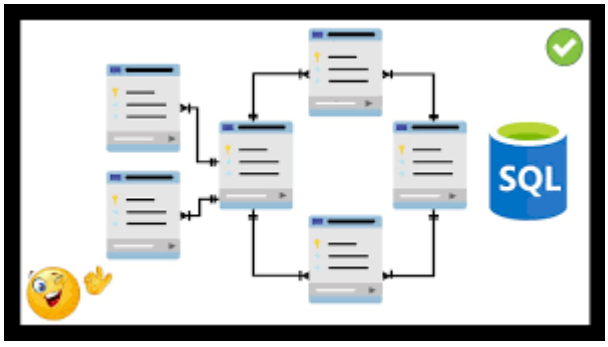
Universidad Autónoma de San Luis Potosí.
Facultad de ingeniería.
Área de ciencias de la computación.



BD CARPINTERIA

Manual del Programador.

Descripción Breve:

Realizar una base de datos para una carpintería, con disparadores y restricciones.

Integrantes:

- **César Ulises Medellín Quintanilla.**
- **Raúl Ruperto Martínez.**
- **Alejandro Zapata Delgado.**

Materia: **Administración de Bases de Datos.**

Semestre: **2022-2023/II.**

Número de Grupo: **231301**

Fecha: 24/05/2023

Índice Pagina

Objetivos:

Consiste en el desarrollo de una aplicación con conexión a un sistema manejador de base de datos donde reside la base de datos. Donde se utilizará lo siguiente:

- SCHEMA, donde residan las tablas de la base de datos.
- RULES, validen los dominios de los atributos de la base de datos, con restricción numérica y de cadena.
- TRIGGER, para automatizar las funciones operativas del sistema.

Requerimientos técnicos del sistema:

Lenguaje de programación utilizado:

- "C#" es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Sistema operativo utilizado:

- **Windows 10 y 11** (Utilizado): Sistema operativo desarrollado por Microsoft como parte de la familia de sistemas operativos Windows NT.

- **Linux**: Sistema operativo semejante a Unix, de código abierto y desarrollado por una comunidad, para computadoras, servidores, mainframes, dispositivos móviles y dispositivos embebidos.

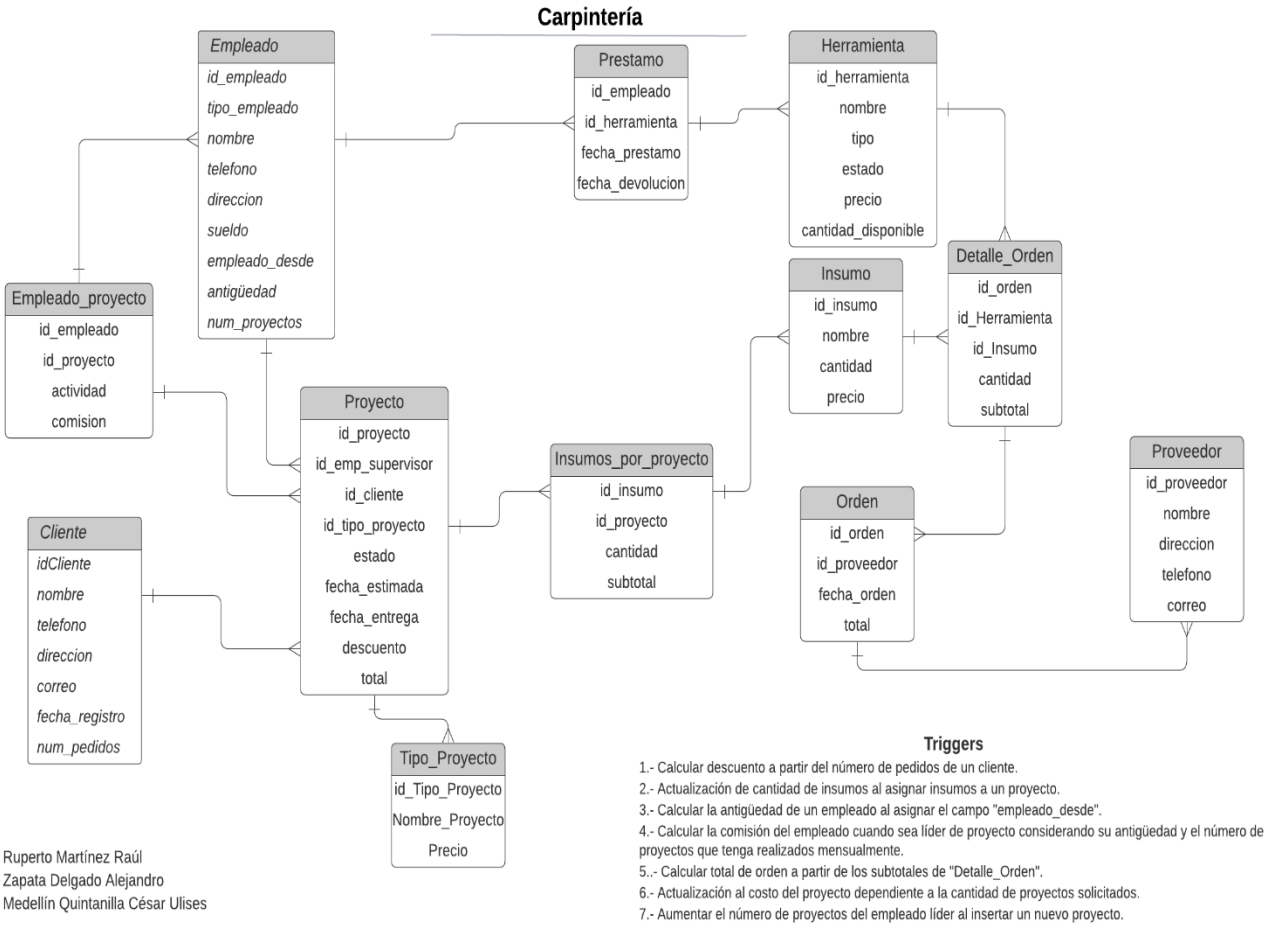
- **Mac OS**: Serie de sistemas operativos gráficos desarrollados y comercializados por Apple.

Programas utilizados:

- **C#,** Es un entorno de desarrollo de software para sistemas operativos.
- **SQL Server Management Studio,** El lenguaje de consulta estructurada (SQL) es un lenguaje de consulta popular que se usa con frecuencia en todos los tipos de aplicaciones.

Nota: Son programas recomendados, si desea utilizar otro de su mayor facilidad es recomendable (link de descarga de los programas en referencias).

Diagrama:



Ruperto Martínez Raúl
Zapata Delgado Alejandro
Medellín Quintanilla César Ulises

Descripción de las tablas de la base de datos:

Se utiliza el Lenguaje de Consulta Estructurada (SQL) es un lenguaje gestor para el manejo de la información en las bases de datos relacionales. Este tipo de lenguaje de programación permite comunicarse con la base de datos y realizar operaciones de acceso y manipulación de la información almacenada.

