MANUAL DE PROGRAMADOR

Detallar el código de manera compresible para tener la idea de la funcionalidad que ejerce la aplicación de escritorio.

Código Generado por C#

Universidad Don Bosco

FACULTDAD DE ESTUDIO TECNOLÓGICOS

ESCUELA DE COMPUTACIÓN

Desarrollo De Aplicaciones Con Software Propietario

Docente: Rene Tejada

Tema: Proyecto De Cátedra

ESTUDIANTES:

Hernández Hernández	**Marco Antonio**	**HH182006**
Javier Ernesto	**Pérez Pablo**	**PP180605**
Márquez Saravia	**Daniel Orlando**	**MS180652**
Molina Ruano	**Alexis Rodrigo**	**MR180477**
Vega Velasco	**Laura Nohemy**	**VV181025**

Contenido

Manual del Programador

Descripción:

Nuestro sistema está orientado hacia el área de administración de bodega principalmente, sin embargo, el sistema nos da la disponibilidad de agregar más empleados a la base de datos, la cual lleva un control tanto de repuestos de automóviles y/o motos, en inventario, así como una lista de sucursales y usuarios que pueden irse agregando a medida que el negocio se expanda o se agreguen mas trabajadores. Los cargos que un empleado puede tener son: Administrador General, Gerente de Bodega Y Empleado.

Cada uno tiene un poder específico; El administrador general puede crear usuarios y sucursales, este tiene el poder absoluto en la empresa.

El gerente de bodega puede llevar el control de cada artículo y editar su información, así como también añadir o eliminar a los mismos.

Finalmente el empleado común puede ver los elementos en inventario, mas no editar, añadir o eliminar nada en el sistema.

MATERIALES Y EQUIPO

Para la realización de la guía de programador se requerirá lo siguiente:

N°	Requerimiento	Cantidad
1	PC con Microsfot Visual Studio. NET instalado	1
2	SQL Server Management Studio	1
3	El Programa de escritorio y web	1

PROCEDIMIENTO

Parte I: Análisis del sistema ante de empezar con el código debemos tener en cuenta la conexión de la base de datos la cuales estar declarado en la siguientes clases

- ✓ Clases Conexión de administrador Principal
- ✓ Clase Conexión de administrador Bodega
- ✓ Clase Conexión de la sucursales

Luego debemos hacer la conexión con la ruta especificada que se muestra abajo. Qué se encuentra en cada una de las clases de conexión menciona anterior.

```
// Cadena de conexión
private SqlConnection conexion = new SqlConnection("DataSource=(local);Initial
Catalog=BDD;Integrated Security=True"); // Establece la conexión de la base de
datos.
private DataSet DS; // Uso Guardad varias tablas llamada datatable. Y mostrar los
datos.
```

Conexión de la clase de administrador Principal

Parte 2: En las clases conexión de administrador Principal tendremos 5 métodos la cuales son Mostrar Datos, Buscar, Insertar, Modificar y eliminar. La cual se detallar su uso.

El método de mostrar Datos: Este método habré la conexión de la base de datos, la cual busca la tabla de los usuarios y toma la información que será mostrada en la datagridview.

```
public DataTable MostrarDatos()
        {
            conexion.Open(); // Abrimos la conexión
            SqlCommand cmd = new SqlCommand("select * from Usuarios", conexion);//
Abre la conexión de la tabla de los Inventario
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirve si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las tablas

            ad.Fill(DS, "tabla"); //Llenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

El método de buscar: Este método sirve para buscar a la persona, la cual le pásame el parámetro el string nombre. La cual servirá para la consulta a la base de datos de la tabla de los Usuarios. Entonces por medio comando select solo buscamos a algo específico.

```
// Método para buscar el nombre de la persona
        public DataTable Buscar(string nombre)
        {
            conexion.Open(); // Abrimos la conexión         seleccionar toda las
filas de los usuarios cuando  Nombre  contenga el argumento
            SqlCommand cmd = new SqlCommand(string.Format("select * from Usuarios
where Nombre like  '%{0}%' ", nombre), conexion);// Abre la conexión de la tabla
de los usuarios
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirver si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las talbas

            ad.Fill(DS, "tabla"); // duda LLenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

