

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte V – Formulario Usuarios vs BBDD</b>	CFGS Desarrollo de Aplicaciones Multiplataforma  Módulo: DDI
--	---	--

**Este diseño se corresponde con el video de apoyo número 5**

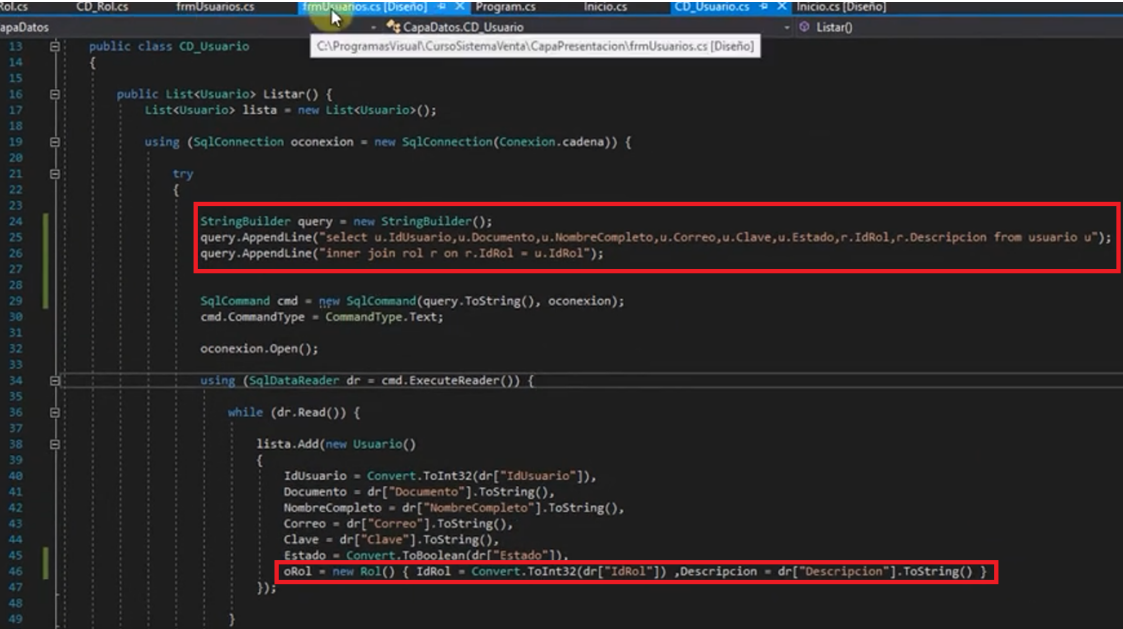
Para las operaciones CRUD de nuestro formulario de usuarios vamos a utilizar la tabla usuario y la tabla rol de nuestra base de datos. Tenemos que recuperar además de los datos de usuario la descripción del rol. La consulta SQL Server podría ser:

use BDSISTEMAVENTAS;

```
SELECT u.IdUsuario, u.Documento, u.NombreCompleto, u.Correo, u.Clave, u.Estado, r.IdRol,
r.Descripcion
FROM usuario u inner join rol r ON u.IdRol = r.IdRol;
```

Modificamos nuestra clase CD\_Usuario, sustituyendo la anterior consulta sql por esta e incorporando los nuevos campos al Data Reader.

