorías:

Muestra las categorías disponibles junto con sus descripciones, utilizando la tabla CATEGORIAS.

Gestión de usuarios:

Registrar usuario: Inserta nuevos usuarios en la base de datos.

Actualizar usuario: Permite editar los datos de usuarios existentes.

Facturación:

Despliega detalles del pedido más reciente realizado por el cliente (como ID de pedido, título del libro, cantidad y total) en un cuadro emergente (MessageBox) después de una compra exitosa.

Detalles técnicos

Conexión SQL:

Utiliza SqlConnection para conectarse a la base de datos.

Las consultas y procedimientos almacenados se ejecutan mediante comandos SQL (SqlCommand).

Manejo de excepciones:

Cada operación está protegida con bloques Try...Catch para manejar errores y proporcionar retroalimentación al usuario.

Validación:

Verifica la existencia de usuarios mediante consultas.

Valida entradas como IDs y rol de usuario para evitar errores en la interacción con la base de datos.

Propósito del formulario

El FrmMenuCliente proporciona una interfaz interactiva que permite a los clientes explorar, gestionar y realizar acciones dentro del sistema de manera efectiva y segura, utilizando una conexión directa con la base de datos. Su diseño está orientado a la usabilidad, centrado exclusivamente en las necesidades de los clientes.

Código del FrmMenuCliente.vb:

```
Imports System.Data.SqlClient
Public Class FrmMenuCliente
    Inherits Form

    Private conexion As SqlConnection
    Private dgvLibros As DataGridView
    Private dgvClientes As DataGridView
    Private dgvMetodosPago As DataGridView ' Nuevo DataGridView para los Métodos de Pago

    Public Sub New(conn As SqlConnection)
        MyBase.New()
        Me.conexion = conn

        ' Configuración del formulario
        Me.Text = "Menú Cliente"
        Me.StartPosition = FormStartPosition.CenterScreen
        Me.BackColor = Color.MidnightBlue
        Me.ForeColor = Color.White
        Me.Font = New Font("Consolas", 10, FontStyle.Regular)
        Me.Width = 800
```

```
    Me.Height = 700 ' Ajustar el tamaño para agregar un DataGridView adicional

    ' Inicializar controles
    InicializarControles()
End Sub

' Inicializar controles
Private Sub InicializarControles()
    ' Panel para botones
    Dim panelBotones As New Panel With {
        .Top = 0,
        .Left = 0,
        .Width = Me.ClientSize.Width,
        .Height = 50,
        .BackColor = Color.WhiteSmoke,
        .Dock = DockStyle.Top
    }
    Me.Controls.Add(panelBotones)

    ' Botones
    Dim botones = New List(Of Button) From {
        CrearBoton("Mostrar Libros", AddressOf MostrarLibros),
        CrearBoton("Mostrar Clientes", AddressOf MostrarClientes),
        CrearBoton("Comprar Libro", AddressOf ComprarLibro),
        CrearBoton("Cancelar Compra", AddressOf CancelarCompra),
        CrearBoton("Registrar Usuario", AddressOf RegistrarUsuario),
        CrearBoton("Actualizar Usuario", AddressOf ActualizarUsuario),
        CrearBoton("Mostrar Categorías", AddressOf MostrarCategorias)
    }

    ' Posicionar botones horizontalmente
```

```
Dim botonAncho = panelBotones.Width \ botones.Count
For i = 0 To botones.Count - 1
    botones(i).Width = botonAncho
    botones(i).Height = panelBotones.Height
    botones(i).Top = 0
    botones(i).Left = i * botonAncho
    panelBotones.Controls.Add(botones(i))
Next

' DataGridView para Libros (Oculto inicialmente)
dgvLibros = New DataGridView With {
    .Name = "dgvLibros",
    .Width = Me.ClientSize.Width - 40,
    .Height = (Me.ClientSize.Height - panelBotones.Height - 90) \ 3,
    .Top = panelBotones.Bottom + 10,
    .Left = 20,
    .Visible = False,
    .BackgroundColor = Color.WhiteSmoke,
    .ForeColor = Color.Black,
    .Font = New Font("Consolas", 10, FontStyle.Regular),
    .RowHeadersVisible = False,
    .SelectionMode = DataGridViewSelectionMode.FullRowSelect
}
Me.Controls.Add(dgvLibros)

' DataGridView para Clientes (Oculto inicialmente)
dgvClientes = New DataGridView With {
    .Name = "dgvClientes",
    .Width = Me.ClientSize.Width - 40,
    .Height = (Me.ClientSize.Height - panelBotones.Height - 90) \ 3,
    .Top = dgvLibros.Bottom + 10,
```

```
        .Left = 20,
        .Visible = False,
        .BackgroundColor = Color.WhiteSmoke,
        .ForeColor = Color.Black,
        .Font = New Font("Consolas", 10, FontStyle.Regular),
        .RowHeadersVisible = False,
        .SelectionMode = DataGridViewSelectionMode.FullRowSelect
    }
    Me.Controls.Add(dgvClientes)

    ' DataGridView para Métodos de Pago (Oculto inicialmente)
    dgvMetodosPago = New DataGridView With {
        .Name = "dgvMetodosPago",
        .Width = Me.ClientSize.Width - 40,
        .Height = (Me.ClientSize.Height - panelBotones.Height - 90) \ 3,
        .Top = dgvClientes.Bottom + 10,
        .Left = 20,
        .Visible = False,
        .BackgroundColor = Color.WhiteSmoke,
        .ForeColor = Color.Black,
        .Font = New Font("Consolas", 10, FontStyle.Regular),
        .RowHeadersVisible = False,
        .SelectionMode = DataGridViewSelectionMode.FullRowSelect
    }
    Me.Controls.Add(dgvMetodosPago)


End Sub

' Crear un botón personalizado
Private Function CrearBoton(texto As String, metodo As EventHandler) As Button
```

```
    Dim boton As New Button With {
        .Text = texto,
        .BackColor = Color.MidnightBlue,
        .ForeColor = Color.White,
        .FlatStyle = FlatStyle.Flat,
        .Font = New Font("Consolas", 10, FontStyle.Regular)
    }
    AddHandler boton.Click, metodo

    ' Animación al pasar el cursor
    AddHandler boton.MouseEnter, Sub(sender As Object, e As EventArgs)
                                boton.BackColor = Color.DarkOrange
                                boton.ForeColor = Color.Black
                            End Sub
    AddHandler boton.MouseLeave, Sub(sender As Object, e As EventArgs)
                                boton.BackColor = Color.MidnightBlue
                                boton.ForeColor = Color.White
                            End Sub

    Return boton
End Function

' Mostrar Libros en el DataGridView
Private Sub MostrarLibros(sender As Object, e As EventArgs)
    dgvLibros.Visible = True
    EjecutarConsulta("SELECT * FROM LIBROS", dgvLibros)
End Sub

' Mostrar Clientes en el DataGridView
Private Sub MostrarClientes(sender As Object, e As EventArgs)
    dgvClientes.Visible = True
```

```
        EjecutarConsulta("SELECT * FROM USUARIOS WHERE id_rol = 2", dgvClientes)
    End Sub

    ' Mostrar Métodos de Pago en el DataGridView
    Private Sub MostrarMetodosPago(sender As Object, e As EventArgs)
        dgvMetodosPago.Visible = True
        EjecutarConsulta("SELECT * FROM METODO_PAGO", dgvMetodosPago)
    End Sub

    ' Método para mostrar las categorías en un DataGridView
    Private Sub MostrarCategorias(sender As Object, e As EventArgs)
        Try
            ' Mostrar categorías en un DataGridView (puedes elegir uno existente o agregar otro)
            If dgvClientes IsNot Nothing Then
                dgvClientes.Visible = True
                dgvLibros.Visible = False
                dgvMetodosPago.Visible = False
            End If

            ' Consulta para obtener las categorías
            Dim query As String = "SELECT id_categoria AS 'ID', nombre_categoria AS 'Categoría',
descripcion AS 'Descripción' FROM CATEGORIAS"
            EjecutarConsulta(query, dgvClientes) ' Utilizamos dgvClientes como visualización
        Catch ex As Exception
            MessageBox.Show("Error al mostrar categorías: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Registrar un Usuario
    Private Sub RegistrarUsuario(sender As Object, e As EventArgs)
        Try
```

```
        ' Solicitar datos del usuario
        Dim nombre As String = InputBox("Ingrese el nombre del usuario:", "Registrar Usuario")
        Dim apellido As String = InputBox("Ingrese el apellido del usuario:", "Registrar Usuario")
        Dim email As String = InputBox("Ingrese el correo del usuario:", "Registrar Usuario")
        Dim telefono As String = InputBox("Ingrese el teléfono del usuario:", "Registrar Usuario")
        Dim direccion As String = InputBox("Ingrese la dirección del usuario:", "Registrar Usuario")
        Dim idRol As String = InputBox("Ingrese el id de su rol (1 para Administrador, 2 para
Cliente):", "Registrar Usuario")

        If String.IsNullOrEmpty(nombre) Or String.IsNullOrEmpty(email) Then
            MessageBox.Show("Debe ingresar todos los datos.")
            Return
        End If

        Dim query As String = "INSERT INTO USUARIOS (nombre, apellido, email, telefono,
direccion, id_rol) " &
                              "VALUES (@nombre, @apellido, @email, @telefono, @direccion, @id_rol)"
        Dim parametros As New Dictionary(Of String, String) From {
            {"@nombre", nombre},
            {"@apellido", apellido},
            {"@email", email},
            {"@telefono", telefono},
            {"@direccion", direccion},
            {"@id_rol", idRol}
        }

        EjecutarComando(query, parametros)
    Catch ex As Exception
        MessageBox.Show("Error al registrar usuario: " & ex.Message)
    End Try
End Sub
```

```
    ' Actualizar un Usuario
    Private Sub ActualizarUsuario(sender As Object, e As EventArgs)
        Try
            ' Solicitar ID del usuario
            Dim idUsuario As String = InputBox("Ingrese el ID del usuario a actualizar:", "Actualizar
Usuario")
            If Not ValidarID(idUsuario) Then Return

            ' Solicitar los nuevos datos del usuario
            Dim nombre As String = InputBox("Ingrese el nuevo nombre del usuario:", "Actualizar
Usuario")
            Dim apellido As String = InputBox("Ingrese el nuevo apellido del usuario:", "Actualizar
Usuario")
            Dim email As String = InputBox("Ingrese el nuevo correo del usuario:", "Actualizar Usuario")
            Dim telefono As String = InputBox("Ingrese el nuevo teléfono del usuario:", "Actualizar
Usuario")
            Dim direccion As String = InputBox("Ingrese la nueva dirección del usuario:", "Actualizar
Usuario")
            Dim idRol As String = InputBox("Ingrese el nuevo id de su rol (1 para Administrador, 2 para
Cliente):", "Actualizar Usuario")

            If String.IsNullOrEmpty(nombre) Or String.IsNullOrEmpty(email) Then
                MessageBox.Show("Debe ingresar todos los datos.")
                Return
            End If

            Dim query As String = "UPDATE USUARIOS SET nombre = @nombre, apellido = @apellido,
email = @email, " &
                                  "telefono = @telefono, direccion = @direccion, id_rol = @id_rol WHERE
id_usuario = @id_usuario"
            Dim parametros As New Dictionary(Of String, String) From {
                {"@nombre", nombre},
                {"@apellido", apellido},
                {"@email", email},
```

```
            {"@telefono", telefono},
            {"@direccion", direccion},
            {"@id_rol", idRol},
            {"@id_usuario", idUsuario}
        }

        EjecutarComando(query, parametros)
    Catch ex As Exception
        MessageBox.Show("Error al actualizar usuario: " & ex.Message)
    End Try
End Sub

Private Sub CancelarCompra(sender As Object, e As EventArgs)
    Try
        Dim idPedido As Integer = CInt(InputBox("Ingrese el ID del Pedido a cancelar:", "Cancelar
Compra"))

        ' Crear el comando para ejecutar el procedimiento almacenado
        Dim comando As New SqlCommand("CancelarCompra", conexion)
        comando.CommandType = CommandType.StoredProcedure

        ' Agregar el parámetro al comando
        comando.Parameters.AddWithValue("@id_pedido", idPedido)

        ' Verificar que la conexión esté abierta
        VerificarConexion()

        ' Ejecutar el procedimiento
        comando.ExecuteNonQuery()

        ' Confirmar la cancelación
        MessageBox.Show("Compra cancelada exitosamente.")
```

```
        Catch ex As Exception
            MessageBox.Show("Error al cancelar la compra: " & ex.Message)
        End Try
    End Sub



    ' Validar si un ID es válido y existe en la base de datos
    Private Function ValidarID(idUsuario As String) As Boolean
        If String.IsNullOrEmpty(idUsuario) Then
            MessageBox.Show("Debe ingresar un ID válido.")
            Return False
        End If

        Dim queryValidar As String = "SELECT COUNT(*) FROM USUARIOS WHERE id_usuario =
@id_usuario"
        Try
            VerificarConexion()
            Dim comando As New SqlCommand(queryValidar, conexion)
            comando.Parameters.AddWithValue("@id_usuario", idUsuario)
            Dim existe As Integer = Convert.ToInt32(comando.ExecuteScalar())
            If existe = 0 Then
                MessageBox.Show("El ID del usuario no existe.")
                Return False
            End If
            Return True
        Catch ex As Exception
            MessageBox.Show("Error al validar ID: " & ex.Message)
            Return False
        End Try
    End Function

    ' Método genérico para ejecutar consultas SQL y mostrar resultados
```