Realizar tareas como las de seleccionar, insertar, actualizar y eliminar datos, así como también las de unir y consultar información de diferentes tablas en una base de datos.

Definición y descripción de variables del sistema:

Librerías:

- **using System:** Brinda algunas clases útiles como Consola o funciones/métodos como WriteLine. El espacio de nombres ProjectName es algo que identifica y encapsula su código dentro de ese espacio de nombres.
- **using System.Collections.Generic:** Contiene interfaces y clases que definen colecciones genéricas, lo que permite a los usuarios crear colecciones fuertemente tipadas que brindan una mejor seguridad y rendimiento de tipos que las colecciones no genéricas fuertemente tipadas.
- **using System.ComponentModel:** Proporciona clases que se utilizan para implementar el comportamiento en tiempo de ejecución y en tiempo de diseño de componentes y controles.
- **using System.Data:** Controla el comportamiento de creación de transacciones mientras se ejecuta una consulta o un comando de la base de datos.
- **using System.Data.SqlClient:** Representa una instrucción Transact-SQL o un procedimiento almacenado para ejecutar en una base de datos de SQL Server .
- using System.Drawing: La manipulación de imágenes, como generadores de códigos QR y representación de texto.
- **using System.Linq:** Ofrece una sintaxis común para consultar cualquier tipo de fuente de datos; por ejemplo, puede consultar un documento XML de la misma manera que consulta una base de datos SQL, un conjunto de datos ADO.NET, una colección en memoria o cualquier otra fuente de datos remota o local a la que haya elegido conectarse y acceder mediante utilizando LINQ.
- **using System.Text:** Contiene clases que representan varias conversiones y codificación de caracteres, además de proporcionar clases auxiliares para manipular y formatear objetos String.
- **using System.Threading.Tasks:** Proporciona un conjunto de subprocesos que se pueden usar para ejecutar tareas, publicar elementos de trabajo, procesar E/S asincrónicas, esperar en nombre de otros subprocesos y procesar temporizadores.
- **using System.Windows.Forms:** Contiene clases para crear aplicaciones basadas en Windows que aprovechan al máximo las ricas características de la interfaz de usuario disponibles en el sistema operativo Microsoft Windows .
- **using static System.Windows.Forms.VisualStyles.VisualStyleElement:** Identifica un elemento de control o interfaz de usuario (UI) que se dibuja con estilos visuales.
- **using Carpinteria_SQLServer:** Referencia a la base de datos.

Clases:

En las siguientes clases tenemos la funcionalidad de consulta, inserción, modificación y eliminación a la base de datos.

- class Cliente.
- class Empelado.
- class Proveedor.
- class Herramienta.
- class Insumo.
- class Tipo_Proyecto.

En las siguientes clases de igual manera tenemos las funcionalidades de consulta, modificación y eliminación a la base de datos. Pero con la diferencio que tenemos llaves foráneas, por lo cual algunas clases tienes la función de inserción, modificación y eliminación en cascada.

- class Proyecto.
- class Préstamo.
- class Empelado_Proyecto.
- class InsumoProyecto.
- class Orden.
- class Detalle_Orden.
- class NuevaOrden.

La siguiente clase tenemos la funcionalidad de llenado de textbox, así como su actualización y limpia.

- class EventosForms.

La clase menú esta diseñada para llamar a las clases anteriormente mencionada de forma ordenada y rápida.

- class Menú.

Variables:

La siguiente variable se utiliza para la conexión con la base de datos, por lo tanto, es utilizada en todas las clases:

- ✓ SqlConnection conexion = new SqlConnection("Server=CESARMEDELLIN\\SQLEXPRESS;Database=Carpinteria;Integrated Security=true");

Variable fecha, para controlar la salida del dateTimePicker:

- ✓ string fecha;

Variable para los comandos del SQL, por ejemplo:

- ✓ string query = string.Concat("SELECT * FROM Persona.Empleado");
- ✓ string consulta = "INSERT INTO Persona.Empleado" +
  - "(tipo_empleado,nombre,telefono,direccion,sueldo,empleado_desde,antiguedad,num_proyectos)"+
  - "VALUES ('"+ comboBox2.SelectedItem +"','"+ textBox1.Text +"','"+ textBox2.Text +"','"+ textBox3.Text +"','"+ textBox4.Text +"','"+ fecha +"','"+ antiguedad +"','"+ textBox5.Text +")";
- ✓ string consulta = "DELETE FROM Persona.Empleado WHERE idEmpleado =" + dataGridView1.CurrentRow.Cells[0].Value;

Variables para obtener los id actuales de nuestra tabla de la base de datos que estemos actualmente y banderas para un mayor control:

- ✓ String idempl;
- ✓ String idproy;
- ✓ String idviejoempl;
- ✓ String idviejoproy;
- ✓ bool bandlider = false;
- ✓ int cantidadActual;
- ✓ int idProyectoActual;
- ✓ int insumoActual;

Variables para obtener una copia de nuestra tabla de la base de datos.

- ✓ DataTable empleados = new DataTable();
- ✓ DataTable proyectos = new DataTable();

Variable Textbox se usa para mostrar o editar texto son formato.

- ✓ TextBox textBox = control as TextBox;

La variable delegada es un tipo que representa referencias a métodos con una lista de parámetros determinados y un tipo de valor devuelto:

- ✓ delegadoActualizaDatos delegado;

Funciones:

Las funciones son un concepto fundamental en programación, una función es una secuencia de comandos que realizan un cómputo.

En C# a las funciones o procedimientos se le conoce con el nombre de métodos. En el ámbito de la programación, una función es un tipo de subalgoritmo, es el termino para describir una secuencia de órdenes que hacen una tarea especifica de una aplicación mas grande.