Como ya hicimos anteriormente como esta consulta también tiene varias líneas construiremos el string utilizando el método StringBuilder (<https://docs.microsoft.com/es-es/dotnet/api/system.text.stringbuilder?view=net-6.0>) Este método tiene varias propiedades. Append añade el texto al final del objeto StringBuilder actual y AppendLine además de añadir el texto añade el salto de línea predeterminado al final del objeto StringBuilder actual.



```

13 public class CD_Usuario
14 {
15
16     public List<Usuario> listar() {
17         List<Usuario> lista = new List<Usuario>();
18
19         using (SqlConnection oconexion = new SqlConnection(Conexion.cadena)) {
20
21             try
22             {
23                 StringBuilder query = new StringBuilder();
24                 query.AppendLine("select u.IdUsuario,u.Documento,u.NombreCompleto,u.Correo,u.Clave,u.Estado,r.IdRol,r.Descripcion from usuario u");
25                 query.AppendLine("inner join rol r on r.IdRol = u.IdRol");
26
27                 SqlCommand cmd = new SqlCommand(query.ToString(), oconexion);
28                 cmd.CommandType = CommandType.Text;
29
30                 oconexion.Open();
31
32                 using (SqlDataReader dr = cmd.ExecuteReader()) {
33
34                     while (dr.Read()) {
35
36                         lista.Add(new Usuario()
37                         {
38                             IdUsuario = Convert.ToInt32(dr["IdUsuario"]),
39                             Documento = dr["Documento"].ToString(),
40                             NombreCompleto = dr["NombreCompleto"].ToString(),
41                             Correo = dr["Correo"].ToString(),
42                             Clave = dr["Clave"].ToString(),
43                             Estado = Convert.ToBoolean(dr["Estado"]),
44                             rRol = new Rol() { IdRol = Convert.ToInt32(dr["IdRol"]), ,Descripcion = dr["Descripcion"].ToString() }
45                         });
46                     }
47                 }
48             }
49             catch { }
50         }
51     }
52 }

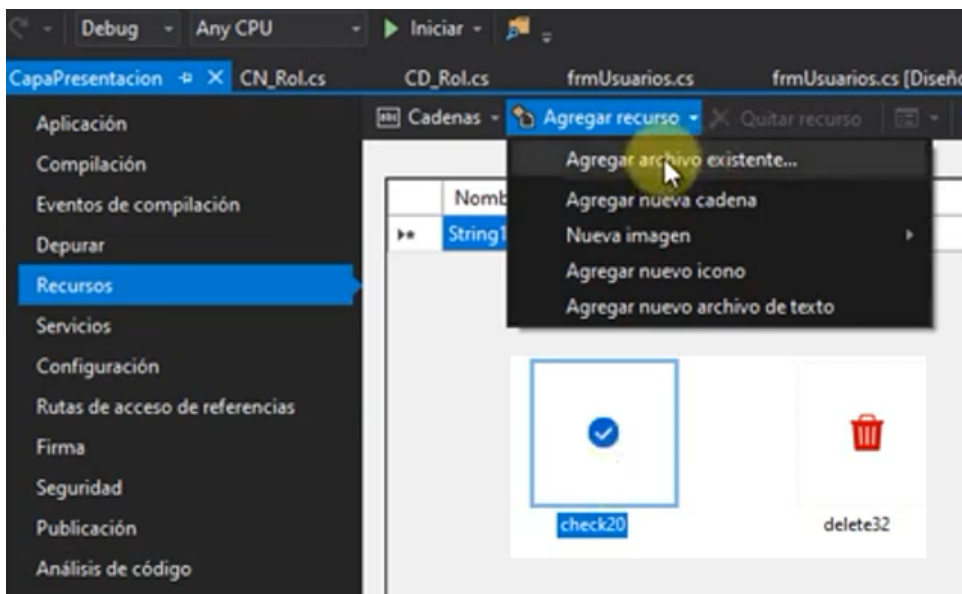
```

En el evento load de nuestro formulario de usuarios programamos la recuperación de la lista de usuarios:

```

55
56
57 //MOSTRAR TODOS LOS USUARIOS
58 List<Usuario> listaUsuario = new CN_Usuario().Listar();
59
60 foreach (Usuario item in listaUsuario)
61 {
62
63     dgvdata.Rows.Add(new object[] { "", item.IdUsuario, item.Documento, item.NombreCompleto, item.Correo, item.Clave,
64         item.oRol.IdRol,
65         item.oRol.Descripcion,
66         item.Estado == true ? 1 : 0,
67         item.Estado == true ? "Activo" : "No Activo"
68     });
69 }
70
71 }
72
73 private void btnGuardar_Click(object sender, EventArgs e)
74 {
75     dgvdata.Rows.Add(new object[] { "", txtid.Text, txtdocumento.Text, txtnombrecompleto.Text, txtcorreo.Text, txtclave.Text,
76         ((OpcionCombo)cborol.SelectedItem).Valor.ToString(),
77         ((OpcionCombo)cborol.SelectedItem).Texto.ToString(),
78         ((OpcionCombo)cboestado.SelectedItem).Valor.ToString(),
79         ((OpcionCombo)cboestado.SelectedItem).Texto.ToString()
80     });
81
82     Limpiar();
83 }
84
    
```