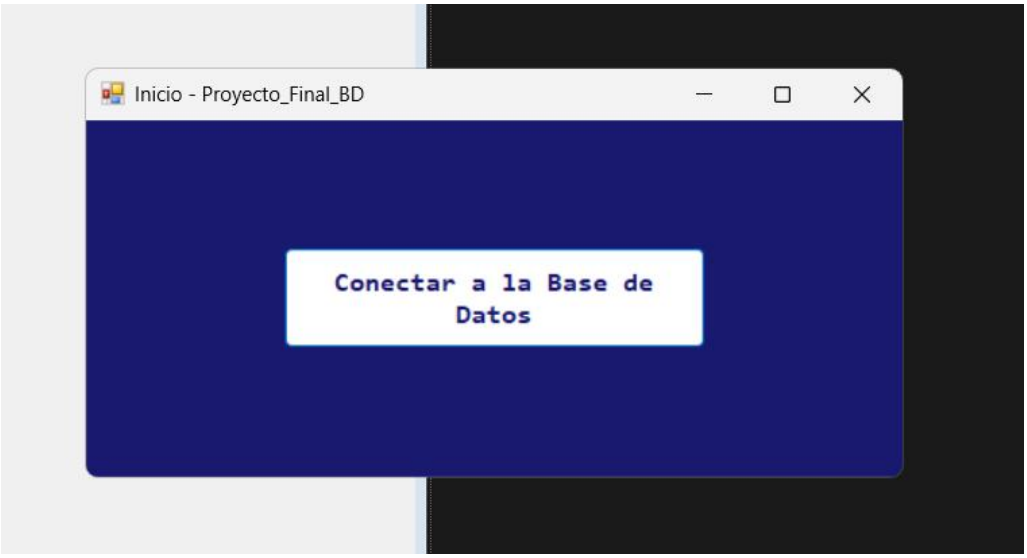
```
    Private Sub EjecutarConsulta(query As String, dgv As DataGridView)
        Try
            VerificarConexion()
            Dim comando As New SqlCommand(query, conexion)
            Dim adaptador As New SqlDataAdapter(comando)
            Dim dataTable As New DataTable()
            adaptador.Fill(dataTable)
            dgv.DataSource = dataTable
        Catch ex As Exception
            MessageBox.Show("Error al ejecutar consulta: " & ex.Message)
        End Try
    End Sub

    ' Comprar Libro
    Private Sub ComprarLibro(sender As Object, e As EventArgs)
        Try
            ' Obtener el ID del usuario
            Dim idUsuario As Integer = CInt(InputBox("Ingrese su ID de Usuario:", "Compra de Libro"))

            ' Obtener el ID del libro
            Dim idLibro As Integer = CInt(InputBox("Ingrese el ID del Libro:", "Compra de Libro"))

            ' Obtener la cantidad a comprar
            Dim cantidad As Integer = CInt(InputBox("Ingrese la cantidad a comprar:", "Compra de
Libro"))

            ' Mostrar los métodos de pago disponibles
            MostrarMetodosPago(sender, e)

            ' Obtener el ID del método de pago
            Dim idMetodoPago As Integer = CInt(InputBox("Ingrese el ID del método de pago
seleccionado:", "Compra de Libro"))
```

```
    ' Ejecutar el procedimiento almacenado para comprar el libro
    Dim comando As New SqlCommand("ComprarLibroCliente2", conexion)
    comando.CommandType = CommandType.StoredProcedure

    ' Agregar los parámetros al comando
    comando.Parameters.AddWithValue("@id_usuario", idUsuario)
    comando.Parameters.AddWithValue("@id_libro", idLibro)
    comando.Parameters.AddWithValue("@cantidad", cantidad)
    comando.Parameters.AddWithValue("@id_metodo_pago", idMetodoPago)

    ' Verificar que la conexión esté abierta
    If conexion.State <> ConnectionState.Open Then conexion.Open()

    ' Ejecutar el procedimiento almacenado
    comando.ExecuteNonQuery()

    ' Confirmar que la compra fue realizada
    MessageBox.Show("Compra realizada exitosamente.")



    ' Mostrar los detalles de la compra (Factura)
    MostrarDetallesDeCompra(idUsuario)

Catch ex As Exception
    MessageBox.Show("Error al realizar la compra: " & ex.Message)
Finally
    ' Cerrar la conexión después de la operación
    If conexion.State = ConnectionState.Open Then conexion.Close()
End Try
```

```
    End Sub

    ' Mostrar detalles de la compra (Factura)
    Private Sub MostrarDetallesDeCompra(idUsuario As Integer)
        Try
            ' Obtener el último pedido realizado por el cliente
            Dim query As String = "SELECT TOP 1 p.id_pedido, dp.id_libro, l.titulo, dp.cantidad,
dp.precio_total " &
                                  "FROM DETALLE_PEDIDOS dp " &
                                  "JOIN PEDIDOS p ON dp.id_pedido = p.id_pedido " &
                                  "JOIN LIBROS l ON dp.id_libro = l.id_libro " &
                                  "WHERE p.id_usuario = @id_usuario ORDER BY p.id_pedido DESC"

            ' Ejecutar la consulta
            Dim comando As New SqlCommand(query, conexion)
            comando.Parameters.AddWithValue("@id_usuario", idUsuario)

            ' Llenar un DataTable con los resultados
            Dim adaptador As New SqlDataAdapter(comando)
            Dim dataTable As New DataTable()
            adaptador.Fill(dataTable)

            ' Mostrar los detalles de la compra en un MessageBox
            If dataTable.Rows.Count > 0 Then
                Dim detalle As DataRow = dataTable.Rows(0)
                Dim factura As String = String.Format("Factura de Compra:\n\n" &
                                        "ID Pedido: {0}\n" &
                                        "Libro: {1}\n" &
                                        "Cantidad: {2}\n" &
                                        "Total: {3:C}",
                                        detalle("id_pedido"), detalle("titulo"), detalle("cantidad"),
detalle("precio_total"))
```

```
            MessageBox.Show(factura, "Detalles de Compra", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Else
            MessageBox.Show("No se encontraron detalles de compra.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End If
    Catch ex As Exception
        MessageBox.Show("Error al obtener los detalles de la compra: " & ex.Message)
    End Try
End Sub

' Método para ejecutar comandos SQL con parámetros
Private Sub EjecutarComando(query As String, parametros As Dictionary(Of String, String))
    Try
        VerificarConexion()
        Dim comando As New SqlCommand(query, conexion)
        For Each kvp In parametros
            comando.Parameters.AddWithValue(kvp.Key, kvp.Value)
        Next
        Dim filasAfectadas = comando.ExecuteNonQuery()
        MessageBox.Show($"Operación completada. Filas afectadas: {filasAfectadas}")
    Catch ex As Exception
        MessageBox.Show("Error al ejecutar comando: " & ex.Message)
    End Try
End Sub

' Verificar y abrir la conexión
Private Sub VerificarConexion()
    If conexion.State <> ConnectionState.Open Then conexion.Open()
End Sub
End Class
```