Método para insertar: Este método esencial para ingreso de información por lo tanto se le pasan parámetros con su respectiva información ingresada la cual serán de tipo de string. Luego en los argumentos será de tipos char para no tener problemas con numéricos es por eso que la validaciones se encargar evaluar que se detallaran más adelante.

```
//Método para insertar los datos del producto
        public bool insertar( string Nombre, string Apellidos,  string direccion,
string contacto, string fechaNacimiento, string usuario, string contrasena ,string
Idcargo)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                                          // inserta la
información de los usuarios por medio de los  argumentos se ira ingresado dicha
información a los parámetros
            SqlCommand cmd = new SqlCommand(string.Format("insert into Usuarios
values ( '{0}' , '{1}' , '{2}', '{3}' ,'{4}', '{5}' , '{6}' , {7}  )", new
string[] { Nombre, Apellidos,  direccion, contacto, fechaNacimiento, usuario,
contrasena, Idcargo }), conexion);
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerramos la conexión
            if (FilasAfectadas > 0) return true; // inserta datos
            else return false;
        }
```

Método para Llenar Cargo: Este método se encargar de traer dicha información de la base de datos. La cual especificamos cual ira atraer en este caso es el cargo la cual en la base de datos están registrado Empleado, Administrador de Bodega y El administrador Principal. La cual se ira llenar al ComboBox que se encuentra en el interfaz.

```
// Llenamos combobox  que traemos dese la base de datos
        public void llenearCargo(ComboBox Cargo)
        {
            conexion.Open();//abrimos la conexión
            DataTable Lista = new DataTable();// Búsqueda  para información de la
tabla
            SqlDataAdapter ada;                // por medio select selecciona Id y
el nombre para traer la dicha información
            SqlCommand cmd = new SqlCommand(string.Format("select ID_De_Cargo
,Nombre from Cargos "), conexion);// Abre la conexión de la tabla de cargo para
traer dicha información ingresada base de datos
            ada = new SqlDataAdapter(cmd);
            ada.Fill(Lista);// LLenamos la información
            Cargo.DataSource = Lista;
            Cargo.DisplayMember = "Nombre"; // Traemos el texto
            Cargo.ValueMember = "ID_De_Cargo"; // Traemos el valor


            conexion.Close();// Cerramos la conexión


        }
```

Método para eliminar: El método de eliminar se encargara de eliminar la fila completa desde la base de datos cual por medio del parámetro ID se identifica la fila actual.

```
public bool Eliminar(string id)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                                          eliminar las filas de
la pieza la cuales  este en el id
            SqlCommand cmd = new SqlCommand(string.Format("delete from Usuarios
where ID_de_Usuario = {0} ", id), conexion);// Formato para eliminar  a la persona
por medio del IDUsuario
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // isnetar datos
            else return false;
        }
```

El método para actualizar los datos: En este método evaluamos cada campo en este caso por medio de los parámetros actualizamos y por los argumentos llenamos el orden ingresado.

```
// Método para actualizar los datos
        public bool Actualizar(string id, string Nombre, string Apellidos , string
direccion, string contacto, string fechaNacimiento, string usuario, string
contrasena, string Idcargo)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                    actualizar las pieza las cuales evaluamos
por cada campo para que sean actualizado
String dato = string.Format("update Usuarios set Nombre = '{0}', Apellidos =
'{1}', Direccion  = '{2}' , Contacto = '{3}' , Fecha_de_nacimiento = '{4}' ,
Usuario = '{5}', Contraseña = '{6}'  , ID_De_Cargo= {7}  where ID_de_Usuario = {8}
", Nombre, Apellidos, direccion, contacto, fechaNacimiento, usuario, contrasena,
Idcargo,  id);//

            SqlCommand cmd = new SqlCommand(dato, conexion);// Formato para
actualizar los datos
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // inserter datos
            else return false;

        }
```

Conexión de la clase de Sucursal

Parte 3: Los métodos son parecidos al anterior así que se detallar tan específico.

En el método de Mostrar Datos Sucursales: Este método habré la conexión de la base de datos la cual busca la tabla de los sucursales cual tomara todo la información que se mostrada en datagridview..

```
// Metodo para mostrar los datos
        public DataTable MostrarDatosSucursal()
        {
            conexion.Open(); // Abrimos la conexión
            SqlCommand cmd = new SqlCommand("select * from sucursal", conexion);//
Abre la conexión de la tabla de la sucursales
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirver si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las talbas

            ad.Fill(DS, "tabla"); //LLenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

El método de buscar: Este método sirve para buscar a la sucursal la cual le pásame el parámetro el string de la sucursal a cual servirá para la consulta a la base de datos de la tabla de la sucursal. Entonces por medio comando select solo buscamos a algo específico.