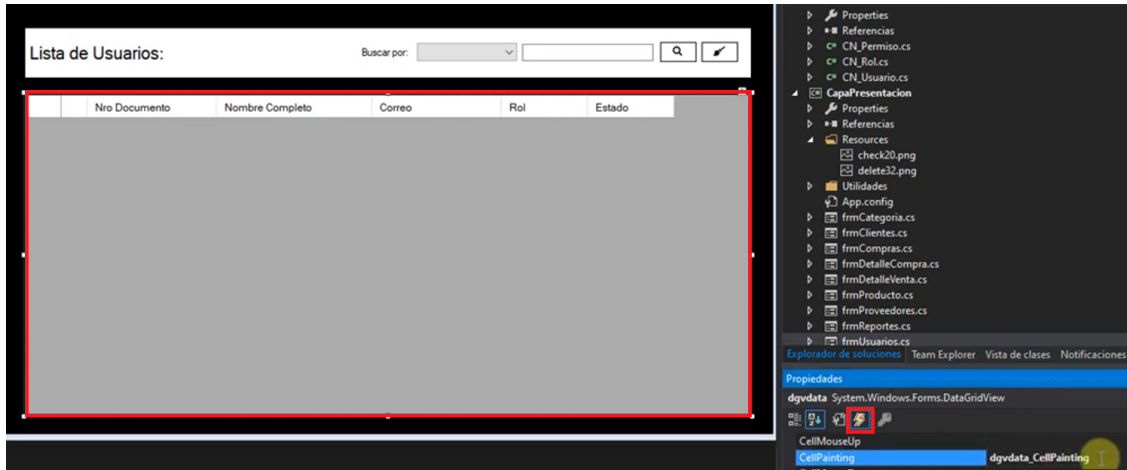
Cuando se seleccione en nuestro grid un usuario vamos a volcar esos datos en el espacio de datos de la izquierda. Esta selección de usuario se realiza pinchando el botón que aparece en la primera columna. Para que sea más visible esta opción incorporamos un icono a este botón. Necesitamos dos imágenes, uno en forma de check de 20x20 y otro en forma de cubo de basura de 32x32 pixeles. Tras guardarlos en nuestro sistema en las propiedades de nuestro proyecto CapaPresentacion los agregamos como recursos archivos ( .png o .jpg). En este caso no son iconos:



Utilizaremos el **evento** CellPainting del grid que tiene lugar cuando es necesario dibujar una celda. Cada vez que agregamos una fila se está dibujando una celda y lo que necesitamos es que cuando se agregue nos dibuje una imagen en la celda.

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte V – Formulario Usuarios vs BBDD</b>	CFGS Desarrollo de Aplicaciones Multiplataforma  Módulo: DDI
--	---	--

En el formulario de usuarios seleccionamos el DataGridView y en sus eventos buscamos CellPainting, al hacer doble clic nos lleva al código del evento:



Lo que necesitamos es centrar dentro del botón nuestra imagen, utilizaremos las propiedades y métodos que nos proporciona la clase DataGridViewCellPainting para el evento CellPainting ( <https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewcellpaintingeventargs?view=windowsdesktop-6.0>)

Define valores para especificar las partes de un [DataGridViewCell](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewcellpaintingeventargs?view=windowsdesktop-6.0) que se van a pintar. = All todas las partes de la celda deben estar pintadas (<https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewcellpaintingeventargs?view=windowsdesktop-6.0>)

```

100 private void dgvdata_CellPainting(object sender, DataGridViewCellPaintingEventArgs e)
101 {
102     if (e.RowIndex < 0)
103         return;
104
105     if (e.ColumnIndex == 0) {
106
107         e.Paint(e.CellBounds, DataGridViewPaintParts.All);
108
109         var w = Properties.Resources.check20.Width;
110         var h = Properties.Resources.check20.Height;
111         var x = e.CellBounds.Left + (e.CellBounds.Width - w) / 2;
112         var y = e.CellBounds.Top + (e.CellBounds.Height - h) / 2;
113
114         e.Graphics.DrawImage(Properties.Resources.check20, new Rectangle(x,y,w,h));
115         e.Handled = true;
116     }
117
118
119
120
121 }

```

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte V – Formulario Usuarios vs BBDD</b>	CFGS Desarrollo de Aplicaciones Multiplataforma  Módulo: DDI
--	---	--

Al seleccionar una fila de nuestro Grid vamos a volcar estos datos en nuestro menú de la izquierda.

