Este diseño se corresponde con el video de apoyo número 6

El procedimiento de eliminar usuario solo debe ejecutarse si el usuario se ha creado por error. Si el usuario ya tiene registradas operaciones a su nombre (ej. Una compra) no debemos eliminarlo ya que se perdería la relación con otras tablas.

```
USE BDSISTEMAVENTAS;
GO
CREATE PROC SP_ELIMINARUSUARIO (
@Idusuario int,
@Documento varchar(50),
--Parámetros de salida resultado de operar con el registro usuario
@Respuesta bit output, --controla 1=se ha realizado operación 0=no se ha realizado la
operacion
@Mensaje varchar(500) output)
AS
BEGIN
	SET @Respuesta=0
	SET @Mensaje=''
	DECLARE @PasoRegla bit = 1

	if  exists(SELECT * FROM COMPRA INNER JOIN USUARIO ON
COMPRA.IdUsuario=USUARIO.IdUsuario
				WHERE USUARIO.IdUsuario= @Idusuario)
	begin
	    set @Respuesta=0
	    SET @PasoRegla=0;
	       set @Mensaje=@Mensaje+'No se puede eliminar porque este usuario tiene
operaciones de compra registradas.\n'
	end

	if  exists(SELECT * FROM VENTA INNER JOIN USUARIO ON
VENTA.IdUsuario=USUARIO.IdUsuario
				WHERE USUARIO.IdUsuario= @Idusuario)
	begin
	    SET @PasoRegla=0
	    set @Respuesta=0
	       set @Mensaje=@Mensaje+'No se puede eliminar porque este usuario tiene
operaciones de venta registradas.\n'
	end

	if (@PasoRegla = 1)
	begin
		DELETE USUARIO where IdUsuario=@IdUsuario
		SET @Respuesta=1
	end
END
```

Para llamar a los procedimientos almacenados que tenemos en nuestra base de datos modificamos nuestra clase CD_Usuario creando un nuevo método Registrar que va a devolver un entero, el idusuario. Recibirá un objeto usuario y tendrá un parámetro de salida de igual forma que funciona en el procedimiento SQL.

```
public int Registrar(Usuario obj ,out string Mensaje) {
    int idusuariogenerado = 0;
    Mensaje = string.Empty;


    try {

        using (SqlConnection oconexion = new SqlConnection(Conexion.cadena)) {

            SqlCommand cmd = new SqlCommand("SP_REGISTRARUSUARIO", oconexion);
            cmd.Parameters.AddWithValue("Documento",obj.Documento);
            cmd.Parameters.AddWithValue("NombreCompleto", obj.NombreCompleto);
            cmd.Parameters.AddWithValue("Correo", obj.Correo);
            cmd.Parameters.AddWithValue("Clave", obj.Clave);
            cmd.Parameters.AddWithValue("IdRol", obj.oRol.IdRol);
            cmd.Parameters.AddWithValue("Estado", obj.Estado);
            cmd.Parameters.Add("IdUsuarioResultado", SqlDbType.Int).Direction = ParameterDirection.Output;
            cmd.Parameters.Add("Mensaje", SqlDbType.VarChar,500).Direction = ParameterDirection.Output;
            cmd.CommandType = CommandType.StoredProcedure;

            oconexion.Open();

            cmd.ExecuteNonQuery();

            idusuariogenerado = Convert.ToInt32(cmd.Parameters["IdUsuarioResultado"].Value);
            Mensaje = cmd.Parameters["Mensaje"].Value.ToString();

        }
    }
    catch (Exception ex) {
        idusuariogenerado = 0;
        Mensaje = ex.Message;
    }


    return idusuariogenerado;
}
```

También crearemos el método editar que nos va a devolver un booleno de respuesta y que ejecutará el procedimiento SPEDITARUSURIO.

```
public bool Editar(Usuario obj, out string Mensaje)
{
    bool respuesta = false;
    Mensaje = string.Empty;


    try
    {

        using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
        {

            SqlCommand cmd = new SqlCommand("SP_EDITARUSUARIO", oconexion);
            cmd.Parameters.AddWithValue("IdUsuario", obj.IdUsuario);
            cmd.Parameters.AddWithValue("Documento", obj.Documento);
            cmd.Parameters.AddWithValue("NombreCompleto", obj.NombreCompleto);
            cmd.Parameters.AddWithValue("Correo", obj.Correo);
            cmd.Parameters.AddWithValue("Clave", obj.Clave);
            cmd.Parameters.AddWithValue("IdRol", obj.oRol.IdRol);
            cmd.Parameters.AddWithValue("Estado", obj.Estado);
            cmd.Parameters.Add("Respuesta", SqlDbType.Int).Direction = ParameterDirection.Output;
            cmd.Parameters.Add("Mensaje", SqlDbType.VarChar,500).Direction = ParameterDirection.Output;
            cmd.CommandType = CommandType.StoredProcedure;

            oconexion.Open();

            cmd.ExecuteNonQuery();

            respuesta = Convert.ToBoolean(cmd.Parameters["Respuesta"].Value);
            Mensaje = cmd.Parameters["Mensaje"].Value.ToString();

        }
    }
    catch (Exception ex) {
        idusuariogenerado = 0;
        Mensaje = ex.Message;
    }


    return idusuariogenerado;
}
```

