Este diseño se corresponde con el video de apoyo número 4.

Incorporamos estos elementos a nuestro frmUsuarios:

Un label con **autosize** false **dock** left **text** vacio **backcolor** blanco **bordersyle** fixedsingle, que va a contener nuestros campos de usuario que tienen correspondencia con nuestra base de datos.

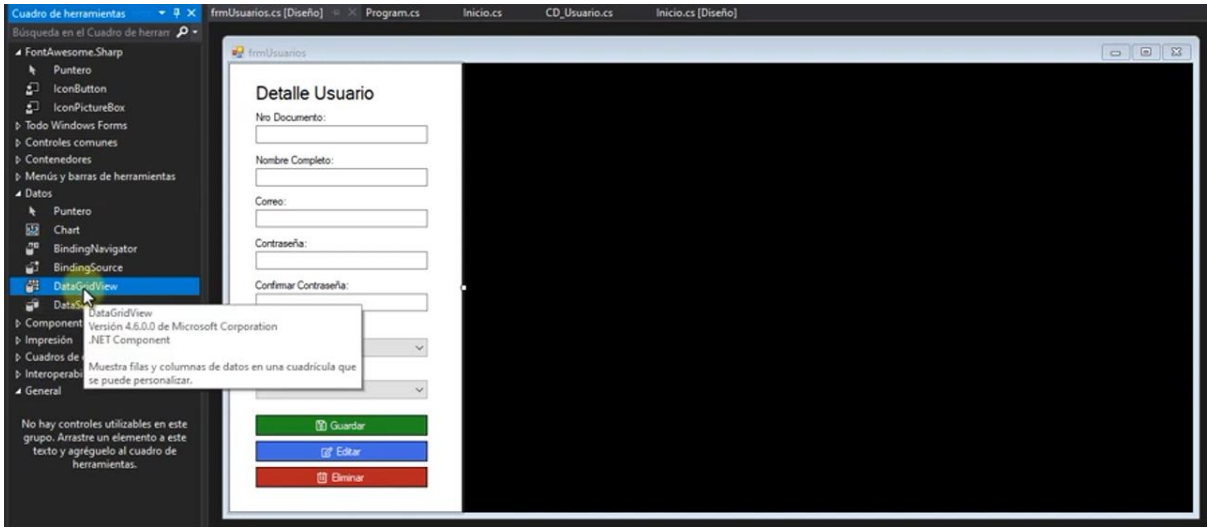
Dentro de este label situamos un **label** que visualizará **Detalle usuario** con tamaño letra 15 **y otros siete** que tienen que tener fondo blanco. El primero tiene que visualizar Usuario el segundo Nombre Completo, el tercero Correo, el cuarto Contraseña, el quinto Confirmar contraseña, el sexto Rol y el séptimo Estado.

Los cinco primeros encabezarán a **cinco textbox** que se llamarán txtdocumento, txtnombrecompleto, txtcorreo, txtclave, txtconfirmarclave.

