**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:

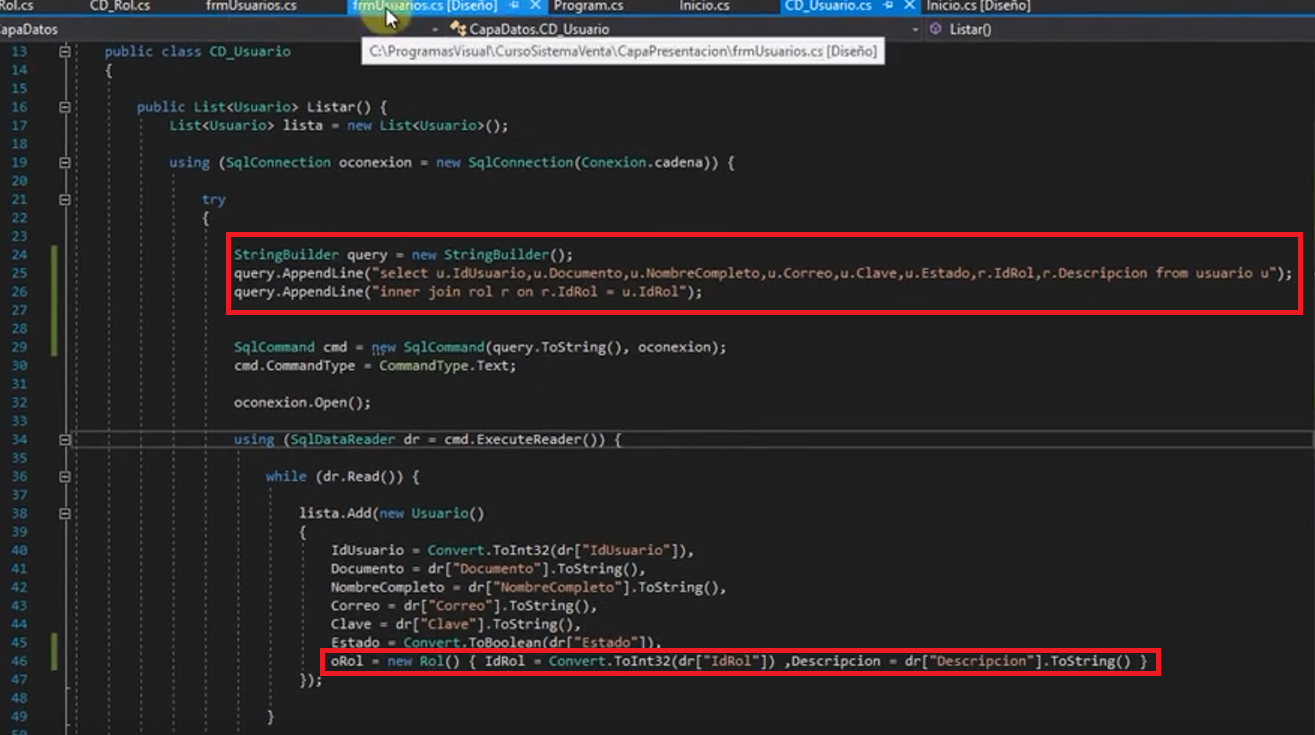
use BDSISTEMAVENTAS;

SELECT u.IdUsuario, u.Documento, u.NombreCompleto, u.Correo, u.Clave, u.Estado, r.IdRol, r.Descripcion

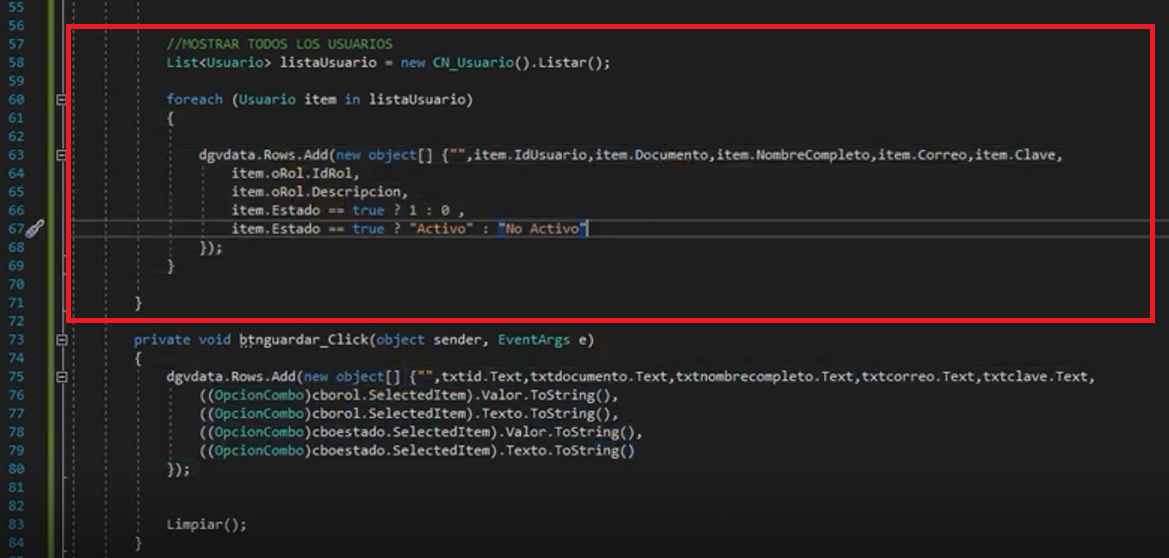
FROM usuario u inner join rol r ON u.idRol = r.IdRol;