- Primero se verá algunas funciones que se repiten en todas las clases de nuestra aplicación, excepto en la class EventosForms:

```
//Conexion a la base de datos.
public int connectaBD()
{
    try
    {
    conexion.Open();
    muestra();
    conexion.Close();
    return 0;
    }
    catch (Exception ex)
    {
    conexion.Close();
    return -1;
    }
}

//Consulta SQL para mostrar información de una tabla de la base de datos.
public void muestra()
{
    string query = string.Concat("SELECT * FROM Proyecto.Tipo_Proyecto");

    SqlCommand cmd = new SqlCommand(query, conexion);
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataTable registro = new DataTable();
    adapter.Fill(registro);
    tablaTipo.DataSource = null;
    tablaTipo.DataSource = registro;
    tablaTipo.AutoSizeColumnsMode =
    DataGridViewAutoSizeColumnsMode.AllCells;

}
```

```
//Consulta SQL para insertar filas en una tabla de la base de datos.
public void insertaRegistro()
{
    try
    {
    conexion.Open();
    string consulta = "INSERT INTO Proyecto.Tipo_Proyecto" +
        "(nombre_proyecto,precio)" +
        " VALUES ('" + textBox1.Text + "'," + textBox2.Text +")";
    SqlCommand comd = new SqlCommand(consulta, conexion);
    comd.ExecuteNonQuery();
    conexion.Close();
    }
    catch (Exception ex)
    {
    conexion.Close();
    MessageBox.Show("Proyecto ya existente, favor de cambiar nombre del
    proyecto");
    }
}

//Consulta SQL para borrar filas de una tabla de la base de datos.
public void eliminarRegistro()
{
    try
    {
    conexion.Open();
    string consulta = "DELETE FROM Proyecto.Tipo_Proyecto WHERE
    idTipo_proyecto = " + tablaTipo.CurrentRow.Cells[0].Value;
    SqlCommand comd = new SqlCommand(consulta, conexion);
    comd.ExecuteNonQuery();
    conexion.Close();
    }
    catch (Exception ex)
    {
    conexion.Close();
    MessageBox.Show("Error de conexion, " + ex.Message);
    }
}
```

```
//Consulta SQL para actualizar información sobre una tabla de la base de datos.
private void btnModificarHerramienta_Click(object sender, EventArgs e)
{
    try
    {
        int id = (int)tablaHerramientas.SelectedRows[0].Cells["idHerramienta"].Value;
        conexion.Open();
        string query = "UPDATE Proyecto.Herramienta SET nombre = @nombre, tipo = @tipo, estado = @estado, cantidad_disponible = @cantidad_disponible , precio = @precio WHERE idHerramienta = @id";
        SqlCommand comando = new SqlCommand(query,conexion);
        EventosForms.parametrosForm(comando, this);
        comando.Parameters.Add(new SqlParameter("id",id));
        comando.ExecuteNonQuery();
        EventosForms.limpiaTextbox(this);
        conexion.Close();
        muestraDatosTabla();
    }catch(Exception ex) { throw new Exception(ex.Message); }
}


//Función para inicializar componentes de nuestra clase, al momento de ejecutar nuestra clase esta función es la primera en ser llamada.
public Proyecto()
{
    InitializeComponent();
    textBox4.Text = "0";
    textBox4.Enabled = false;
    textBox5.Enabled = false;
    comboBox4.Items.Add("Comienzo");
    comboBox4.Items.Add("Pendiente");
    comboBox4.SelectedIndex = 0;
    comboBox4.Enabled = false;
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker2.Enabled = false;
    connectaBD();
    LiderBusca();
    ClienteBusca();
    TipoProyectoBusca();
}
```

```
//Función para el botón de insertar de nuestro formulario, al hacer clic en el botón se llamará la función siguiente:
private void button1_Click(object sender, EventArgs e)
{
    fecha = dateTimePicker1.Value.Year + "-" + dateTimePicker1.Value.Month + "-" + dateTimePicker1.Value.Day;
    insertaRegistro();
    connectaBD();
    textBox5.Clear();
    comboBox1.SelectedIndex = -1;
    comboBox2.SelectedIndex = -1;
    comboBox3.SelectedIndex = -1;
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker2.Value = DateTime.Now;
}

//Función para el botón de modificar de nuestro formulario, al hacer clic en el botón se llamará la función siguiente:
private void button2_Click(object sender, EventArgs e)
{
    fecha = dateTimePicker1.Value.Year + "-" + dateTimePicker1.Value.Month + "-" + dateTimePicker1.Value.Day;
    fecha2 = dateTimePicker2.Value.Year + "-" + dateTimePicker2.Value.Month + "-" + dateTimePicker2.Value.Day;
    modificarRegistro();
    connectaBD();
    textBox5.Clear();
    textBox4.Text = "0";
    comboBox1.Enabled = true;
    comboBox2.Enabled = true;
    comboBox2.SelectedIndex = -1;
    comboBox1.SelectedIndex = -1;
    comboBox3.SelectedIndex = -1;
    comboBox4.SelectedIndex = 0;
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker1.Enabled = true;
    dateTimePicker2.Enabled = false;
    comboBox3 .Enabled = true;
    comboBox4.Enabled = false;
}
```

```
//Función para el botón de eliminar de nuestro formulario, al hacer clic en el botón se llamará la función siguiente:
private void button3_Click(object sender, EventArgs e)
{
    eliminarRegistroEP();
    eliminarRegistro();
    connectaBD();
    textBox5.Clear();
    textBox4.Text = "0";
    comboBox2.SelectedIndex = -1;
    comboBox1.SelectedIndex= -1;
    comboBox3.SelectedIndex= -1;
    dateTimePicker1.Value = DateTime.Now;
    dateTimePicker2.Value = DateTime.Now;
}


//Función DataGridWiew, al hacer clic sobre los datos del datagridview se autocompletará los textbox con dichos datos.
private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (dataGridView1.CurrentRow.Cells[5].Value.ToString() == "Pendiente")
    {
    string client = "";
    string emple;
    string tipop;

    for (int i = 0; i <= cliente.Rows.Count - 1; i++)
    {
        client = cliente.Rows[i]["nombre"].ToString() + "-" +
        cliente.Rows[i]["correo"].ToString();
        if (client == dataGridView1.CurrentRow.Cells[2].Value.ToString())
        {
        comboBox2.SelectedIndex = i;
        idclienteAnt = cliente.Rows[i]["idCliente"].ToString();
        }
    }
    for (int i = 0; i <= empleado.Rows.Count - 1; i++)
    {
        emple = empleado.Rows[i]["nombre"].ToString() + "-" +
        empleado.Rows[i]["antiguedad"].ToString();
        if (emple == dataGridView1.CurrentRow.Cells[1].Value.ToString())
        {
        comboBox1.SelectedIndex = i;
        idmpleadoAnt = empleado.Rows[i]["idEmpleado"].ToString();
```

```
            }
        }
        for (int i = 0; i <= tipo_proyecto.Rows.Count - 1; i++)
        {
            tipop = tipo_proyecto.Rows[i]["idProyecto"].ToString() + "-" +
            tipo_proyecto.Rows[i]["nombre_proyecto"].ToString();
            if (tipop == dataGridView1.CurrentRow.Cells[0].Value.ToString())
            {
            comboBox3.SelectedIndex = i;
            }
        }

        textBox4.Text = dataGridView1.CurrentRow.Cells[6].Value.ToString();
        textBox5.Text = dataGridView1.CurrentRow.Cells[7].Value.ToString();
        dateTimePicker1.Text =
        dataGridView1.CurrentRow.Cells[4].Value.ToString();
        dateTimePicker2.Value = DateTime.Now;
        comboBox1.Enabled = true;
        comboBox3.Enabled = true;
        comboBox4.Enabled = true;
        }
        else
        {
        MessageBox.Show("Proyecto terminado!!!!");
        }


    }


    //Segundo ejemplo de la función DataGridWiew, al hacer clic sobre los datos del
datagridview se autocompletará los textbox con dichos datos.
    private void dataGridView1_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
    {
        textBox1.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        textBox2.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
        textBox3.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
        textBox4.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();
        textBox5.Text = dataGridView1.CurrentRow.Cells[6].Value.ToString();
        dateTimePicker1.Text =
        dataGridView1.CurrentRow.Cells[5].Value.ToString();
        dateTimePicker1.Enabled = true;
    }
```