Capturas de pantalla FrmMenu (Menú de Administradores)

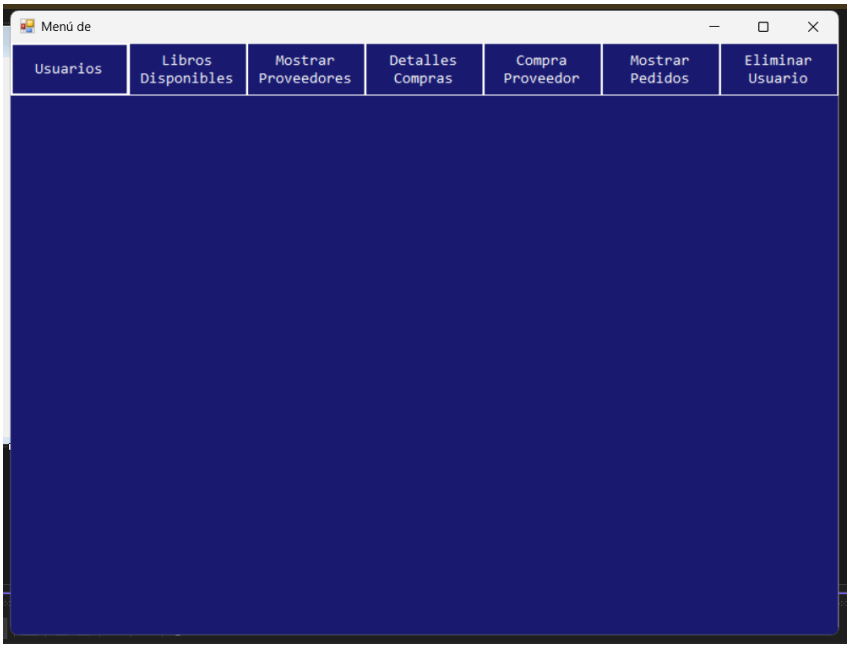


En esta imagen podemos ver la ejecución del Form1.vb en donde se nos muestra el botón que nos permitirá conectarnos a la base de datos Proyecto_Final_BD.

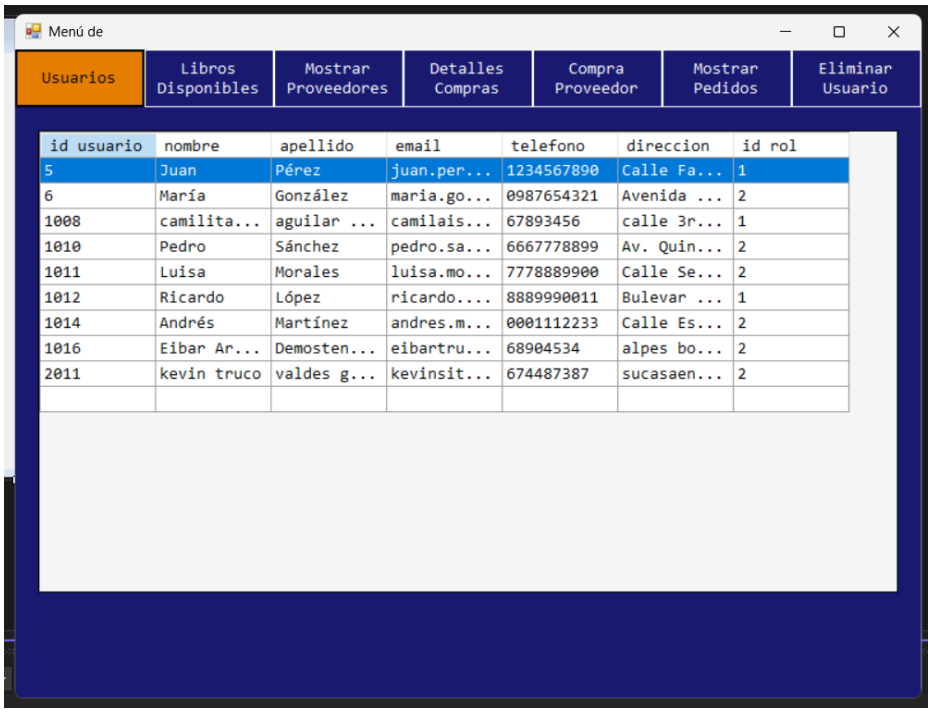
Al momento de hacer click en el botón para conectar a la base de datos, nos solicita un usuario, contraseña, y el id de rol que de acuerdo a la programación establecida:

1 es para administradores y 2 es para clientes. Esto a su vez, determina que menú se abrirá, si el FrmMenu (administradores) o el FrmMenuCliente (clientes).