```
// Método para buscar los sucursales
        public DataTable BuscarSucursal(string sucursal)
        {
            conexion.Open(); // Abrimos la conexión         seleccionar toda las
filas de los usuarios cuando  Nombre  contenga el argumento
            SqlCommand cmd = new SqlCommand(string.Format("select * from sucursal
where Nombre like  '%{0}%' ", sucursal), conexion);// Abre la conexión de la tabla
de los usuarios
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirver si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las talbas

            ad.Fill(DS, "tabla"); // duda LLenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

El método para insertar: Se encargar de insertar la información a la base de datos la cual por medio de los parámetros y traemos la información de los campos.

```
public bool insertar( string NombreSucursal, string Direccion)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                                        // inserta la
información de la sucursales por medeio de los  argumentos se ira ingresado dicha
información a los parametros
            SqlCommand cmd = new SqlCommand(string.Format("insert into sucursal
values ('{0}' , '{1}'  )", new string[] { NombreSucursal , Direccion }),
conexion);
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // isnetar datos
            else return false;
        }
```

El método eliminar la sucursal: Por medio de la id se encargara de eliminar la fila que ha seleccionado id la cual por medio de la base de datos se elimina.

```
public bool EliminarSucursal(string id)
        {
            conexion.Open();
            SqlCommand cmd = new SqlCommand(string.Format("delete from sucursal
where ID_Sucursal = {0} ", id), conexion);
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // isnetar datos
         }
```

El método Para Actualizar: En este caso debemos evaluar por cada campo entonces debemos pasar dichos parámetros para la actualización en la base de datos.

```
public bool ActualizarSucursal(string id, string NombreSucursal, string Direccion)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                   actualizar a los sucursales
            String dato = string.Format("update sucursal set Nombre = '{0}',
Direccion = '{1}'   where ID_Sucursal = '{2}' ", NombreSucursal,Direccion, id);
SqlCommand cmd = new SqlCommand(dato, conexion);// Formato para actualizar los
datos
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerramos la conexión
            if (FilasAfectadas > 0) return true; // insertar datos
            else return false;

        }
```

Conexión de la clases de administrador Bodega

Parte 4: Los métodos son parecidos al anterior así que se detallar tan específico.

El método de Mostrar datos: se encargar de enviar la información de la datagridview la cual extraído de la base de datos.

```
// Método para mostrar los datos
        public DataTable MostrarDatos()
        {
            conexion.Open(); // Abrimos la conexión
            SqlCommand cmd = new SqlCommand("select * from Inventario",
conexion);// Abre la conexión de la tabla del inventario
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirve si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las talbas

            ad.Fill(DS, "tabla"); //  Llenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

Método para buscar: le pasamos el parámetro string de pieza la cual le indicamos que busque algo en específico en la base de datos.

```
// Método para buscar la pieza
        public DataTable Buscar(string pieza)
        {
            conexion.Open(); // Abrimos la conexión         seleccionar toda las
filas de los usuarios cuando  pieza  contenga el argumento
            SqlCommand cmd = new SqlCommand(string.Format("select * from
Inventario where Pieza like  '%{0}%' ", pieza), conexion);// Abre la conexión de
la tabla del inventarios
            SqlDataAdapter ad = new SqlDataAdapter(cmd);// sirve si el comando es
de tipo select
            DS = new DataSet(); //Borrar todas las talbas

            ad.Fill(DS, "tabla"); // Llenemos a la tabla
            conexion.Close();// cerramos la conexión para ocupar en otro lugar
            return DS.Tables["tabla"]; // retomar el valor
        }
```

En el método para insertar: es la parte esencial ya que debemos tener en cuentas cada parámetro que será enviados hacia a la base de datos, En este caso todo son de tipos de varchar pero teniendo las validaciones para dicho campos

```
// Método para insertar
        public bool insertar(string pieza , string codigo , string modelo ,string
marca , string precio , string cantidad, string NumeroEstante,string Fabricante,
string descripcion,string FechaRegistro )
        {
            conexion.Open(); // Abrimos la conexión          // insertamos por
medio de los argumentos los datos en el orden
llenamos los datos por medio de varchar
            SqlCommand cmd = new SqlCommand(string.Format("insert into Inventario
values ('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}')", new
string[]
{pieza,codigo,modelo,marca,precio,cantidad,NumeroEstante,Fabricante,descripcion,Fe
chaRegistro }),conexion);
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // isnetar datos
            else return false;
```

El método para eliminar: se identificar por el id la cual seleccionara todo la fila para eliminar la dicha información de la base de datos.