Clase Empleado:

```
//Función del botón líder del proyecto, donde accedemos a la clase Empleado Proyecto.
private void button4_Click(object sender, EventArgs e)
{
    Empleado_Proyecto empP = new Empleado_Proyecto();
    empP.Show();
    this.Close();
}
```

Clase Empleado_Proyecto:

```
//Función EmpleadoBuscar, donde hacemos una consulta SQL a la tabla Empleado donde obtenemos los datos de un empleado líder especifico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
public void EmpleadoBusca()
{
    try
    {
    conexion.Open();
    if (bandlider == false)
    {
        string consulta = string.Concat("SELECT *" +
        " FROM Persona.Empleado WHERE tipo_empleado = 'Empleado'");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(empleados);
    }
    else
    {
        string consulta = string.Concat("SELECT *" +
        " FROM Persona.Empleado WHERE tipo_empleado = 'Lider'");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(empleados);
    }
    conexion.Close();

    }
    catch (Exception ex)
    {
    conexion.Close();
    MessageBox.Show("No se encuentran empleados");
    }
```

```
        for (int i = 0; i <= empleados.Rows.Count - 1; i++)
        comboBox1.Items.Add(empleados.Rows[i]["nombre"].ToString());

    }

    //Función ProyectoBuscar, donde hacemos una consulta SQL a la tabla Proyecto y Tipo_Proyecto donde obtenemos los datos de un proyecto específico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
    public void ProyectoBusca()
    {
        try
        {
        conexion.Open();
        string consulta = string.Concat("SELECT p.idProyecto,
            t.nombre_proyecto" +
        " FROM Proyecto.Proyecto p " +
        "INNER JOIN Proyecto.Tipo_Proyecto t ON p.idTipo_proyecto =
            t.idTipo_proyecto ");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(proyectos);
        conexion.Close();

        }
        catch (Exception ex)
        {
        conexion.Close();
        MessageBox.Show("No se encuentran proyectos");
        }

        for (int i = 0; i <= proyectos.Rows.Count - 1; i++)
        comboBox2.Items.Add(proyectos.Rows[i]["nombre_proyecto"].ToString());

    }


    //La siguiente función nos ayuda a identificar que empleado se esta seleccionando en el comboBox y se guarda esa información.
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox1.SelectedIndex != -1)
        idempl =
            empleados.Rows[comboBox1.SelectedIndex]["idEmpleado"].ToString();
    }
```

```
//La siguiente función nos ayuda a identificar que proyecto se está seleccionando en el comboBox y se guarda esa información.
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    if (comboBox2.SelectedIndex != -1)
    idproy = proyectos.Rows[comboBox2.SelectedIndex]["idProyecto"].ToString();
}


//Función cerrar ventana, se cierra la ventana actual.
private void Empleado_Proyecto_FormClosed(object sender, FormClosedEventArgs e)
{
    Empleado emp = new Empleado();
    emp.Show();
}
```

Clase EventoForms:

```
//Función llenar textbox, se autocompletará los textbox con los datos seleccionados.

public static void llenaTextBox(object sender, EventArgs e)
{
        DataGridView tabla = (DataGridView)sender;
        Form pantalla = tabla.Parent as Form;
        if (tabla.SelectedRows.Count > 0)
        {
            DataGridViewRow filaSeleccionada = tabla.SelectedRows[0];
            foreach (DataGridViewCell celda in filaSeleccionada.Cells)
            {
                    if (celda.ColumnIndex != 0)
                    {

                            Control textBox =
                            pantalla.Controls.OfType<Control>().FirstOrDefault(c
                            => c.Tag?.ToString() ==
                            celda.OwningColumn.Name);
                            textBox.Text = celda.Value.ToString();
                    }
                }
        }
}
```

```
//Función parámetros, accedemos a las variables con los datos seleccionados con el textbox.
public static void parametrosForm(SqlCommand comando, Form pantalla)
{
    foreach (Control control in pantalla.Controls)
    {
        if (control is TextBox)
        {
            comando.Parameters.Add(new SqlParameter(control.Tag.ToString(), control.Text));
        }
    }
}

//Función limpia textbox, limpia de datos que contengan los textbox.
public static void limpiaTextbox(Form pantalla)
{
    foreach (Control control in pantalla.Controls)
    {
        if (control is TextBox)
        {
            TextBox textBox = control as TextBox;
            textBox.Clear();
        }
    }
}
```

Clase Herramienta:

```
//Función del botón Préstamo, donde accedemos a la clase préstamo.
private void button1_Click (object sender, EventArgs e)
{
        Prestamo press = new Prestamo();
        press.Show();
        this.Close();
}
```

Clase Préstamo:

```
    //Función HerramientaBusca, donde hacemos una consulta SQL a la tabla Herramienta donde obtenemos los datos de una herramienta específica. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
    public void HerramientaBusca()
    {
        try
        {
            conexion.Open();
            string consulta = string.Concat("SELECT *" +
            " FROM Proyecto.Herramienta");
            SqlCommand cmd = new SqlCommand(consulta, conexion);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(herramienta);
            conexion.Close();

        }
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show("Herramienta no encontrada");
        }

        for (int i = 0; i <= herramienta.Rows.Count - 1; i++)
    comboBox1.Items.Add(herramienta.Rows[i]["nombre"].ToString());

    }

    //Función EmpleadoBusca, donde hacemos una consulta SQL a la tabla Empleado donde obtenemos los datos de un empleado específico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
    public void EmpleadoBusca()
    {
        try
        {
            conexion.Open();
            string consulta = string.Concat("SELECT *" +
            " FROM Persona.Empleado");
            SqlCommand cmd = new SqlCommand(consulta, conexion);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            adapter.Fill(empleado);
            conexion.Close();

        }
        catch (Exception ex)
        {
```

```
            conexion.Close();
            MessageBox.Show("Empleado no encontrado");
        }

        for (int i = 0; i <= empleado.Rows.Count - 1; i++)
            comboBox2.Items.Add(empleado.Rows[i]["nombre"].ToString());

    }


    //La siguiente función nos ayuda a identificar qué herramienta se está seleccionando en el comboBox y se guarda esa información.
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox1.SelectedIndex != -1)
        idherr = herramienta.Rows[comboBox1.SelectedIndex]["idHerramienta"].ToString();
    }

    //La siguiente función nos ayuda a identificar qué empleado se está seleccionando en el comboBox y se guarda esa información.
    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox2.SelectedIndex != -1)
        {
        idempl = empleado.Rows[comboBox2.SelectedIndex]["idEmpleado"].ToString();
        comboBox1.Enabled = true;
        dateTimePicker1.Visible = true;
        dateTimePicker1.Enabled = false;
        }

    }

    //Función cerrar ventana, se cierra la ventana actual.
    private void Prestamo_FormClosed(object sender, FormClosedEventArgs e)
    {
        Herramienta herr = new Herramienta();
        herr.Show();
    }
```

```
    //Consulta SQL para actualizar información sobre la tabla préstamo de la base de datos.
    public void EntregaRegistro()
    {
        try
        {
        conexion.Open();
        if (dataGridView1.CurrentRow != null)
        {
            string consulta = "UPDATE Proyecto.Prestamo " +
            "SET fecha_devolucion = '" + fecha2 + "'" +
            "WHERE id_empleado = " + idempl + " AND " +
            "id_herramienta = " + idant + " AND " +
            "fecha_prestamo = '" + dataGridView1.CurrentRow.Cells[2].Value + "'";

            SqlCommand comd = new SqlCommand(consulta, conexion);
            comd.ExecuteNonQuery();
        }
        else
        {
            MessageBox.Show("No hay datos que modificar");
        }
        conexion.Close();

        }
        catch (Exception ex)
        {
        conexion.Close();
        MessageBox.Show("Error de conexion, " + ex.Message);
        }
    }


    //Función del botón Entregar.
    private void button4_Click(object sender, EventArgs e)
    {
        fecha2 = dateTimePicker1.Value.Year + "-" + dateTimePicker1.Value.Month + "-" + dateTimePicker1.Value.Day;
        EntregaRegistro();
        connectaBD();
        comboBox2.Enabled = true;
        comboBox1.Enabled = false;
        dateTimePicker1.Visible = false;
        dateTimePicker2.Visible = false;
        dateTimePicker1.Value = DateTime.Now;
```

```
        comboBox2.SelectedIndex = -1;
        comboBox1.SelectedIndex = -1;

    }
```