En este caso usaremos el rol de administradores. (1)

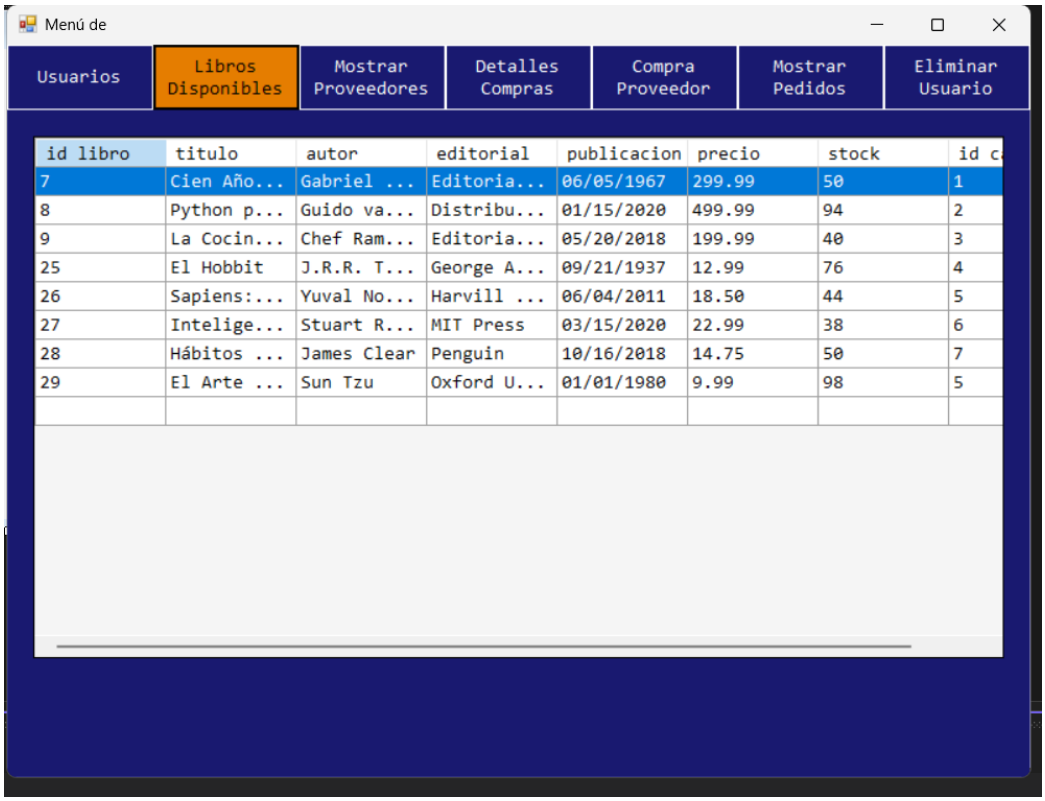


Una vez accedimos a la base de datos con el rol de administradores, se nos presenta el siguiente menú administrativo. Contamos con diversas opciones para manipular y supervisar la información ingresada, actualizada o eliminada de la base de datos.

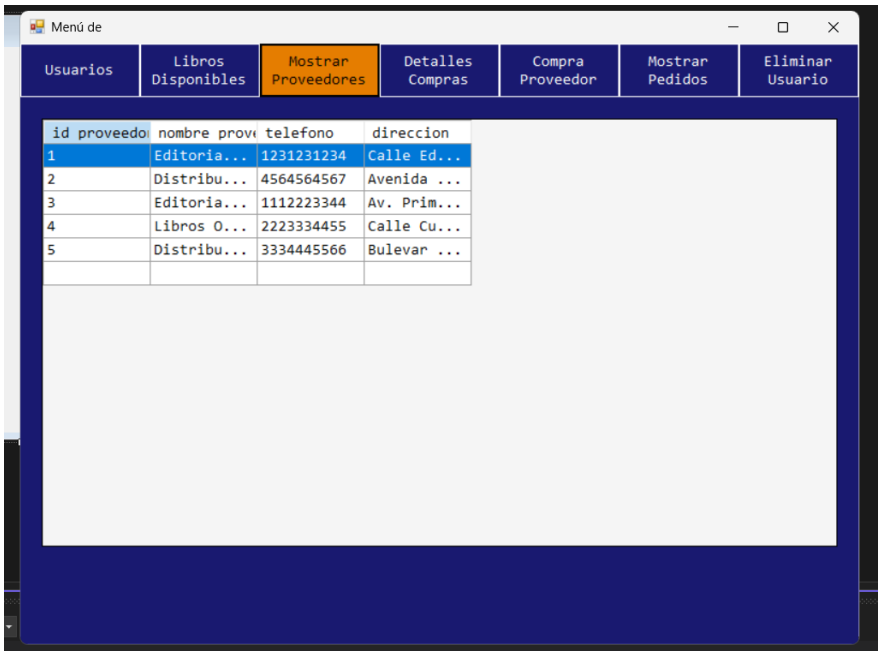


id usuario	nombre	apellido	email	telefono	direccion	id rol
5	Juan	Pérez	juan.per...	1234567890	Calle Fa...	1
6	María	González	maria.go...	0987654321	Avenida ...	2
1008	camilita...	aguilar ...	camilais...	67893456	calle 3r...	1
1010	Pedro	Sánchez	pedro.sa...	6667778899	Av. Quin...	2
1011	Luisa	Morales	luisa.mo...	7778889900	Calle Se...	2
1012	Ricardo	López	ricardo....	8889990011	Bulevar ...	1
1014	Andrés	Martínez	andres.m...	0001112233	Calle Es...	2
1016	Eibar Ar...	Demosten...	eibartru...	68904534	alpes bo...	2
2011	kevin truco	valdes g...	kevinsit...	674487387	sucasaen...	2

Utilizando el botón de usuarios podemos administrar absolutamente todos los usuarios independientemente del tipo de usuario que sean. Podremos ver sus datos como su nombre, apellido, email, teléfono, dirección y el rol al que pertenecen.



Al accionar el botón de libros disponibles se nos mostrará un DataGridView con todos los libros disponibles para la venta sus títulos, el autor, la editorial, su fecha de publicación, precio y la cantidad o stock disponible.