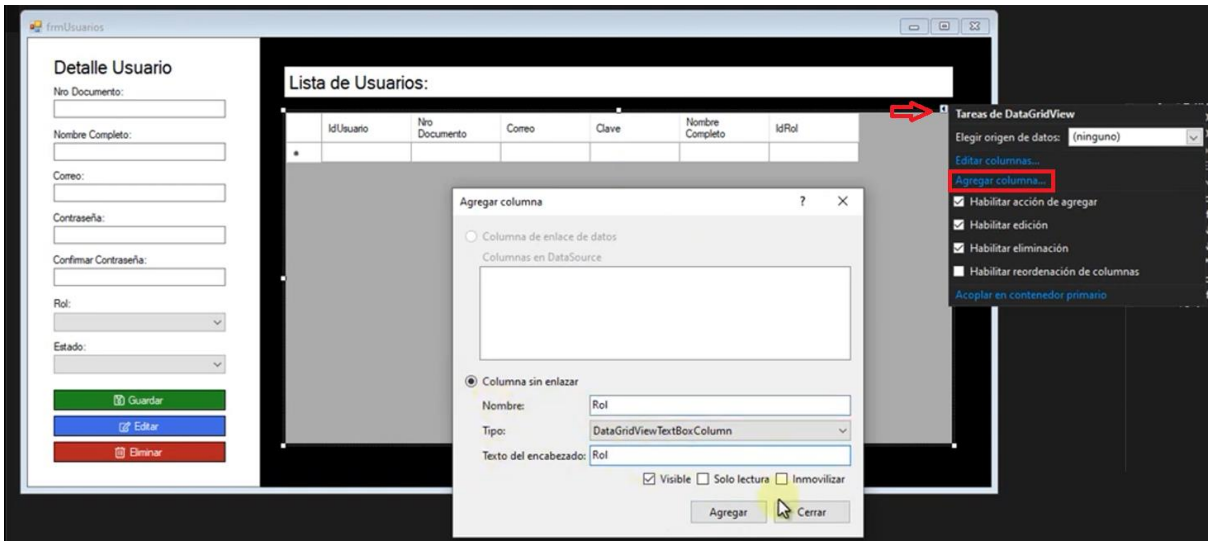
Los dos últimos encabezarán **dos combobox** cborol y cboestado tendrán su propiedad **DropDownStyle** a DropDownList para que no se puedan introducir datos en estos campos ya que visualizaremos una lista de opciones.

Desde herramientas FontAweresome arrastramos **tres botones** (btnguardar btnlimpiar y btneliminar) con **texto** Guardar, Limpiar y Eliminar, **IconChar** Save, Broom y TrashAlt y **backcolor**

ForestGreen, RoyalBlue y Firebrick respectivamente con propiedades **IconColor** blanco, **IconSize** 18, **TextAlign** MiddleRight, **TextImageRelacion** ImageBeforeText, **Cursor** en forma de mano, **FlatStyle** (apariencia) en modo flat, **ForeColor** (color fuente) blanco, dentro de FlatApparece la propiedad **bordercolor** black, tamaño iconos 18.
Arrastramos un DataGridView que llamaremos dgvdata a la parte derecha de nuestro formulario:

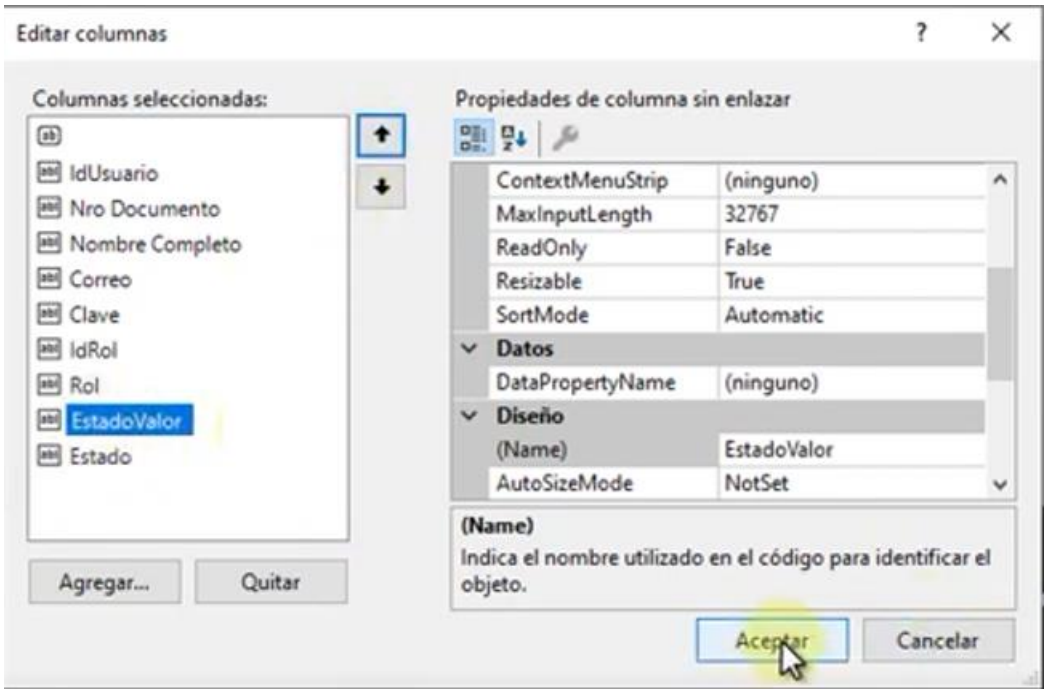


En su cabecera ponemos una etiqueta Lista de Usuarios: con, **autosize**=false para redimensionarlo, **textalign** a la izquierda, **size**=15 y BorderStyle=FixedSingle
Agregamos las columnas a nuestro DataGridView,



Y podemos editarlas si alguna propiedad no está bien establecida (IdUsuario y Clave no tiene que ser visible), ordenarlas y añadir nuevas columnas Estado valor y un botón btnSeleccionar

(propiedad ColumnType=DataGridViewButtonColumn) que nos va a servir para elegir un usuario.



Ajustamos el ancho de nuestras columnas visibles a 30. 150, 180, 150, 100, 100.
Cambiamos algunas propiedades de nuestro DataGridView. **AllowUserToAddRows** = false, **ColumnHeaderDefaultCellStyle** => Size=9, Padding 2 (all) **MultiSelect** = false, **ReadOnly** = true. **RowDefaultCellStyle** =>SelectionBackColor=blanco SelectionForeColor=negro. Row Templete **Height** (alto columna)=28

Por último, incorporamos el textbox txtId en nuestro menú de la derecha que va a contener el Id de usuario seleccionado de forma oculta (visible=false) inicializada con valor 0 y el textbox txtindice también oculto que va a contener la fila seleccionada del Data Grid (incializarla con el valor -1)

FUNCIONALIDAD DATA GRID VIEW

Para dar de alta un nuevo usuario se introducirán los valores en el menú "Detalle Usuario" y al pulsar Guardar si son válidos se almacenarán en nuestra base de datos y se visualizan en nuestro DataGridView.

Para poder obtener los Roles de nuestra BBDD (mismo proceso que recuperar usuarios) creamos las clases correspondientes en nuestra capa de datos y capa de negocio:

```
CapaDatos                                   CapaDatos.CD_Rol                    Listar(int idusuario)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Data;
using System.Data.SqlClient;
using CapaEntidad;


namespace CapaDatos
{
    public class CD_Rol
    {
        public List<Rol> Listar()
        {
            List<Rol> lista = new List<Rol>();

            using (SqlConnection oconexion = new SqlConnection(Conexion.cadena))
            {
                try
                {
                    StringBuilder query = new StringBuilder();
                    query.AppendLine("select IdRol,Descripcion from ROL");

                    SqlCommand cmd = new SqlCommand(query.ToString(), oconexion);
                    cmd.CommandType = CommandType.Text;

                    oconexion.Open();

                    using (SqlDataReader dr = cmd.ExecuteReader())
                    {
                        while (dr.Read())
                        {
                            lista.Add(new Rol()
                            {
                                IdRol = Convert.ToInt32(dr["IdRol"]),
                                Descripcion = dr["Descripcion"].ToString()
                            });
                        }
                    }
                }
                catch (Exception ex)
                {

                    lista = new List<Rol>();
                }
            }

            return lista;

        }

    }
}
```