Clase Menu:

```
//Función del botón Menú, donde accedemos a la clase Herramienta.
private void button4_Click(object sender, EventArgs e)
{
    Herramienta herr = new Herramienta();
    herr.Show();
}

//Función del botón Menú, donde accedemos a la clase Cliente.
private void button1_Click(object sender, EventArgs e)
{
    Cliente client = new Cliente();
    client.Show();
}

//Función del botón Menú, donde accedemos a la clase Empleado.
private void button2_Click(object sender, EventArgs e)
{
    Empleado empl = new Empleado();
    empl.Show();
}

//Función del botón Menú, donde accedemos a la clase Insumo.
private void button3_Click(object sender, EventArgs e)
{
    Insumo insumo = new Insumo();
    insumo.Show();
}

//Función del botón Menú, donde accedemos a la clase Proyecto.
private void button5_Click(object sender, EventArgs e)
{
    Proyecto proyecto = new Proyecto();
    proyecto.Show();
}
```

```
//Función del botón Menú, donde accedemos a la clase Proveedor.
private void button6_Click(object sender, EventArgs e)
{
    Proveedor proveedor = new Proveedor();
    proveedor.Show();
}

//Función del botón Menú, donde accedemos a la clase Orden.
private void button7_Click(object sender, EventArgs e)
{
    Orden orden = new Orden();
    orden.Show();
}
```

Clase Proyecto:

```
//Función LiderBusca, donde hacemos una consulta SQL a la tabla Empleado con la condición de que sea líder de proyecto donde obtenemos los datos de un empleado específico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
public void LiderBusca()
{
    try
    {
        conexion.Open();
        string consulta = string.Concat("SELECT *" +
        " FROM Persona.Empleado" +
        " WHERE tipo_empleado = 'Lider'");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(empleado);
        conexion.Close();

    }
    catch (Exception ex)
    {
        conexion.Close();
        MessageBox.Show("Empleado no encontrado");
    }

    for(int i = 0; i <= empleado.Rows.Count-1; i++)
        comboBox1.Items.Add(empleado.Rows[i]["nombre"].ToString());

}
```

```
//Función ClienteBusca, donde hacemos una consulta SQL a la tabla Cliente donde obtenemos los datos de un cliente específico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
public void ClienteBusca()
{
    try
    {
        conexion.Open();
        string consulta = string.Concat("SELECT *" +
        " FROM Persona.Cliente");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(cliente);
        conexion.Close();
    }
    catch (Exception ex)
    {
        conexion.Close();
        MessageBox.Show("Cliente no encontrado");
    }

    for (int i = 0; i <= cliente.Rows.Count - 1; i++)
        comboBox2.Items.Add(cliente.Rows[i]["nombre"].ToString());

}

//Función TipoProyectoBusca, donde hacemos una consulta SQL a la tabla Proyecto y Tipo_Proyecto donde obtenemos los datos de un proyecto específico. Y dichos datos se guardan en una variable para tener mayor control y accesibilidad a ellos.
public void TipoProyectoBusca()
{
    try
    {
        conexion.Open();
        string consulta = string.Concat("SELECT p.idProyecto,
        t.idTipo_proyecto, t.nombre_proyecto, t.precio" +
        " FROM Proyecto.Tipo_Proyecto t" +
        " LEFT JOIN Proyecto.Proyecto p ON t.idTipo_proyecto =
        p.idTipo_proyecto");
        SqlCommand cmd = new SqlCommand(consulta, conexion);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(tipo_proyecto);
        conexion.Close();
    }
```

```
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show("Proyecto no encontrado");
        }

        for (int i = 0; i <= tipo_proyecto.Rows.Count - 1; i++)
            comboBox3.Items.Add(tipo_proyecto.Rows[i]["nombre_proyecto"].T
            oString());

    }

    //Función, en donde se realiza una consulta SQL para actualizar información sobre
la tabla proyecto de la base de datos.
    public void EntregaRegistro()
    {
        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.Proyecto " +
            "SET fecha_entrega = '" + fecha2 + "', estado = 'terminado'
            WHERE idProyecto = " +
            dataGridView1.CurrentRow.Cells[0].Value;
            SqlCommand comd = new SqlCommand(consulta, conexion);
            comd.ExecuteNonQuery();
            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show("La informacion se esta utilizando en la tabla
            Empleado Proyecto");
        }
    }

    // Función, en donde se realiza una consulta SQL para actualizar información sobre
la tabla proyecto de la base de datos, en donde se suma el total del proyecto con el
subtotal de los insumos agregados.
    public void TotalProyecto()
    {
        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.Proyecto" +
```

```
                    " SET Total = Total + (SELECT ISNULL(SUM(subtotal),0) FROM Proyecto.InsumoProyecto WHERE idProyecto = "+
                    dataGridView1.CurrentRow.Cells[0].Value.ToString() + ")" +
                    " FROM Proyecto.Proyecto INNER JOIN Proyecto.InsumoProyecto i" +
                    " ON i.idProyecto = Proyecto.idProyecto AND Proyecto.idProyecto = " + dataGridView1.CurrentRow.Cells[0].Value.ToString() +"";
                    SqlCommand comd = new SqlCommand(consulta, conexion);
                    comd.ExecuteNonQuery();
                    conexion.Close();
            }
            catch (Exception ex)
            {
                    conexion.Close();
                    MessageBox.Show("No hay insumos utilizados en este proyecto");
            }
        }

    //La siguiente función nos ayuda a identificar qué empleado se está seleccionando en el comboBox y se guarda esa información.
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        if(comboBox1.SelectedIndex != -1)
        idmpleado = empleado.Rows[comboBox1.SelectedIndex]["idEmpleado"].ToString();
    }

    //La siguiente función nos ayuda a identificar qué cliente se está seleccionando en el comboBox y se guarda esa información.
    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox2.SelectedIndex != -1)
        idcliente = cliente.Rows[comboBox2.SelectedIndex]["idCliente"].ToString();
    }

    //La siguiente función nos ayuda a identificar qué tipo proyecto se está seleccionando en el comboBox y se guarda esa información.
    private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBox3.SelectedIndex != -1)
        {
                idtipo_proyecto = tipo_proyecto.Rows[comboBox3.SelectedIndex]["idTipo_proyecto"].
                ToString();
```

```
                textBox5.Text = tipo_proyecto.Rows[comboBox3.SelectedIndex]["precio"].ToString();
            }
        }

    //Función del botón Entregar.
        private void button4_Click(object sender, EventArgs e)
        {
            fecha2 = dateTimePicker2.Value.Year + "-" + dateTimePicker2.Value.Month + "-" + dateTimePicker2.Value.Day;
            EntregaRegistro();
            TotalProyecto();
            connectaBD();
            textBox5.Clear();
            textBox4.Text = "0";
            comboBox1.Enabled = true;
            comboBox2.Enabled = true;
            comboBox3.Enabled = true;
            comboBox2.SelectedIndex = -1;
            dateTimePicker1.Value = DateTime.Now;
            dateTimePicker1.Enabled = true;
            dateTimePicker2.Enabled = false;
        }

    //Función delegado, hace referencia a un tipo de valor devuelto y accedemos a la clase InsumoProyecto.
        private void button5_Click(object sender, EventArgs e)
        {
            delegadoActualizaDatos delgado = new delegadoActualizaDatos(connectaBD);
            InsumosProyecto insproy = new InsumosProyecto(delgado);
            insproy.Show();
        }


    //Función del botón nuevo proyecto, donde accedemos a la clase Tipo_Proyecto.
        private void button6_Click(object sender, EventArgs e)
        {
            Tipo_Proyecto tipo_p = new Tipo_Proyecto();
            tipo_p.Show();
            this.Close();
        }
```