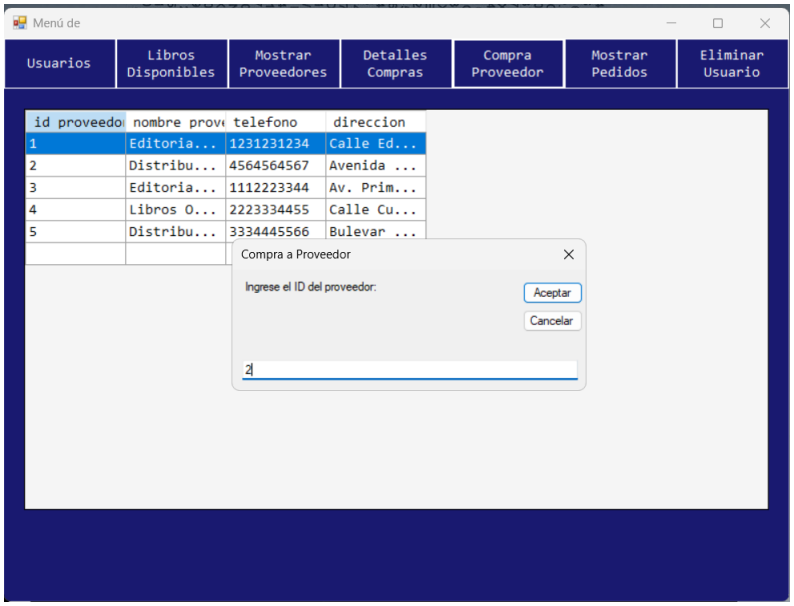


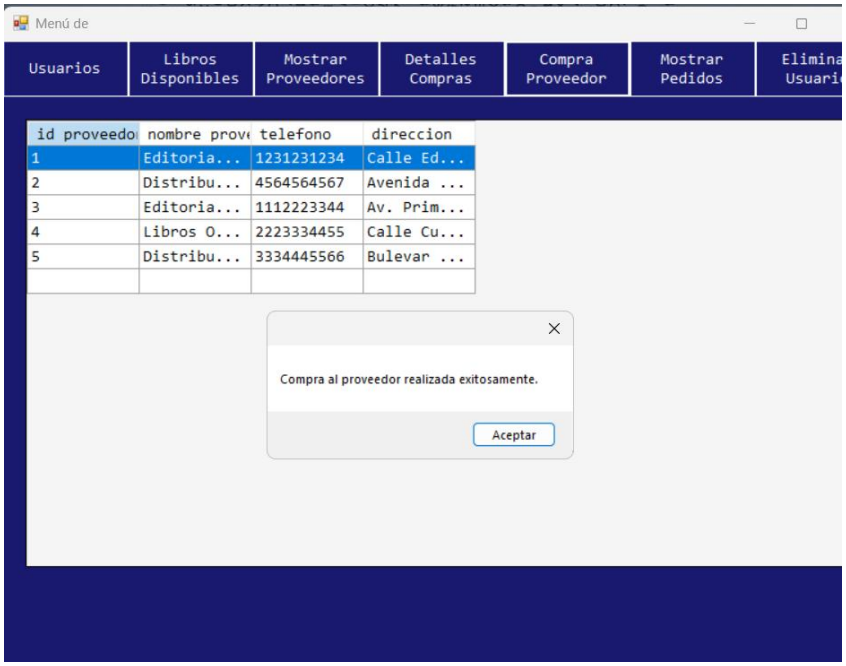
Usando el botón Mostrar Proveedores, podremos ver los proveedores con los que contamos registrados en la base de datos cuyo id nos servirá para realizar las compras a proveedores y llenar nuestro stock para tener más libros.

Realizar compras a proveedores:

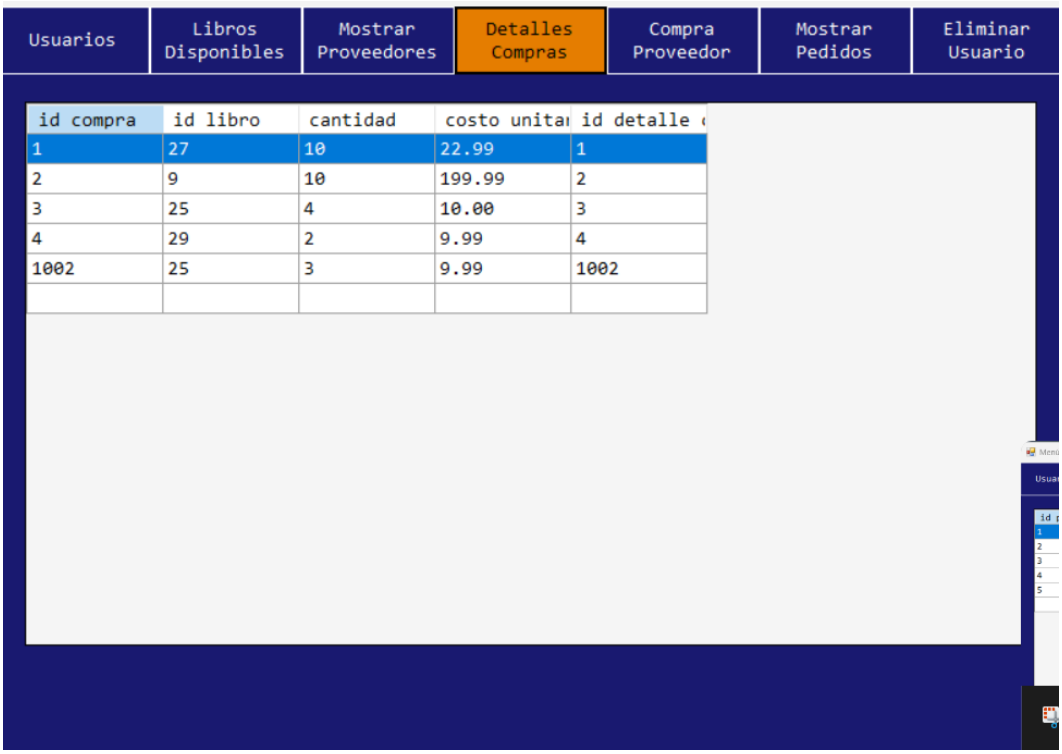
Al momento de realizar compras a proveedores para agrandar el almacenamiento de libros de nuestro stock, se nos solicitarán algunos datos específicos y necesarios:

Ingresar el ID del proveedor (proporcionado en el botón Mostrar Proveedores), también debemos ingresar el ID del libro que compraremos para nuestra Librería, la cantidad y el costo unitario.



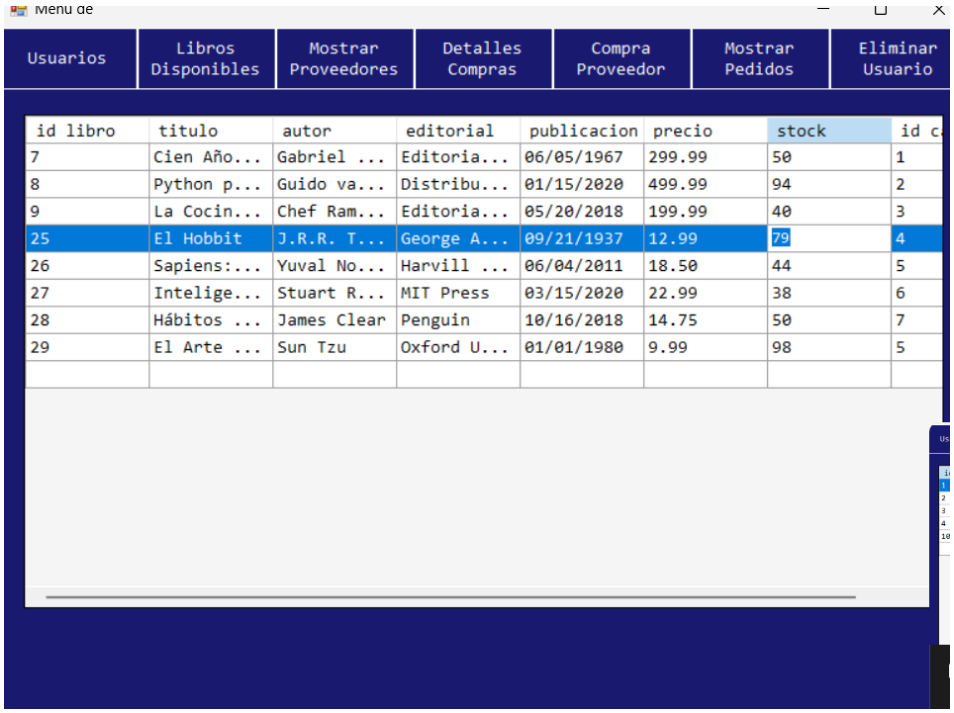


“Imagen al momento de realizar la compra exitosamente”



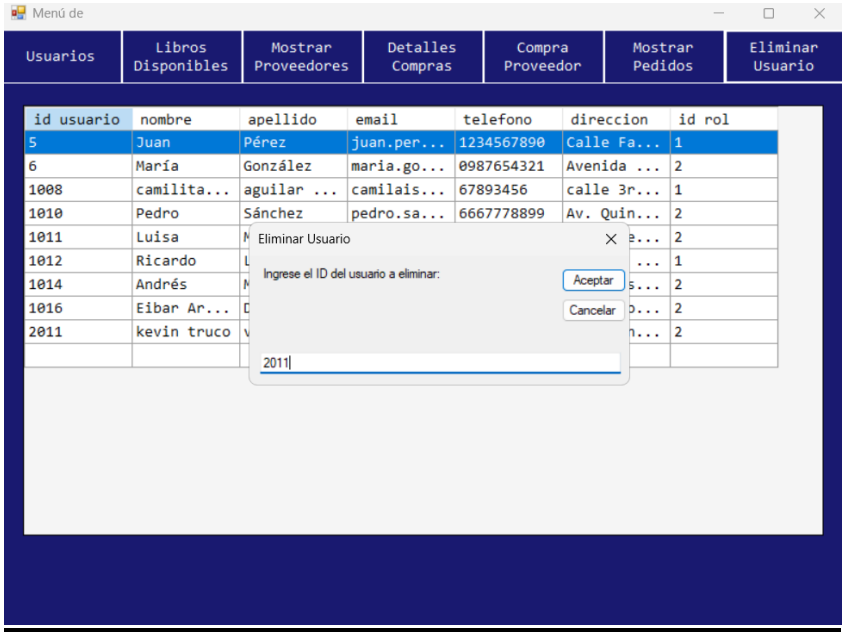
Aquí podemos ver que al momento de realizar una compra a cualquier proveedor exitosamente, utilizando el botón de Detalles Compras podemos observar que la compra se guarda a modo de registro o factura en la base de datos. En este apartado compramos el libro con el id 25 El hobbit.