```
public bool Eliminar(string id)
        {
            conexion.Open(); // Abrimos la conexión
            //comando                                        eliminar las filas de
los inventarios la cuales  este en el id
            SqlCommand cmd = new SqlCommand(string.Format("delete from Inventario
where ID_de_inventario = {0} ", id), conexion);// Formato para eliminar  a la
persono por medio del IDventario
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerramos la conexión
            if (FilasAfectadas > 0) return true; // insertar datos
            else return false;
        }
```

El método para actualizar los datos evaluamos cada campo para hacer los cambios es necesarios pasar todos los parámetros para cambios para que reenvié hacia la base de datos.

```
public bool Actualizar(string id, string pieza, string codigo, string modelo,
string marca, string precio, string cantidad, string NumeroEstante, string
Fabricante, string descripcion, string FechaRegistro)
        {
            conexion.Open(); // Abrimos la conexión

            //comando                   actualizar a los usuarios para eso
evaluamos por cada cada argumento y por medio de parámetros
            String dato = string.Format("update Inventarios set Pieza='{0}' ,
Codigo = '{1}', Modelo = '{2}', Marca = '{3}' , Precio = '{4}' , Cantidad = '{5}'
, Numero_de_estante = '{6}', Fabricante='{7}', Descripcion ='{8}',
Fecha_de_registro = '{9}' , where ID_de_inventario = {10} ",
pieza,codigo,modelo,marca,precio,cantidad,NumeroEstante,Fabricante,descripcion,Fec
haRegistro,id);

            SqlCommand cmd = new SqlCommand(dato, conexion);// Formato para
actualizar los datos
            int FilasAfectadas = cmd.ExecuteNonQuery();// uso para ver las filas
ha sido afectadas
            conexion.Close(); // cerrramos la conexión
            if (FilasAfectadas > 0) return true; // isnetar datos
            else return false;
        }
```

Clases para validar Campos

En esta clase validamos los campos que vamos a utilizar en nuestro formulario. Este consta de Tres métodos manejados por evento

ValidarCampoLetra(): En este método se verifica mediante un método bool, el cual está compuesto de varios If, los cuales verifican si el dato ingresado es un carácter y/o, si no lo es, este método mediante el else de un if, retorna false, verificando que es un número el que se ha ingresado y no permitirá que se digite.

```
//Validar si son letras  por medio de los eventos
        public void ValidarCampoLetras(KeyPressEventArgs e)
        {
            try
            {
                if (char.IsLetter(e.KeyChar)) // verifica si es una letra que la
escriba
                {
                    e.Handled = false;  // si no es un numero que escriba la letra
                }
                else if (char.IsControl(e.KeyChar))// tecla de control que permita
borrar
                {
                    e.Handled = false; // permite borrar
                }
                else if (char.IsSeparator(e.KeyChar)) // permiter el uso de
espaciador
                {
                    e.Handled = false; //si presiona el espaciador dejara un
espacio
                }

                else
                {
                    e.Handled = true; // Si no es letra, ni borrar ni espacio
entonce que no lo escriba.
                }
            }
            catch { }
        }// Fin de validar campos de letras
```

ValidarCampoNumero(): En este método se sigue el mismo proceso anterior, se valida mediante el método IF si cumple las características de un número, si cumple alguna condición, retorna True, pero si este no es el caso, retorna False.

```
// Validar un número por medio de los eventos KeyPress
        public void ValidarCampoNumeros(KeyPressEventArgs e)
        {
            try
            {
                if (char.IsNumber(e.KeyChar)) // verifica si es un numero
                {
                    e.Handled = false; // Si no es una letra que escriba el número
                }
                else if (char.IsControl(e.KeyChar))// tecla de control que permita
borrar
                {
                    e.Handled = false;
                }
                else if (char.IsSeparator(e.KeyChar)) // permiter los  espacio
                {
                    e.Handled = true;
                }
                else
                {
                    e.Handled = true; // Si no es un número que no escriba
                }
            }
            catch { } // Fin de validar campos de letras
        }// Fin del Metodo validar Número
```

ValidarCampoVacio():En este caso es similar, solo que para que este método se ejecute, no debe ser ni carácter ni número, debe estar vacío el campo, si lo está; se ejecuta este método.