Seleccionamos nuestro DataGridView y en sus eventos buscamos CellContentClick y programamos su funcionalidad en caso de que se haya pulsado btnseleccionar:

```

120 private void dgvdata_CellContentClick(object sender, DataGridViewCellEventArgs e)
121 {
122     if (dgvdata.Columns[e.ColumnIndex].Name == "btnseleccionar") {
123
124         int indice = e.RowIndex;
125
126         if (indice >= 0) {
127             txtid.Text = dgvdata.Rows[indice].Cells["Id"].Value.ToString();
128             txtdocumento.Text = dgvdata.Rows[indice].Cells["Documento"].Value.ToString();
129             txtnombrecompleto.Text = dgvdata.Rows[indice].Cells["NombreCompleto"].Value.ToString();
130             txtcorreo.Text = dgvdata.Rows[indice].Cells["Correo"].Value.ToString();
131             txtclave.Text = dgvdata.Rows[indice].Cells["Clave"].Value.ToString();
132             txtconfirmarclave.Text = dgvdata.Rows[indice].Cells["Clave"].Value.ToString();
133
134             foreach (OpcionCombo oc in cborol.Items) {
135
136                 if (Convert.ToInt32(oc.Valor) == Convert.ToInt32(dgvdata.Rows[indice].Cells["IdRol"].Value)) {
137                     int indice_combo = cborol.Items.IndexOf(oc);
138                     cborol.SelectedIndex = indice_combo;
139                     break;
140                 }
141             }
142
143             foreach (OpcionCombo oc in cboestado.Items)
144             {
145                 if (Convert.ToInt32(oc.Valor) == Convert.ToInt32(dgvdata.Rows[indice].Cells["EstadoValor"].Value))
146                 {
147                     int indice_combo = cboestado.Items.IndexOf(oc);
148                     cboestado.SelectedIndex = indice_combo;
149                     break;
150                 }
151             }
152         }
153     }
154 }
155
156
157
158
159

```

Para poder guardar datos de usuario en nuestra bbdd tenemos que crear un procedimiento sql:

```

USE BDSISTEMAVENTAS;
GO
CREATE PROC SP_REGISTRARUSUARIO (
@Documento varchar(50),
@NombreCompleto varchar(50),
@Clave varchar(50),
@IdRol int,
@Estado bit,
@Correo varchar(50),
--Parámetros de salida resultado de operar con el registro usuario
@IdUsuarioResultado int output,
@Mensaje varchar(500) output)
AS
BEGIN
    SET @IdUsuarioResultado=0
    SET @Mensaje=''
    if not exists(select * from USUARIO where Documento=@Documento)
    begin
        insert into USUARIO (Documento, NombreCompleto, Clave, IdRol, Estado,
Correo ) values

```

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte V – Formulario Usuarios vs BBDD</b>	CFGS Desarrollo de Aplicaciones Multiplataforma  Módulo: DDI
--	---	--

```

        (@Documento, @NombreCompleto, @Clave, @IdRol, @Estado, @Correo)

        --id usuario se graba automáticamente y queda almacenado en SCOPE_IDENTITY
        SET @IdUsuarioResultado = SCOPE_IDENTITY()
    end
    ELSE
        SET @Mensaje='Ese usuario ya existe, no se puede repetir, Introduzca otro
distinto.'
    END

```

Podemos comprobar el resultado de este procedimiento ejecutando varias veces esta query:

```

select * from usuario
DECLARE @IdUsuarioRes int
DECLARE @mensa varchar(500)

exec SP_REGISTRARUSUARIO '404040', 'Procedimiento', '12345', 2, 1, '@', @IdUsuarioRes output,
@mensa output

select @IdUsuarioRes as IdUsuario, @mensa as Mensajeresultado
select * from usuario

```

Creamos un nuevo procedimiento de editar usuario:

```

USE BDSISTEMAVENTAS;
GO
CREATE PROC SP_EDITARUSUARIO (
    @Idusuario int,
    @Documento varchar(50),
    @NombreCompleto varchar(50),
    @Clave varchar(50),
    @IdRol int,
    @Estado bit,
    @Correo 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=''
    if not exists(select * from USUARIO where Documento=@Documento and idusuario
!=@IdUsuario)
    begin
        update USUARIO set
            Documento=@Documento,
            NombreCompleto=@NombreCompleto,
            Clave=@Clave,
            IdRol=@IdRol,
            Estado=@Estado,
            Correo=@Correo
        where IdUsuario=@Idusuario

        set @Respuesta=1
    end
    ELSE

```

	<b>UT 2 – <i>Práctica Sistema de Ventas</i></b> <b>Parte V – Formulario Usuarios vs BBDD</b>	CFGS Desarrollo de Aplicaciones Multiplataforma <b>Módulo: DDI</b>
--	---	---

```

        SET @Mensaje='Ese usuario ya existe, no se puede repetir.'
END
Y comprobamos su funcionamiento en SQL Server:
select * from USUARIO

DECLARE @resultado bit
DECLARE @mensa varchar(500)

exec SP_EDITARUSUARIO 3, '303030', 'Procedimiento Editar', '12345', 2, 2, '@', @Resultado
output, @mensa output

select @resultado as resultado, @mensa as Mensajeresultado

select * from USUARIO

```