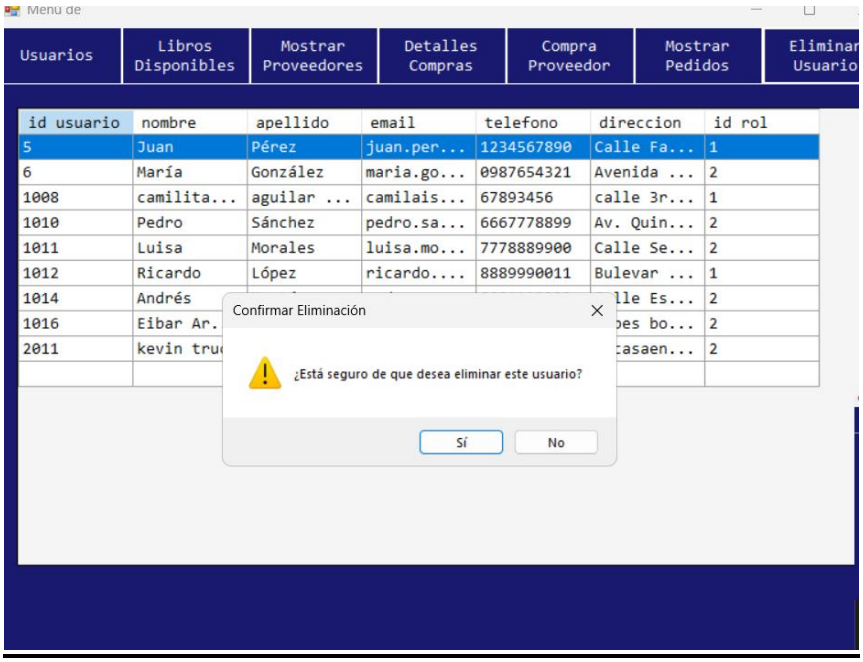
Y en la siguiente imagen podemos observar que nuestro stock inicial ha sido modificado sumando los 3 libros que compramos a los proveedores haciendo que nuestro stock pase de 76 a 79.

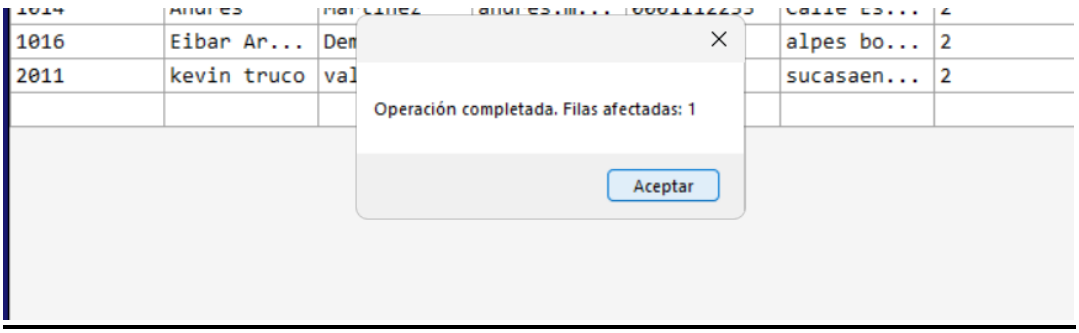


Eliminar Usuarios de la base de datos:

Usando el botón Eliminar Usuario se nos solicitará ingresar el ID del usuario al que deseamos eliminar y se nos pedirá una confirmación.