```
public bool CamposVacio(string campo) // en el parametro envia la información la
cual es verificada en el form
        {
            try
            {
                // Codigo comprovación de textbox
                if (campo == "")
                {
                    MessageBox.Show("Campo vacio");

                    return false; // si el campo esta vacio no permitara guardar
el dato

                }

            }// Fin del try

            catch { }
            return true;
        }// fin del meto campos vacio
```

Instancias declaradas por cada interfaz

Parte 5: Declaramos las instancia por cada interfaz para hacer uso de los métodos que fueron asignado en la clases la cuales fueron `insertar, modificar, Eliminar , buscar O llenar Combobox`.

Interfaz del Administrador Principal

```
// Instancia
conexionAdministradorPrincipal sqlConexion = new conexionAdministradorPrincipal();
// instancia de la clase de conexión Administrador Principal
```

`Instancia conexión administrador Sucursal traemos lo mismo insertar, modificar, eliminar o buscar pero para la sucursal`

```
Sucursal sqlConexionSucusal = new Sucursal(); // Instancia de la clase derivada de
la conexión de administrador Principal
```

`Instancia para validar los campos lo cuales traemos diferentes métodos de validar Número, Letras o los campos Vacío.`

```
ValidacionesDeCampos Validar = new ValidacionesDeCampos(); // Instancia de la
clase de validar los campos
```

Interfaz del Administrador de Bodega

Instancia conexión de administrador de bodega traemos lo mismo insertar, modificar, eliminar o buscar pero para la sucursal.

```
// Instancia de la clase de conexión Administrador Bodega
 ConexionAdministradorBodega sqlConexion = new ConexionAdministradorBodega();
```

`Instancia para validar los campos lo cuales traemos diferentes métodos de validar Número, Letras o los campos Vacío.`

```
ValidacionesDeCampos Validar = new ValidacionesDeCampos(); // Instancia de la
clase de validar los campos
```

Métodos Públicos

Parte 6

Interfaz del Administrador Principal

El método para Fecha: se especializa de evaluar el día, mes y el año por datetimepicker la cual sirve para sacar la fecha de nacimiento de la persona.

```
//Métodos Públicos
        public void Fecha() // Metodo para sacar la fecha de nacimiento del
empleado
        {
            dia = dtpFechaNac.Value.Day; // Evalua el dia
            mes = dtpFechaNac.Value.Month; // Evalua mes
            año = dtpFechaNac.Value.Year; // Evalua Año
            fechaNacimiento = dia + "/" + mes + "/" + año; // Une la fachas
        }
```

El **método para Borrar Campos user**: solo se encargar de limpiar las cajas de textos de crear cuenta de usuario.

```
//Método para borrar los campos del usuario
        public void BorrarCamposUser()
        {  // Limipa las cajas
            txtNombre.Clear();
            txtApellido.Clear();
            txtDireccion.Clear();
            txtContacto.Clear();
            txtUser.Clear();
            txtPasword.Clear();
            txtCargo.Text = "";


        }
```

EL método para borrar Campo Sucursal: solo se encarga de limpiar las cajas de textos de crear los sucursales

```
// Método para borrar los campos de la Sucursal
        public void BorrarCamposSucursal()
        { // limpiar la caja de textos
            txtNombreSucur.Clear();
            txtDireccionSucursal.Clear();
        }
```

El método de cerrar sesión: se encargar de cerrar la aplicación y volver al formulario de login y de igual forma para todos.

```
//Método para Regresar Inicar Sesión
        public void CerrarSesion()
        {
Login RegrearLogin = new Login(); // Instancia del formulario del login
RegrearLogin.Show(); //Dirigí al formulario del login
            this.Hide(); // Cerramos el formulario Actual
        }
```

Interfaz del Administrador de Bodega:

El método para Fecha se especializa de evaluar el día, mes y el año por datetimepicker la cual sirve para sacar la fecha de registro del producto.

```
public void Fechas() // Metodo para sacar la fehca de los registros
        {
            dia = dtpFechaRigistro.Value.Day; // Evalua el dia
            mes = dtpFechaRigistro.Value.Month; // Evalua mes
            año = dtpFechaRigistro.Value.Year; // Evalua Año
            fechaRegistro = dia + "/" + mes + "/" + año; // Une la fachas
        }
```

El método para borrar campos solo se encargar de limpiar las cajas de textos del inventario.