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.

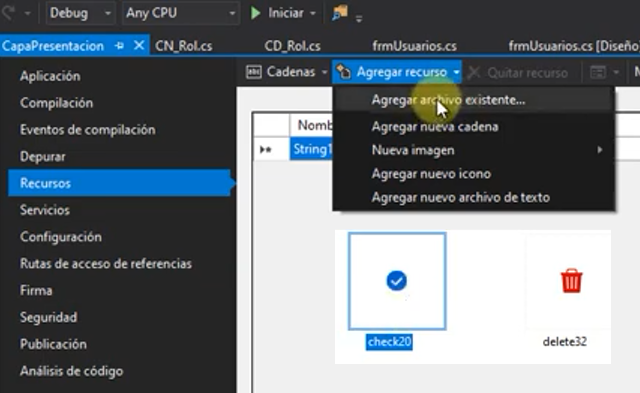


En el evento load de nuestro formulario de usuarios programamos la recuperación de la lista de usuarios:



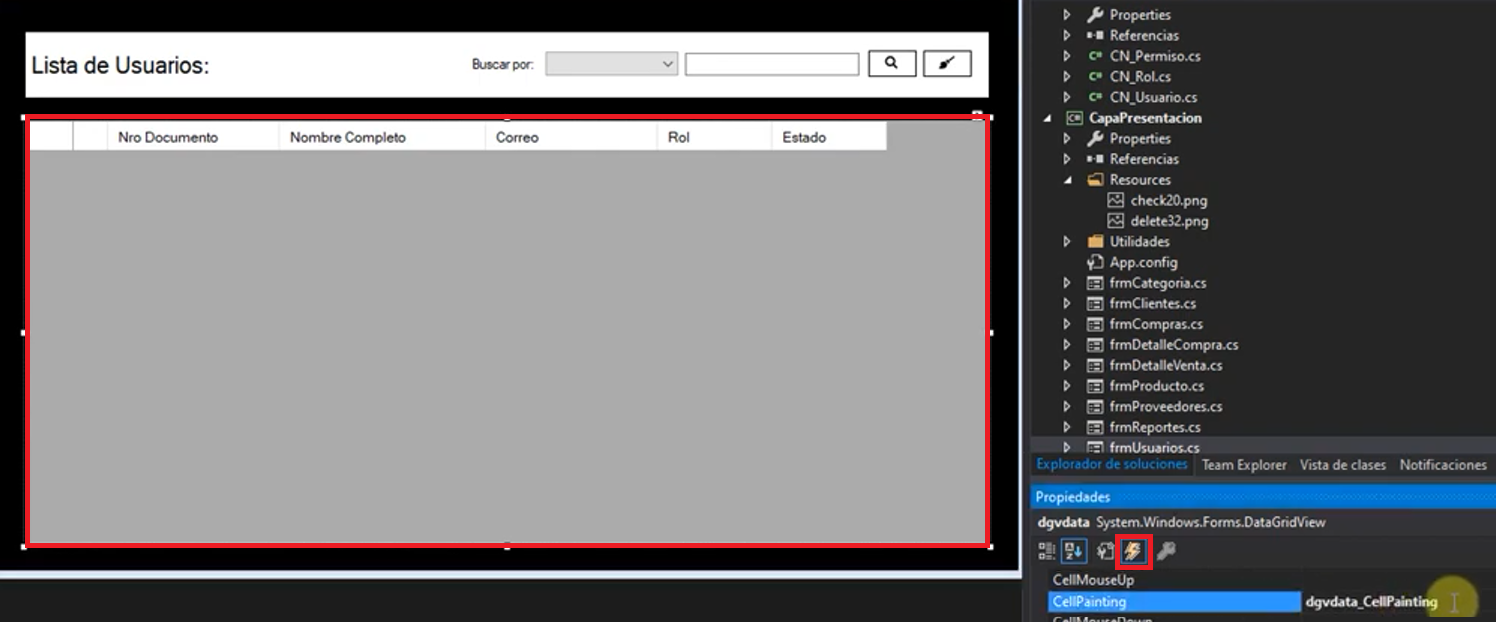
Cuando se seleccione en nuestro grid un usuario vamos a volcar esos datos en el espacio de datos de la izquierda. Esta selección de usuario se realiza pinchando el botón que aparece en la primera columna. Para que sea más visible esta opción incorporamos un icono a este botón.