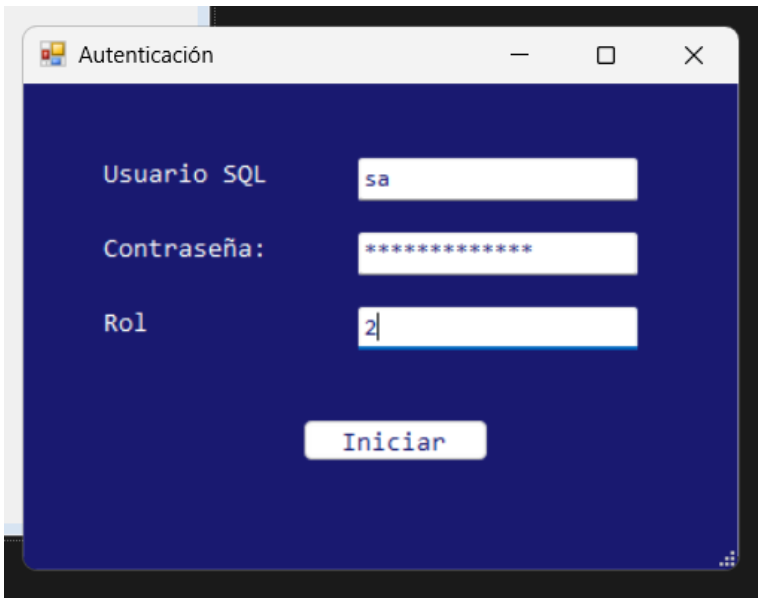




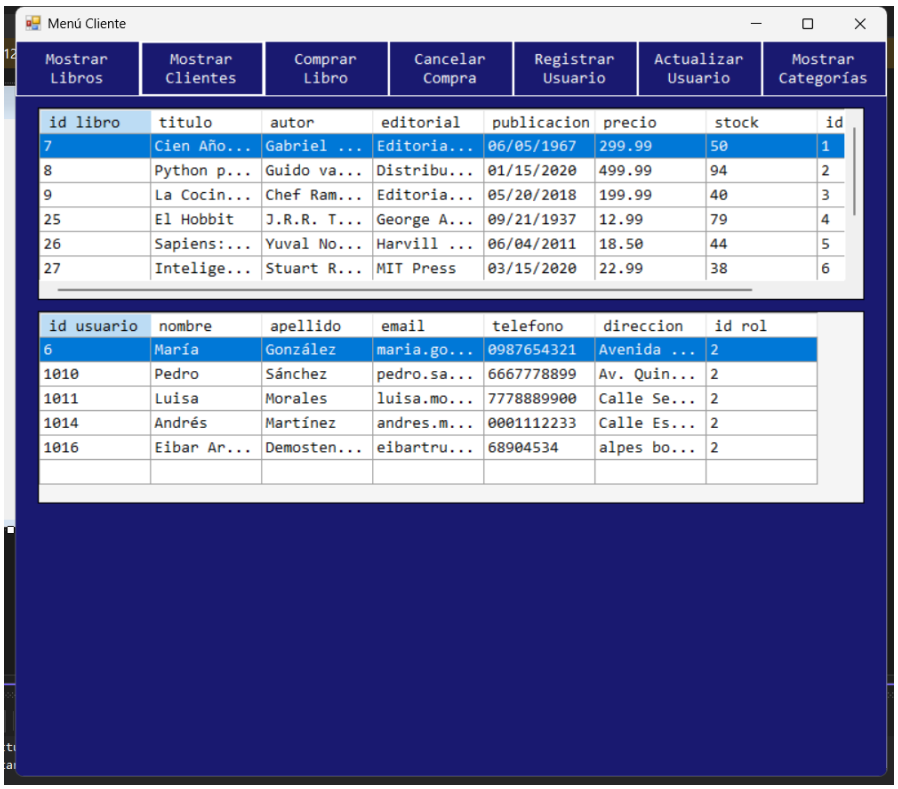


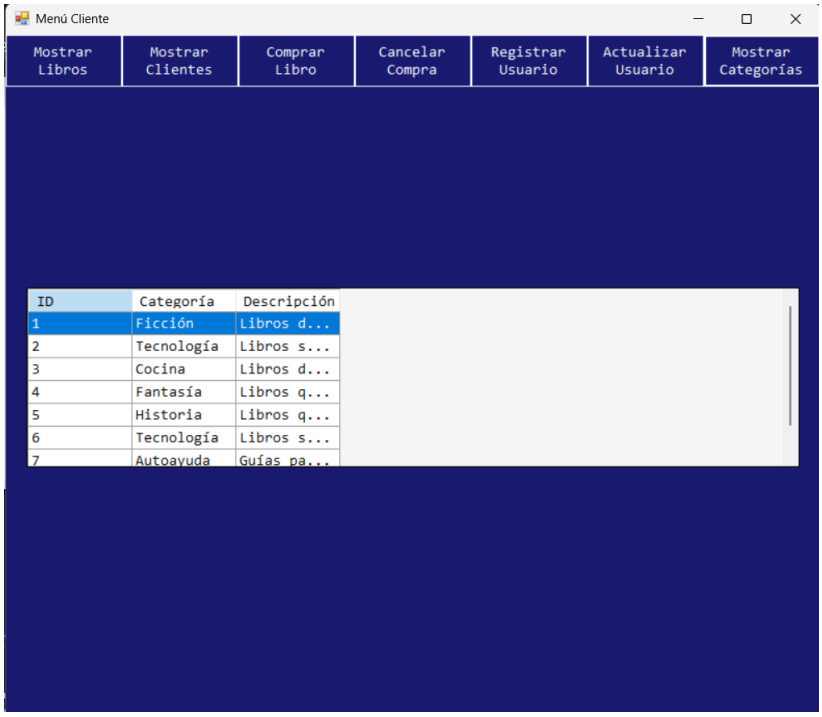
Capturas de Pantalla FrmMenuClientes(Clientes)

Al momento de iniciar sesión e ingresar el rol 2 se accede al menú de clientes.



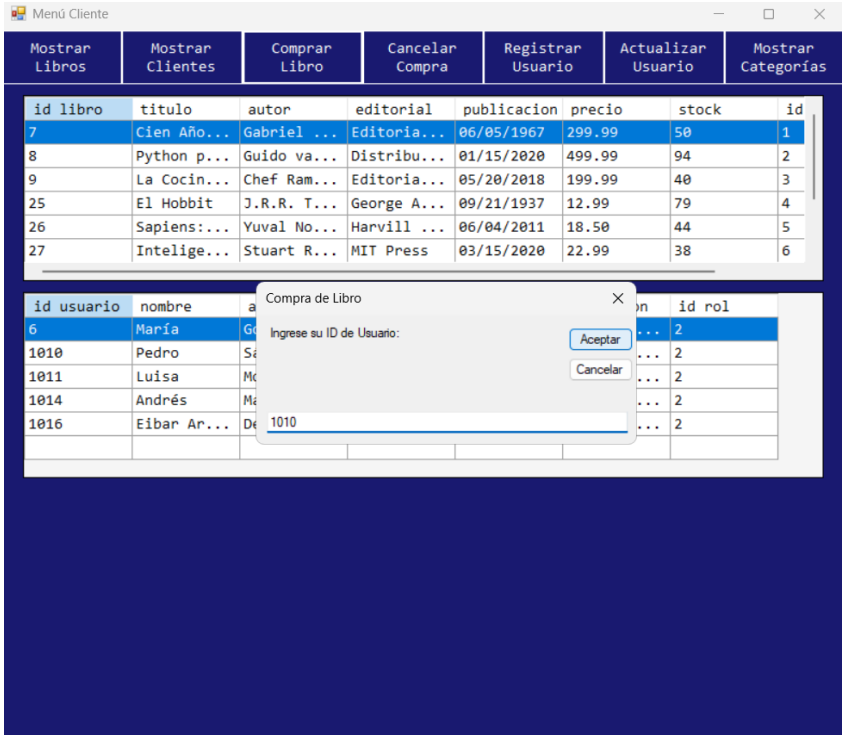
Usando los botones Mostrar Libros y Mostrar Clientes podemos ver dichos datos ingresados en 2 DataGridView que están almacenados en la base de datos.



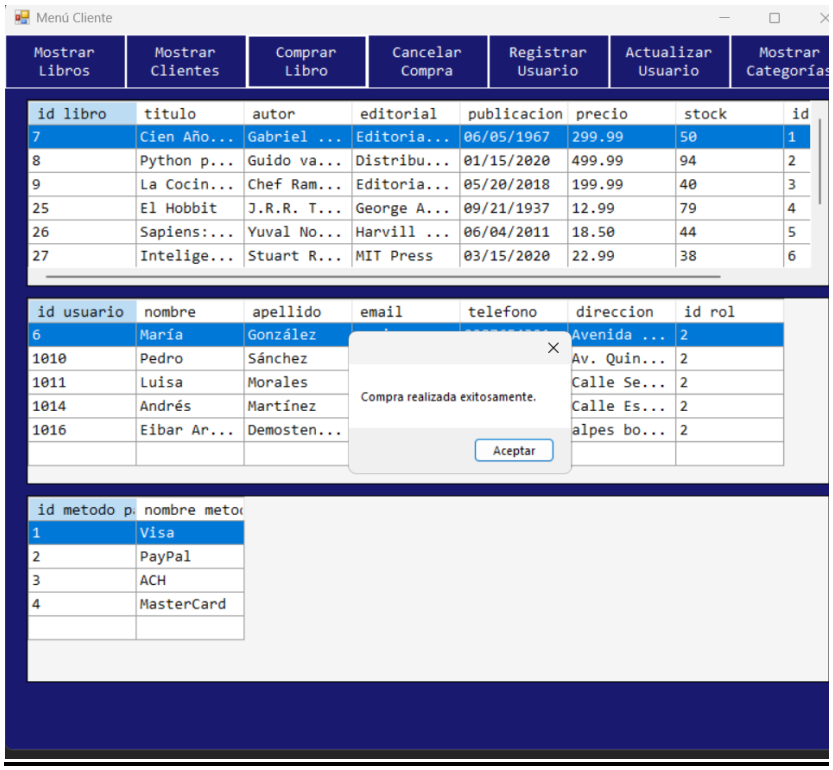


Con el botón Mostrar Categorías se muestran las temáticas de los libros que se venden en la librería.

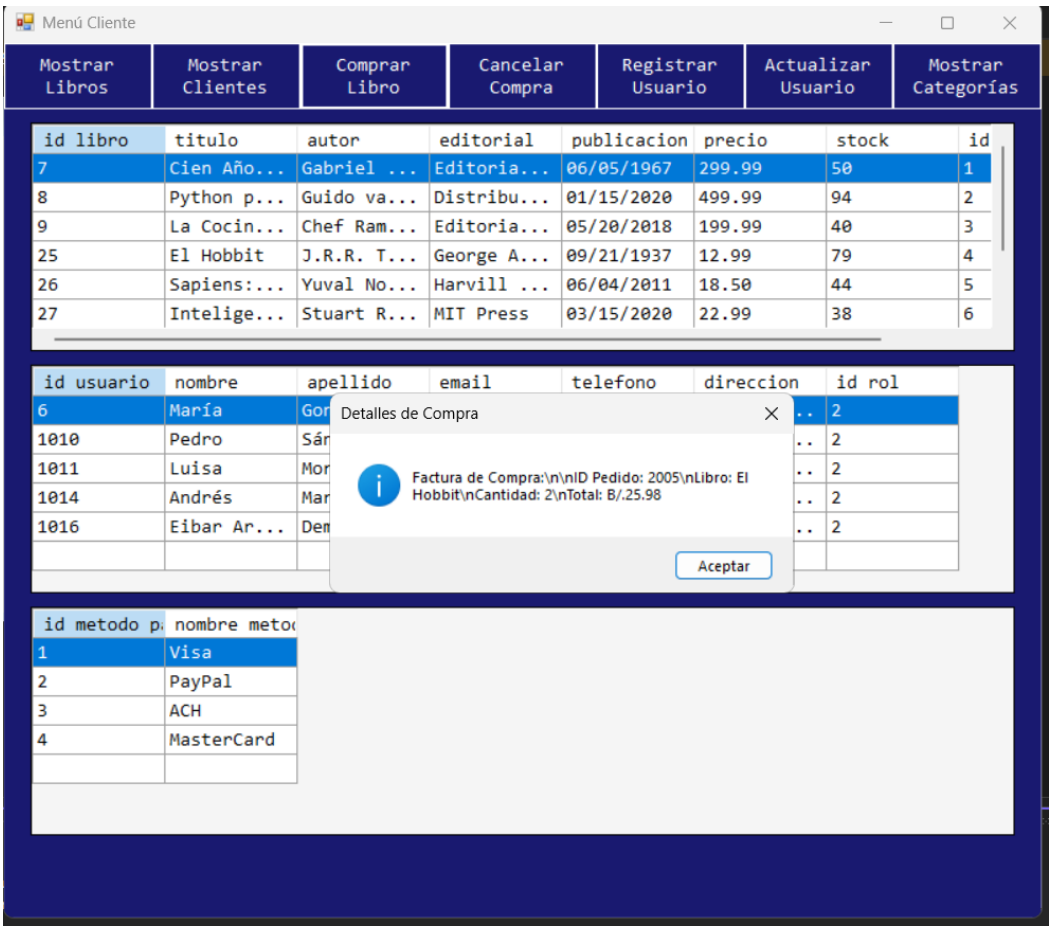
Estos resultados se muestran en el 2do DataGridView.