```
// Método para borrar los campos de los textos
        public void BorrarCampos()
        {
            txtPieza.Clear();
            txtFabricante.Clear();
            txtCodigo.Clear();
            txtCantidad.Clear();
            txtModelo.Clear();
            txtMarca.Clear();
            txtPrecio.Clear();
            txtNumeroEstante.Clear();
            txtDescipcion.Clear();
        }
```

ADMINISTRADOR GENERAL

```
// Metodo para bloquear las pestañas
public void BloqueoDePestañas()
{
((Control)this.tabPage2).Enabled = false;
((Control)this.tabPage1).Enabled = true;
((Control)this.txtIDInventario).Enabled = false;
}
```

En esta función, bloqueamos la tabPage2 donde se puede ver la bodega, habilitamos la pestañas principal y bloqueamos la caja de texto del ID

```
// Metodo para Dirigir a la pestañas de Ver los productos
public void DirigirPestaña()
{
tbcFormularioAdm.SelectedIndex = 1;
((Control)this.tabPage1).Enabled = false;
((Control)this.tabPage2).Enabled = true;
}
```

En este método redirigimos hacia la pestaña inventario, habilitamos la pestaña del administrador y deshabilita la pestaña del ver los registro del inventario.

```
//Metodo para regresar a la pestañas de inico
public void RegresarInicio()
{
tbcFormularioAdm.SelectedIndex = 0;
((Control)this.tabPage1).Enabled = true;
((Control)this.tabPage2).Enabled = false;
}
```

En este Método regresamos a la pestaña principal de administrador, Habilita la pestaña actual y deshabilita la pestaña de los registro de productos

GERENTE DE BODEGA

```
//Metodo para bloquear las pestañas
public void BloqueoDePestaña()
{
((Control)this.tabPage1).Enabled = true;
((Control)this.tabPage2).Enabled = false;
((Control)this.tabPage3).Enabled = false;
((Control)this.txtID).Enabled = false;
((Control)this.txtIDSucursal).Enabled = false;
}
```

En este método podemos habilitamos la pestaña principal, bloqueamos la pestaña de ver los usuario, bloqueamos la pestaña de las sucursales, bloqueamos la caja de texto del id de los usuario y bloqueamos la caja de texto id sucursales

```
// Metodo para Dirigir a la pestaña ver usuario
public void DirigirPestañasUser()
{
tbcFormularioAdm.SelectedIndex = 1; //
((Control)this.tabPage1).Enabled = false;//
((Control)this.tabPage2).Enabled = true; //
}
```

En este método dirige a la pestaña para ver a los usuario, Bloquea la pestaña principal y habilita la pestaña de los usuario

```
//Metodo para Dirigir a la pestañas de ver Sucursales
public void DirigirPestañaSucursales()
{
tbcFormularioAdm.SelectedIndex = 2;
((Control)this.tabPage1).Enabled = false;
((Control)this.tabPage3).Enabled = true;
}
```

En este método podemos Dirigirnos a la pestaña de las sucursales, bloquear la pestaña principal y habilitar la pestaña de las sucursales.

```
// Metodo para Regresar al Inicio
public void DirigirPestañaRegreso()
{
tbcFormularioAdm.SelectedIndex = 0;
((Control)this.tabPage1).Enabled = true;
((Control)this.tabPage2).Enabled = false;
((Control)this.tabPage3).Enabled = false;
}
```

En este método podemos dirigirnos a la pestaña principal del administrador, habilitar la pestaña principal, bloquear las pestañas de los usuarios, bloquear las pestañas de las sucursales.

EMPLEADO

```
public void BloqueoPestaña()
{
((Control)this.tabPage1).Enabled = true;//
((Control)this.tabPage2).Enabled = false; //
((Control)this.dgvProductos).Enabled = false; //
}
```

En este método podemos Bloquear pestaña 1, habilitar la pestaña del empleado, bloque la pestaña 2 y bloquear el dgv.