Clase InsumoProyecto:

```
//Función, en donde se llamada a realizar una consulta SQL.
private void InsumosProyecto_Load(object sender, EventArgs e)
{
    muestraDatosTabla();
    muestraDatosResto();
    tablaInsumosProyecto.SelectionChanged += llenaTextBox;
}


//Función, donde se llama llamar la función delegado.
private void button1_Click(object sender, EventArgs e)
{
    insertaProyectoInsumo();
    muestraDatosTabla();
    muestraDatosResto();
    EventosForms.limpiaTextbox(this);

    delegado();
}


// Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla insumo de la base de datos, en donde se resta la cantidad de insumos.
private void restaCantidad(int idInsumo, int idProyecto, int diferencia)
{
    try
    {
        string texto2 = textBox1.Text;

        conexion.Open();
        string query = "UPDATE Proyecto.Insumo " +
            "SET cantidad = cantidad - @diferencia " +

            "WHERE idInsumo = @idInsumo";
        SqlCommand comando = new SqlCommand(query, conexion);

        comando.Parameters.Add(new SqlParameter("idInsumo",
        idInsumo));
        comando.Parameters.Add(new SqlParameter("diferencia",
        diferencia));

        comando.ExecuteNonQuery();
        conexion.Close();
    }
```

```
        catch (Exception ex)
        {
            throw new Exception(ex.Message);
        }
    }
```

```
    // Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla insumo de la base de datos, en donde se suma la cantidad de insumos.
    private void sumaCantidad(int idInsumo, int idProyecto, int diferencia)
    {
        try
        {
            string texto2 = textBox1.Text;
            diferencia = diferencia * -1;
            conexion.Open();
            string query = "UPDATE Proyecto.Insumo " +
            "SET cantidad = cantidad + @diferencia "  +
            "WHERE idInsumo = @idInsumo";
            SqlCommand comando = new SqlCommand(query, conexion);
            comando.Parameters.Add(new SqlParameter("idInsumo", idInsumo));
            comando.Parameters.Add(new SqlParameter("diferencia", diferencia));
            comando.ExecuteNonQuery();
            conexion.Close();
        }
        catch (Exception ex)
        {
            Throw new Exception(ex.Message);
        }
    }
```

```
    // Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla proyecto de la base de datos, en donde se suma el total del proyecto con el subtotal de los insumos agregados.
    public void CambioProyect(int idInsumo, int idProyecto, int diferencia)
    {
        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.Proyecto " +
            "SET  Total = Total - ((SELECT precio FROM Proyecto.Insumo WHERE idInsumo = " + idInsumo + ") * " + cantidadActual + ")" +
            " WHERE idProyecto = " + idProyectoActual + "";
```

```
                SqlCommand comd = new SqlCommand(consulta, conexion);
                comd.ExecuteNonQuery();
                string consulta2 = "UPDATE Proyecto.Proyecto " +
                "SET  Total = Total + ((SELECT precio FROM Proyecto.Insumo
                WHERE idInsumo = " + idInsumo + ") * " + cantidadActual + ")" +
                " WHERE idProyecto = " + idProyecto + "";
                SqlCommand comd2 = new SqlCommand(consulta2, conexion);
                comd2.ExecuteNonQuery();
                conexion.Close();
            }
            catch (Exception ex)
            {
                conexion.Close();
                MessageBox.Show("La informacion se esta utilizando en la tabla
                Empleado Proyecto");
            }
    }


    // Función, en donde se realiza una consulta SQL para actualizar información sobre
la tabla proyecto de la base de datos, en donde se suma el total del proyecto con el
subtotal de los insumos agregados.
    public void CambioInsumo(int idInsumo, int idProyecto, int diferencia)
    {
        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.Proyecto " +
            "SET  Total = Total - ((SELECT subtotal FROM
            Proyecto.InsumoProyecto WHERE idInsumo = " + insumoActual +
            "))" +
            " WHERE idProyecto = " + idProyectoActual + "";
            SqlCommand comd = new SqlCommand(consulta, conexion);
            comd.ExecuteNonQuery();
            string consulta2 = "UPDATE Proyecto.Insumo " +
            "SET  cantidad = cantidad + " + cantidadActual + "" +
            " WHERE idInsumo = " + insumoActual + "";
            SqlCommand comd2 = new SqlCommand(consulta2, conexion);
            comd2.ExecuteNonQuery();
            string consulta3 = "UPDATE Proyecto.Insumo " +
            "SET  cantidad = cantidad - " + cantidadActual + "" +
            " WHERE idInsumo = " + idInsumo + "";
            SqlCommand comd3 = new SqlCommand(consulta3, conexion);
            comd3.ExecuteNonQuery();
            string consulta4 = "UPDATE Proyecto.InsumoProyecto " +
```

```
                "SET  idInsumo = "+idInsumo+", cantidad = " + cantidadActual + ",
                subtotal = ((SELECT precio FROM Proyecto.Insumo WHERE
                idInsumo = " + idInsumo +")) * "+cantidadActual+"" +
                " WHERE idInsumo = " + insumoActual + "";
                SqlCommand comd4 = new SqlCommand(consulta4, conexion);
                comd4.ExecuteNonQuery();
                string consulta5 = "UPDATE Proyecto.Proyecto " +
                "SET  Total = Total + ((SELECT precio FROM Proyecto.Insumo
                WHERE idInsumo = " + idInsumo + ")) * " + cantidadActual + "" +
                " WHERE idProyecto = " + idProyecto + "";
                SqlCommand comd5 = new SqlCommand(consulta5, conexion);
                comd5.ExecuteNonQuery();
                conexion.Close();
            }
            catch (Exception ex)
            {
                conexion.Close();
                MessageBox.Show("La informacion se esta utilizando en la tabla
                Empleado Proyecto");
            }
        }


    // Función del botón modificar, donde llamamos a las funciones de suma, resta de
cantidad de insumo y modificación del total de proyecto.
    //Asu vez verificamos si al modificar se esta disminuyendo o aumentando el pedido
de insumo (variable cantidad).
    private void button2_Click(object sender, EventArgs e)
    {
        DataRowView proyectoSel =
        (DataRowView)comboProyectos.SelectedItem;
        int idProyecto = (int)proyectoSel["idProyecto"];


        DataRowView insumoSel = (DataRowView)comboInsumos.SelectedItem;
        int idInsumo = (int)insumoSel["idInsumo"];

        int cantidad = Int32.Parse(textBox3.Text);
        int act = cantidadActual;
        modificar(idInsumo, idProyecto);
        int diferencia = cantidad - cantidadActual;
        if (idInsumo != insumoActual)
            CambioInsumo(idInsumo, idProyecto, diferencia);
        if(idProyecto != idProyectoActual)
            CambioProyect(idInsumo, idProyecto, diferencia);
        if (diferencia < 0)
```

```
            {
                sumaCantidad(idInsumo,idProyecto,diferencia);
                restaProyecto(idInsumo, idProyecto, diferencia);
            }
        else
            {
                restaCantidad(idInsumo,idProyecto, diferencia);
                sumaProyecto(idInsumo, idProyecto, diferencia);
            }

        conexion.Close();
        muestraDatosTabla();
        muestraDatosResto();
        EventosForms.limpiaTextbox(this);
        delegado();
    }

    // Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla proyecto de la base de datos, en donde se suma el total del proyecto con el subtotal.
    public void sumaProyecto(int idInsumo, int idProyecto, int diferencia)
    {
        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.Proyecto " +
            "SET  Total = Total + ((SELECT precio FROM Proyecto.Insumo WHERE idInsumo = "+idInsumo+") * "+diferencia+")" +
            " WHERE idProyecto = "+idProyecto+"";
            SqlCommand comd = new SqlCommand(consulta, conexion);
            comd.ExecuteNonQuery();
            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show("La informacion se esta utilizando en la tabla Empleado Proyecto");
        }

        try
        {
            conexion.Open();
            string consulta = "UPDATE Proyecto.InsumoProyecto " +
```

```
                "SET  subtotal = subtotal + ((SELECT precio FROM
                Proyecto.Insumo WHERE idInsumo = " + idInsumo + ") * " +
                diferencia + ")" +
                " WHERE idProyecto = " + idProyecto + "";
                SqlCommand comd = new SqlCommand(consulta, conexion);
                comd.ExecuteNonQuery();
                conexion.Close();
        }
        catch (Exception ex)
        {
                conexion.Close();
                MessageBox.Show("La informacion se esta utilizando en la tabla
                Empleado Proyecto");
        }
}


    // Función, en donde se realiza una consulta SQL para actualizar información sobre
la tabla proyecto de la base de datos, en donde se resta el total del proyecto con el
subtotal.
    public void restaProyecto(int idInsumo, int idProyecto, int diferencia)
    {
        diferencia = diferencia * -1;
        try
        {
                conexion.Open();
                string consulta = "UPDATE Proyecto.Proyecto " +
                "SET  Total = Total - ((SELECT precio FROM Proyecto.Insumo
                WHERE idInsumo = " + idInsumo + ") * " + diferencia + ")" +
                " WHERE idProyecto = " + idProyecto + "";
                SqlCommand comd = new SqlCommand(consulta, conexion);
                comd.ExecuteNonQuery();
                conexion.Close();
        }
        catch (Exception ex)
        {
                conexion.Close();
                MessageBox.Show("La informacion se esta utilizando en la tabla
                Empleado Proyecto");
        }
        try
        {
                conexion.Open();
                        string consulta = "UPDATE Proyecto.InsumoProyecto " +
```

```
            "SET  subtotal = subtotal - ((SELECT precio FROM
            Proyecto.Insumo WHERE idInsumo = " + idInsumo + ") * " +
            diferencia + ")" +
            " WHERE idProyecto = " + idProyecto + "";
            SqlCommand comd = new SqlCommand(consulta, conexion);
            comd.ExecuteNonQuery();
            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show("La informacion se esta utilizando en la tabla
            Empleado Proyecto");
        }
    }
```