Y por último el método de eliminar usuario que igualmente devolverá una respuesta booleana

```
156     public bool Eliminar(Usuario obj, out string Mensaje)
157     {
158         bool respuesta = false;
159         Mensaje = string.Empty;
160
161
162         try
163         {
164
165             using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
166             {
167
168
169                 SqlCommand cmd = new SqlCommand("SP_ELIMINARUSUARIO", oconexion);
170                 cmd.Parameters.AddWithValue("IdUsuario", obj.IdUsuario);
171                 cmd.Parameters.Add("Respuesta", SqlDbType.Int).Direction = ParameterDirection.Output;
172                 cmd.Parameters.Add("Mensaje", SqlDbType.VarChar,500).Direction = ParameterDirection.Output;
173                 cmd.CommandType = CommandType.StoredProcedure;
174
175                 oconexion.Open();
176
177                 cmd.ExecuteNonQuery();
178
179                 respuesta = Convert.ToBoolean(cmd.Parameters["Respuesta"].Value);
180                 Mensaje = cmd.Parameters["Mensaje"].Value.ToString();
181
182             }
183
184         }
185         catch (Exception ex)
186         {
187             respuesta = false;
188             Mensaje = ex.Message;
189         }
190
191         return respuesta;
192     }
193
194
195 }
196 }
```

Ahora desde nuestra capa de negocio vamos a realizar el puente entre la capa de datos y la capa de presentación. La capa de negocio debe comprobar que se cumplen las reglas. Agregaremos algunas validaciones. También se pueden añadir en la capa de presentación, pero en siguiendo el modelo por capas debemos agregar las reglas de negocio.

El método de registrar no debería ejecutarse si no ha pasado las reglas

```
public class CN_Usuario
{

    private CD_Usuario objcd_usuario = new CD_Usuario();


    public List<Usuario> Listar()
    {
        return objcd_usuario.Listar();
    }

    public int Registrar(Usuario obj,out string Mensaje)
    {
        Mensaje = string.Empty;

        if (obj.Documento == "") {
            Mensaje += "Es necesario el documento del usuario\n";
        }

        if (obj.NombreCompleto == "")
        {
            Mensaje += "Es necesario el nombre completo del usuario\n";
        }

        if (obj.Clave == "")
        {
            Mensaje += "Es necesario la clave del usuario\n";
        }

        if (Mensaje != string.Empty)
        {

            return 0;
        }
        else {
            return objcd_usuario.Registrar(obj, out Mensaje);
        }


    }
```

De igual modo el método Editar realizará sus controles:

```
public bool Editar(Usuario obj, out string Mensaje)
{

    Mensaje = string.Empty;

    if (obj.Documento == "")
    {
        Mensaje += "Es necesario el documento del usuario\n";
    }

    if (obj.NombreCompleto == "")
    {
        Mensaje += "Es necesario el nombre completo del usuario\n";
    }

    if (obj.Clave == "")
    {
        Mensaje += "Es necesario la clave del usuario\n";
    }


    if (Mensaje != string.Empty)
    {
        return false;
    }
    else
    {
        return objcd_usuario.Editar(obj, out Mensaje);
    }


}
```

Por último, incorporamos el método de eliminar a nuestra capa de negocio:

```
public bool Eliminar(Usuario obj, out string Mensaje)
{
    return objcd_usuario.Eliminar(obj, out Mensaje);
}
```

Regresamos a nuestro formulario de usuario de la capa de presentación. En el evento de click del botón guardar eliminamos (o comentamos) lo que habíamos programado y generamos el código correcto con acceso a la bbdd. El botón guardar dará de alta un nuevo usuario si no existe y modificará un usuario si ya existe.

```
private void btnguardar_Click(object sender, EventArgs e)
{
    string mensaje = string.Empty;

    Usuario objusuario = new Usuario() {
        IdUsuario = Convert.ToInt32(txtid.Text),
        Documento = txtdocumento.Text,
        NombreCompleto = txtnombrecompleto.Text,
        Correo = txtcorreo.Text,
        Clave = txtclave.Text,
        oRol = new Rol() { IdRol = Convert.ToInt32(((OpcionCombo)cborol.SelectedItem).Valor) },
        Estado = Convert.ToInt32(((OpcionCombo)cboestado.SelectedItem).Valor) == 1 ? true : false
    };

    int idusuariogenerado = new CN_Usuario().Registrar(objusuario, out mensaje);

    if (idusuariogenerado != 0)
    {

        dgvdata.Rows.Add(new object[] {"",idusuariogenerado,txtdocumento.Text,txtnombrecompleto.Text,txtcorreo.Text,txtclave.Text,
        ((OpcionCombo)cborol.SelectedItem).Valor.ToString(),
        ((OpcionCombo)cborol.SelectedItem).Texto.ToString(),
        ((OpcionCombo)cboestado.SelectedItem).Valor.ToString(),
        ((OpcionCombo)cboestado.SelectedItem).Texto.ToString()
        });

        Limpiar();
    }
    else {

        MessageBox.Show(mensaje);

    }
```