```
public void DirigirPestaña()
{
tbcEmpleado.SelectedIndex = 1;
((Control)this.tabPage1).Enabled = false;
((Control)this.tabPage2).Enabled = true;
}
```

En este método podemos dirigirnos a pestaña de inventario, bloquear la pestaña 1 y habilitar la pestaña 2

```
public void Regresar()
{
tbcEmpleado.SelectedIndex = 0;//
((Control)this.tabPage1).Enabled = true; //
((Control)this.tabPage2).Enabled = false; // }
```

En este método podemos Regresar a la pestaña actual, habilitar la pestaña 1, bloquear la pestaña 2

Método para Insertar, Eliminar, Actualizar, buscar

El método de validar campos e insertar usuario se encargar primero de verificar los campos vacío luego insertar la información la cual que debemos pasar medio del método que fue creado en la clase de conexión de administrador principal nos traemos la instancia sqlConexión.Insertar y le pasamos la información por medio de parámetros. Luego tenemos la condición si se ha insertado los datos le dirá Datos Insertados con éxito y le mostrar la información o no se ha insertado la información.

Insertar Usuarios

```
// Insertamos los datos hacia la base de datos
  txtID.Text = dgv_tablauser.Rows.Count.ToString(); // Mostrar el indice del
Usuario es decir el ID del Usuario  // Insertamos la dicha información que sera
dirigido a la clase por medio de parametros
 if (sqlConexion.insertar(txtNombre.Text, txtApellido.Text, txtDireccion.Text,
txtContacto.Text, fechaNacimiento, txtUser.Text, txtPasword.Text,
txtCargo.SelectedValue.ToString()))// Insertamos los datos hacia la base de datos
  {
  MessageBox.Show("Datos Insertados con exito");// Mensaje que se insertó los
datos
 dgv_tablauser.DataSource = sqlConexion.MostrarDatos(); // El resultado se
mostrara datagridview
}
else
{
MessageBox.Show("No se han podido Insertar los datos"); // No se ha podido
insertar los datos
}
```

Insertar Sucursales.

El método de validar campo e insertar Sucursales es el mismo proceso que se explicó anteriormente.

```
// Insertamos los datos hacia la base de datos de la sucursal
 txtIDSucursal.Text = dgvSucursales.Rows.Count.ToString();// Mostrar el indice de
la sucurales es decir el ID de la sucursales
   if (sqlConexionSucusal.insertar(txtNombreSucur.Text,
txtDireccionSucursal.Text))
         {
  MessageBox.Show("Datos Insertados con exito");// Mensaje que se inserto los
datos
   dgvSucursales.DataSource = sqlConexionSucusal.MostrarDatosSucursal();
   }
      else
{
 MessageBox.Show("No se han podido Insetar los datos"); // No se ha podido insetar
los datos
  }
```

Insertar inventario

De igual de forma para insertar el inventario

```
txtIDInventario.Text = dgvProductos.Rows.Count.ToString(); // Mostrar número del
producto es decir el ID Producto

// Por medio de la condición insertamos la información que le  pasmos por
parametros  hacia a la clase  sqlConexión hacia al metodo de insertar
if (sqlConexion.insertar( txtPieza.Text, txtCodigo.Text, txtModelo.Text,
txtMarca.Text, txtPrecio.Text, txtCantidad.Text, txtNumeroEstante.Text,
txtFabricante.Text, txtDescipcion.Text, fechaRegistro))
  {
MessageBox.Show("Inserto la información con exito");
dgvProductos.DataSource = sqlConexion.MostrarDatos(); // Una vez ingresa los datos
se recargar mostrarDatos
   }
else
{
MessageBox.Show("No se han podido insertar los datos");// Mensaje de Error al
ingresar los datos
}
```

Luego para eliminar debemos llamar a la instancia sqlConexión.Eliminar y pasar el parámetro que fue declaro en la clase y método se encargar de ser su proceso que se detalló ante. De igual forma es para eliminar una sucursal solo que será sqlConexiónSucursal.eliminar.

Eliminar Usuario

```
if (sqlConexion.Eliminar(txtID.Text))// Eliminamos los datos hacia la base de
datos
{
MessageBox.Show("Datos Eliminados");// Mensaje que se inserto los datos
                dgv_tablauser.DataSource = sqlConexion.MostrarDatos(); // El
resultado mostrara datagriview
}
else
 {  MessageBox.Show("No se han podido eliminar los datos"); // No se ha podido
eliminar los datos
            }
```

Eliminar Sucursal

```
if (sqlConexionSucusal.EliminarSucursal(txtIDSucursal.Text))// Por medio de la id
va seleccionar todo la fila para eliminar los datos
            {
// donde se realizara por medio del metodo traido por clase
                MessageBox.Show("Datos Eliminados");
```