Clase Orden:

```
    //Función obtenerProveedor, donde hacemos una consulta SQL a la tabla proveedor
donde obtenemos los datos de un proveedor específico. Y dichos datos se guardan en
una variable para tener mayor control y accesibilidad a ellos.
    private void obtenerProveedores()
    {
        try
        {
            conexion.Open();
            string query = "SELECT idProveedor, nombre FROM
            Empresa.Proveedor";
            SqlCommand comando = new SqlCommand(query, conexion);
            SqlDataAdapter adaptador = new SqlDataAdapter(comando);
            DataTable dt = new DataTable();
            adaptador.Fill(dt);
            comboProveedores.DataSource = dt;

            comboProveedores.DisplayMember= "nombre";
            comboProveedores.ValueMember= "idProveedor";

            conexion.Close();
        }
        catch (Exception ex) {
            conexion.Close();
            throw new Exception(ex.Message);
        }
    }
```

```
//Función del botón Nueva orden, en donde insertamos una nueva orden a un proyecto específico.
private void btnAgregar_Click(object sender, EventArgs e)
{
    try
    {
        DataRowView proveedorSel = (DataRowView) comboProveedores.SelectedItem;
        int idProveedor = (int)proveedorSel["idProveedor"];

        conexion.Open();
        string query = "INSERT INTO Orden.Orden(idProveedor,fechaOrden,total) VALUES (@idProveedor,GETDATE(),0)";

        SqlCommand comando = new SqlCommand(query, conexion);
        comando.Parameters.Add(new SqlParameter("idProveedor", idProveedor));

        comando.ExecuteNonQuery();
        conexion.Close();
        actualizaTablas();
        desactivarBotones();

    }
    catch (Exception ex)
    {
        conexion.Close();
        throw new Exception(ex.Message);
    }
 }

//Función del botón Ordenar Herramientas.
private void button1_Click(object sender, EventArgs e)
{
    DataGridViewRow ordenSeleccionada = tablaOrden.SelectedRows[0];
    long idOrdenSeleccionada = (long)ordenSeleccionada.Cells[0].Value;

    delegadoActualizaDatosTabla delegado = new delegadoActualizaDatosTabla(muestraDatosTabla);

    NuevaOrden ordenHerramienta = new NuevaOrden("Herramienta",idOrdenSeleccionada,delegado);
    ordenHerramienta.ShowDialog();
}
```

```
//Función del botón Ordenar Insumo.
private void button2_Click(object sender, EventArgs e)
{
    DataGridViewRow ordenSeleccionada = tablaOrden.SelectedRows[0];
    long idOrdenSeleccionada = (long)ordenSeleccionada.Cells[0].Value;

    delegadoActualizaDatosTabla delegado = new
    delegadoActualizaDatosTabla(muestraDatosTabla);

    NuevaOrden ordenInsumo = new
    NuevaOrden("insumo",idOrdenSeleccionada, delegado);

    ordenInsumo.ShowDialog();
}
```