Para la compra de libros desde el FrmMenuCliente se utiliza la opción Comprar Libro y se nos solicita el ID del usuario, el ID del libro, la cantidad a comprar y a su vez se despliega un tercer DataGridView para mostrar los métodos de pago disponibles.



Al completar la compra exitosamente muestra un mensaje confirmando la transacción. Y se muestra una notificación a modo de factura presentando los datos.



TRANSACCIONES Y PROCEDIMIENTOS UTILIZADOS

```
--COMPRA DE LIBROS A PROVEEDORES--

--ESTE PROCEDIMIENTO:
-- 1.Registra una compra en la tabla COMPRAS_PROVEEDOR
-- 2.Agrega los detalles de la compra en DETALLE_COMPRAS
-- 3.Actualiza el stock en la tabla LIBROS

CREATE PROCEDURE ComprarLibroProveedor
        @id_proveedor INT,
        @id_libro INT,
        @cantidad INT,
        @costo_unitario DECIMAL (10,2)

AS

BEGIN
        BEGIN TRANSACTION;

        --Insertar una nueva compra
        DECLARE @id_compra INT;
        INSERT INTO COMPRAS_PROVEEDOR (id_proveedor, fecha_compra, monto_total)
        VALUES (@id_proveedor, GETDATE(), @cantidad * @costo_unitario);
        SET @id_compra = SCOPE_IDENTITY();

        --Insert el detalle de la compra
        INSERT INTO DETALLE_COMPRAS (id_compra, id_libro, cantidad, costo_unitario)
        VALUES (@id_compra, @id_libro, @cantidad, @costo_unitario);

        --Incrementar el stock del libro
```

```
    UPDATE LIBROS
    SET stock = stock + @cantidad
    WHERE id_libro = @id_libro;

    COMMIT TRANSACTION;
    PRINT 'Compra al proveedor registrada exitosamente.';
END;
```

```
USE [Proyecto_Final_BD]
GO

/****** Object:  StoredProcedure [dbo].[ComprarLibroCliente2]    Script Date: 12/05/2024 11:20:11 p. m. ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

--COMPRAS DE LIBROS POR CLIENTES--

-- Este procedimiento:
-- 1. Crea un pedido en la tabla PEDIDOS
-- 2. Agrega los detalles del pedido en la tabla DETALLE_PEDIDOS
-- 3. Actualiza el stock en la tabla LIBROS
-- 4. Asigna un estado al pedido dependiendo del método de pago

ALTER PROCEDURE [dbo].[ComprarLibroCliente2]
    @id_usuario INT,
    @id_metodo_pago INT,
    @id_libro INT,
    @cantidad INT
AS
BEGIN
    BEGIN TRANSACTION;

    -- Verificar si hay suficiente stock disponible
    IF EXISTS (SELECT 1 FROM LIBROS WHERE id_libro = @id_libro AND stock >= @cantidad)
    BEGIN
        -- Insertar un nuevo pedido
```

```
DECLARE @id_pedido INT;
DECLARE @estado_pedido VARCHAR(20);

-- Asignar el estado del pedido basado en el método de pago
IF @id_metodo_pago IN (1, 2, 4) -- Visa, Paypal, MaterCard
BEGIN
    SET @estado_pedido = 'Completada';
END
ELSE IF @id_metodo_pago = 3 -- ACH
BEGIN
    SET @estado_pedido = 'Pendiente';
END

-- Insertar en la tabla PEDIDOS con el estado determinado
INSERT INTO PEDIDOS (id_usuario, id_metodo_pago, fecha_pedido, estado_pedido)
VALUES (@id_usuario, @id_metodo_pago, GETDATE(), @estado_pedido);
SET @id_pedido = SCOPE_IDENTITY();  -- Obtener el ID del pedido recién insertado

-- Insertar el detalle del pedido en la tabla DETALLE_PEDIDOS
DECLARE @precio_unitario DECIMAL(10,2);
SELECT @precio_unitario = precio FROM LIBROS WHERE id_libro = @id_libro;

INSERT INTO DETALLE_PEDIDOS (id_pedido, id_libro, cantidad, precio_total)
VALUES (@id_pedido, @id_libro, @cantidad, @precio_unitario * @cantidad);

-- Actualizar el stock del libro
UPDATE LIBROS
SET stock = stock - @cantidad
WHERE id_libro = @id_libro;

COMMIT TRANSACTION;  -- Finaliza la transacción si todo ha ido bien
```

```
        PRINT 'Compra registrada exitosamente.';
    END
    ELSE
    BEGIN
        ROLLBACK TRANSACTION;  -- Revertir la transacción si no hay suficiente stock
        PRINT 'Stock insuficiente para realizar la compra.';
    END
END;
GO
```

```
USE Proyecto_Final_BD;
GO

CREATE PROCEDURE CancelarCompra
    @id_pedido INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Verificar si el pedido tiene estado 'Pendiente'
        IF EXISTS (
            SELECT 1
            FROM PEDIDOS
            WHERE id_pedido = @id_pedido AND estado_pedido = 'Pendiente'
        )
        BEGIN
            -- Revertir el stock del libro
            UPDATE LIBROS
            SET stock = stock + dp.cantidad
            FROM DETALLE_PEDIDOS dp
            WHERE dp.id_pedido = @id_pedido
              AND LIBROS.id_libro = dp.id_libro;

            -- Eliminar el detalle del pedido
            DELETE FROM DETALLE_PEDIDOS
            WHERE id_pedido = @id_pedido;

            -- Eliminar el pedido
            DELETE FROM PEDIDOS
            WHERE id_pedido = @id_pedido;
```

```
        COMMIT TRANSACTION;
        PRINT 'Compra cancelada exitosamente.';
    END
    ELSE
    BEGIN
        ROLLBACK TRANSACTION;
        PRINT 'El pedido no existe o no tiene estado Pendiente.';
    END
  END TRY
  BEGIN CATCH
    ROLLBACK TRANSACTION;
    PRINT 'Ocurrió un error al cancelar la compra: ' + ERROR_MESSAGE();
  END CATCH
END;
GO
```

Conclusión

El Sistema de Gestión de Librería es un ejemplo sólido de la aplicación de los conceptos aprendidos en la materia Base de Datos 1, combinando diseño relacional y programación visual para crear una herramienta funcional y educativa.

Refleja el esfuerzo colaborativo del equipo, así como nuestro compromiso con la calidad y la usabilidad. Este proyecto no solo cumple con los objetivos académicos, sino que también establece una base sólida para futuras implementaciones en entornos reales.

Resultados y Logros

Integración funcional exitosa:

La base de datos y la interfaz gráfica interactúan de manera fluida, con cada operación reflejada inmediatamente en la base de datos.

Sistema escalable:

El diseño modular permite futuras expansiones, como agregar nuevas funcionalidades o roles.

Enriquecimiento académico:

Cada miembro del equipo adquirió experiencia práctica en:

Normalización de bases de datos.

Procedimientos almacenados y transacciones SQL.

Desarrollo de interfaces gráficas conectadas a bases de datos.

Retroalimentación del sistema:

Se proporciona al usuario información clara sobre el estado de sus acciones (compras, cancelaciones, registros).

Limitaciones y Mejoras Futuras

Límites en la gestión de inventario:

Actualmente, el sistema carece de alertas automáticas para niveles bajos de stock.

Falta de reportes:

Sería útil implementar reportes gráficos para analizar ventas y compras.

Seguridad adicional:

Se puede fortalecer la autenticación mediante encriptación de contraseñas y roles más dinámicos.