```
dgvSucursales.DataSource = sqlConexionSucusal.MostrarDatosSucursal();
            }
            else
            {
MessageBox.Show("No se han podido actualizar los datos"); // No se ha podido
insertar los datos
            }
```

Eliminar Inventario

```
if(sqlConexion.Eliminar(txtIDInventario.Text)) // Eliminamos los datos hacia la
base de datos
            {
                MessageBox.Show("Datos Eliminados");
                dgvProductos.DataSource = sqlConexion.MostrarDatos(); // Se
recargar los datos y vuelve mostrar los datos que estan en la base de datos
            }
            else
            {
                MessageBox.Show("No se han podido eliminar los datos"); // No se
ha podido elimar los datos
            }
```

Luego tenemos para **actualizar** por medio del método que es llamado por la clase le pasamos los parámetros la cual el método se hará el cargo de hacer los cambios.

Actualizar Usuarios

```
if (sqlConexion.Actualizar(txtID.Text, txtNombre.Text, txtApellido.Text,
txtDireccion.Text, txtContacto.Text, fechaNacimiento, txtUser.Text,
txtPasword.Text, txtCargo.SelectedValue.ToString()))// Insertamos los datos hacia
la base de datos
            {
MessageBox.Show("Datos actualizados");// Mensaje que se insertó los datos
                dgv_tablauser.DataSource = sqlConexion.MostrarDatos(); // El
resultado mostrara datagriview
            }
            else
            {
MessageBox.Show("No se han podido actualizar los datos"); // No se ha podido
inserta los datos
            }
```

Actualizar Sucursales

```
  // Vuelve a verificar por cada campo si hay un cambio  para poder insertar de
nuevo la información
if (sqlConexionSucusal.ActualizarSucursal(txtIDSucursal.Text, txtNombreSucur.Text,
txtDireccionSucursal.Text))
            {
MessageBox.Show("Datos actualizados");// Mensaje que se inserto los datos
```

```
                dgvSucursales.DataSource
=sqlConexionSucusal.MostrarDatosSucursal();
            }
            else
            {
 MessageBox.Show("No se han podido actualizar los datos"); // No se ha podido
insertar los datos
            }
```

Actualizar el inventario

```
if (sqlConexion.Actualizar(txtIDInventario.Text, txtPieza.Text, txtCodigo.Text,
txtModelo.Text, txtMarca.Text, txtPrecio.Text, txtCantidad.Text,
txtNumeroEstante.Text, txtFabricante.Text, txtDescipcion.Text, fechaRegistro))
            {
MessageBox.Show("Datos Actualizados"); // Mensaje que se inserto los datos
                dgvProductos.DataSource = sqlConexion.MostrarDatos(); // Recargar
la base de datos para mostrar los datos ingresado
            }
            else
            {
MessageBox.Show("No se han podido actualizar los datos");
            }
```

El buscador se encargar de hacer la consulta de la base de datos la cual por medio del método busca el nombre de la persona, pieza o nombre de la sucursal.

Buscador del Inventario

```
Buscador del Inventario
// Tiene que ser disto a vacío          Si ingresa la información  se mostrara la
dicha información que busca
if (txtBuscadorInventario.Text != "") dgvProductos.DataSource =
sqlConexion.Buscar(txtBuscadorInventario.Text);
            else dgvProductos.DataSource = sqlConexion.MostrarDatos(); // si no
existe que no muestre nada que se mantenga igual.
```

Buscador de la sucursal

```
// Tiene que ser disto a vacío          Si ingresa la información  se mostrara la
dicha información que busca
if (txtBuscadorSucursal.Text != "") dgvSucursales.DataSource =
sqlConexionSucusal.BuscarSucursal(txtBuscadorSucursal.Text);
else dgvSucursales.DataSource = sqlConexionSucusal.MostrarDatosSucursal(); // si
no existe que no muestre nada que se mantenga igual
```

Buscador de inventario

```
Buscador del Inventario
// Tiene que ser distinto a vacío    si ingresa la información se mostrar la dicha
información que busca
```

```
if (txtBuscadorUser.Text != "") dgv_tablauser.DataSource =
sqlConexion.Buscar(txtBuscadorUser.Text);
else dgv_tablauser.DataSource = sqlConexion.MostrarDatos(); // si no existe que no
muestra nada
```