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:



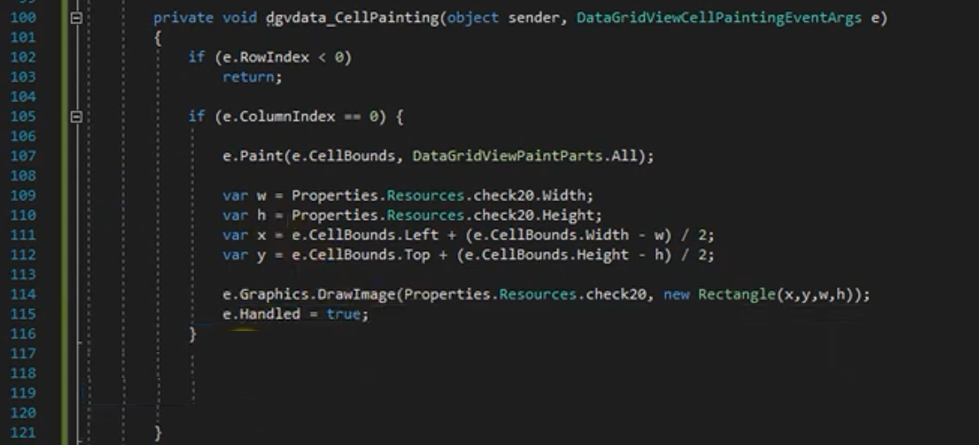
Utilizaremos el **evento** CellPainting del grid que tiene lugar cuando es necesario dibujar una celda. Cada vez que agregamos una fila se está dibujando una celda y lo que necesitamos es que cuando se agregue nos dibuje una imagen en la celda.

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



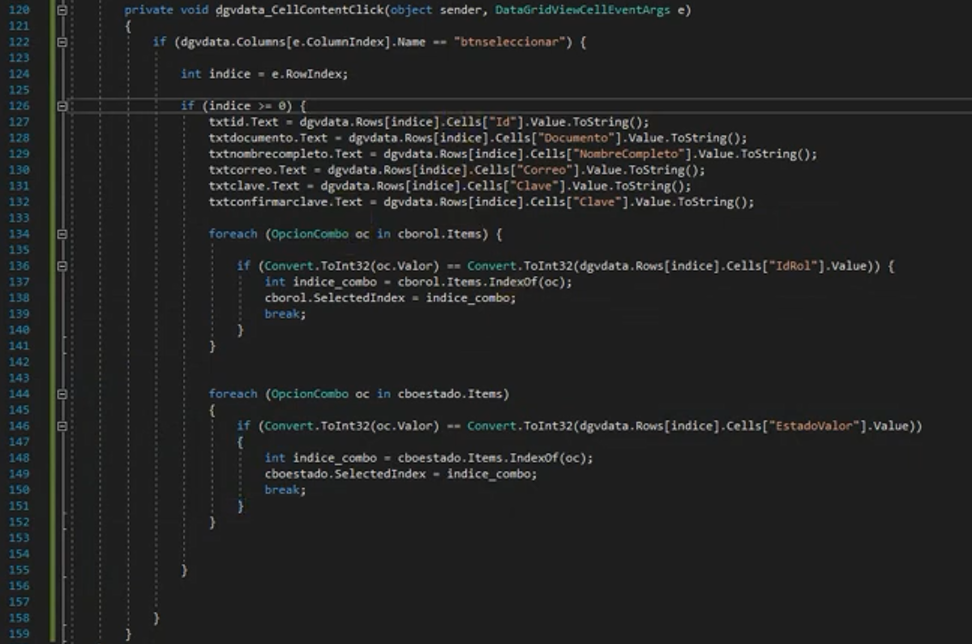
Lo que necesitamos es centrar dentro del botón nuestra imagen, utilizaremos las propiedades y métodos que nos proporciona la clase DataGridViewCellPainting para el evento CellPainting ( <https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewcellpaintingeventargs?view=windowsdesktop-6.0>)

Define valores para especificar las partes de un [DataGridViewCell](https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.datagridviewcell?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.datagridviewpaintparts?view=windowsdesktop-6.0>)



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:



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

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

--id usuario se graba automáticamete 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

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