Clase NuevaOrden:

```
//Consulta SQL para mostrar información de la tabla herramienta de la base de
datos.
private void muestraDatosTablaHerramienta()
{
    try
    {
        conexion.Open();
        string query = @"SELECT do.idOrden,do.idHerramienta, h.nombre
        AS herramienta,do.cantidad, do.subtotal
                        FROM Orden.DetalleOrden do
                        INNER JOIN Proyecto.Herramienta h
                        ON do.idHerramienta = h.idHerramienta
                        WHERE do.idOrden = @idOrden";
        SqlCommand comando = new SqlCommand(query, conexion);
        comando.Parameters.Add(new SqlParameter("idOrden", idOrden));
        SqlDataAdapter adaptador = new SqlDataAdapter(comando);
        DataTable dt = new DataTable();
        adaptador.Fill(dt);
        dataGridView1.DataSource = dt;
        dataGridView1.AutoSize = true;
        conexion.Close();
    }
    catch (Exception ex)
    {
        conexion.Close();
        throw new Exception(ex.Message);
    }
}
```

```
//Consulta SQL para mostrar información de la tabla insumo de la base de datos.
private void muestraDatosTablaInsumos()
{
    try
    {
        conexion.Open();
        string query = @"SELECT do.idOrden,do.idInsumo, i.nombre AS
        insumo, do.cantidad, do.subtotal
                                FROM Orden.DetalleOrden do
                                INNER JOIN Proyecto.Insumo i
                                ON do.idInsumo = i.idInsumo
                                WHERE do.idOrden = @idOrden";

        SqlCommand comando = new SqlCommand(query, conexion);
        comando.Parameters.Add(new SqlParameter("idOrden", idOrden));

        SqlDataAdapter adaptador = new SqlDataAdapter(comando);
        DataTable dt = new DataTable();
        adaptador.Fill(dt);
        dataGridView1.DataSource = dt;
        dataGridView1.AutoSize = true;
        conexion.Close();
    }
    catch (Exception ex)
    {
        conexion.Close();
        throw new Exception(ex.Message);
    }
}

//Función de inicialización, ya sea de insumo o herramienta.
private void inicializaComponentes()
    {
        if (this.producto.Equals("insumo", StringComparison.OrdinalIgnoreCase)){
            lblProducto.Text = "Insumo";
            obtenerComboInsumos();
            btnAgregar.Click += agregarInsumo;
            btnModificar.Click += modificarInsumo;
            btnEliminar.Click += eliminaDetalleInsumo;
            muestraDatosTablaInsumos();
        }
        else
        {
            lblProducto.Text = "Herramienta";
            obtenerComboHerramientas();
```

```
            btnAgregar.Click += agregarHerramienta;
            btnModificar.Click += modificarHerramienta;
            btnEliminar.Click += eliminaDetalleHerramienta;
            muestraDatosTablaHerramienta();
        }
    }

    //La siguiente función nos ayuda a identificar qué herramienta se está seleccionando en el comboBox y se guarda esa información.
    private void obtenerComboHerramientas()
    {
        try
        {
            conexion.Open();
            string query = "SELECT idHerramienta, nombre, cantidad_disponible, precio FROM Proyecto.Herramienta";
            SqlCommand comando = new SqlCommand(query, conexion);
            SqlDataAdapter adaptador = new SqlDataAdapter(comando);
            DataTable dt = new DataTable();
            adaptador.Fill(dt);
            comboProducto.DataSource = dt;

            comboProducto.DisplayMember = "nombre";
            comboProducto.ValueMember = "idHerramienta";

            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            throw new Exception(ex.Message);
        }
    }


    //La siguiente función nos ayuda a identificar qué insumo se está seleccionando en el comboBox y se guarda esa información.
    private void obtenerComboInsumos()
    {
        try
        {
            conexion.Open();
            string query = "SELECT * FROM Proyecto.Insumo";
            SqlCommand comando = new SqlCommand(query, conexion);
            SqlDataAdapter adaptador = new SqlDataAdapter(comando);
            DataTable dt = new DataTable();
```

```
            adaptador.Fill(dt);
            comboProducto.DataSource = dt;

            comboProducto.DisplayMember = "nombre";
            comboProducto.ValueMember = "idInsumo";

            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            throw new Exception(ex.Message);
        }
    }

    // Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla orden de la base de datos, en donde se resta el subtotal.
    private void restaSubtotalActual()
    {
        conexion.Open();
        string query = @"UPDATE Orden.Orden
                         SET total = total - @subtotal
                         WHERE idOrden = @idOrden";

        SqlCommand comando = new SqlCommand(query, conexion);
        comando.Parameters.Add(new SqlParameter("idOrden", idOrden));
        comando.Parameters.Add(new SqlParameter("subtotal", subtotalActual));

        comando.ExecuteNonQuery();
        conexion.Close();
    }


    // Función, en donde se realiza una consulta SQL para actualizar información sobre la tabla orden de la base de datos, en donde se suma el subtotal.
    private void sumaSubtotalActual(decimal nuevoSubtotal)
    {
        conexion.Open();
        string query = @"UPDATE Orden.Orden
                         SET total = total + @subtotal
                         WHERE idOrden = @idOrden";

        SqlCommand comando = new SqlCommand(query, conexion);
        comando.Parameters.Add(new SqlParameter("idOrden", idOrden));
        comando.Parameters.Add(new SqlParameter("subtotal", nuevoSubtotal));
```

```
            comando.ExecuteNonQuery();
            conexion.Close();
        }

    //Función delegado, hace referencia a un tipo de valor devuelto y accedemos a la clase Orden.
    public void actualizarTablaOrden()
        {
            delegado();
        }

    //Función para restablecer controles.
    private void reiniciarControles()
        {
            textBoxCantidad.Clear();
            btnModificar.Enabled = false;
            btnEliminar.Enabled = false;
        }
```

References

Microsoft. (2023, 05 23). *Delegates (C# Programming Guide)*. Retrieved from https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/

Microsoft. (2023, 05 23). *Documentación de C#*. Retrieved from https://learn.microsoft.com/es-es/dotnet/csharp/

Microsoft. (2023, 05 23). *SQL Server technical documentation*. Retrieved from https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16