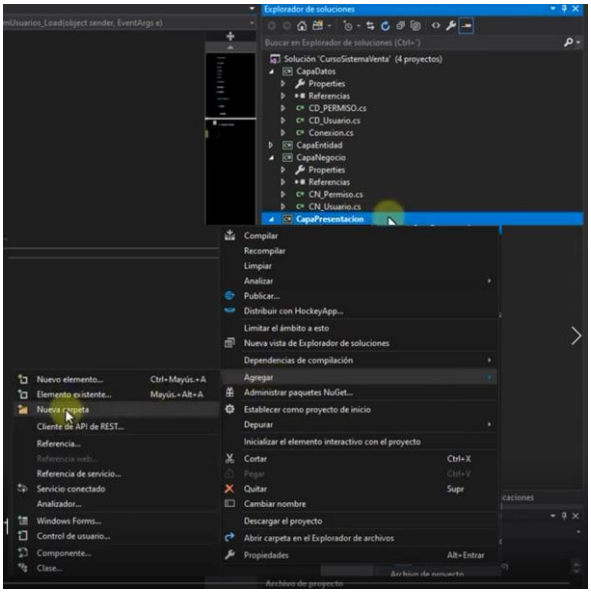
```
CN_Rol.cs    CD_Rol.cs    frmUsuarios.cs    frmUsuarios.cs [Diseño]
CapaNegocio                                 CapaNegocio.CN_Rol
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using CapaDatos;
using CapaEntidad;

namespace CapaNegocio
{
    public class CN_Rol
    {

        private CD_Rol objcd_rol = new CD_Rol();


        public List<Rol> Listar()
        {
            return objcd_rol.Listar();
        }
    }
}
```

En los combos de Rol y Estado se deben visualizar los valores permitidos. Este proceso se deber realizar en el evento Load de nuestro formulario inicio.

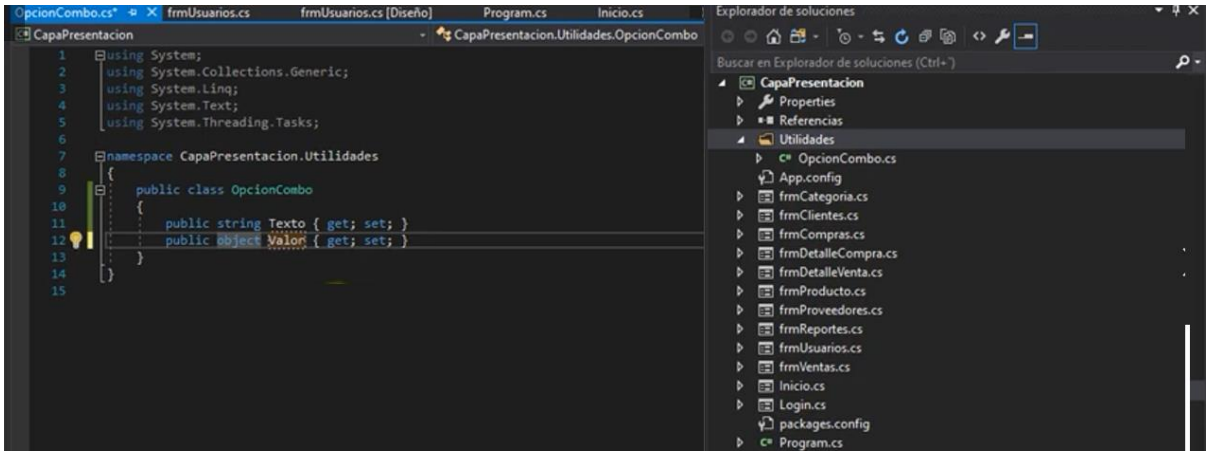
Los combobox trabajan con ítems. Podemos cargar estos combobox de diferentes maneras: manualmente, con arrays, diccionarios o con objetos.

Vamos a cargar nuestros ComboBox con objetos que van a tener dos propiedades texto y valor, el texto se va a visualizar en el combo y el valor va a ser un dato interno. Para ello crearemos una nueva clase que nos servirá para este propósito.

En nuestra capa presentación creamos una carpeta llamada Utilidades y dentro de ella creamos una nueva clase OpcionCombo que nos va a servir para manejar los ítems del combo



Dentro de la carpeta agregamos una nueva clase pública OpcionCombo que va a tener dos valores, uno visible en el combo con un literal y otro interno con un valor asociado al literal (misma funcionalidad que los diccionarios que hemos trabajado).



Introducimos en el evento de Load el código para visualizar los valores correspondientes.

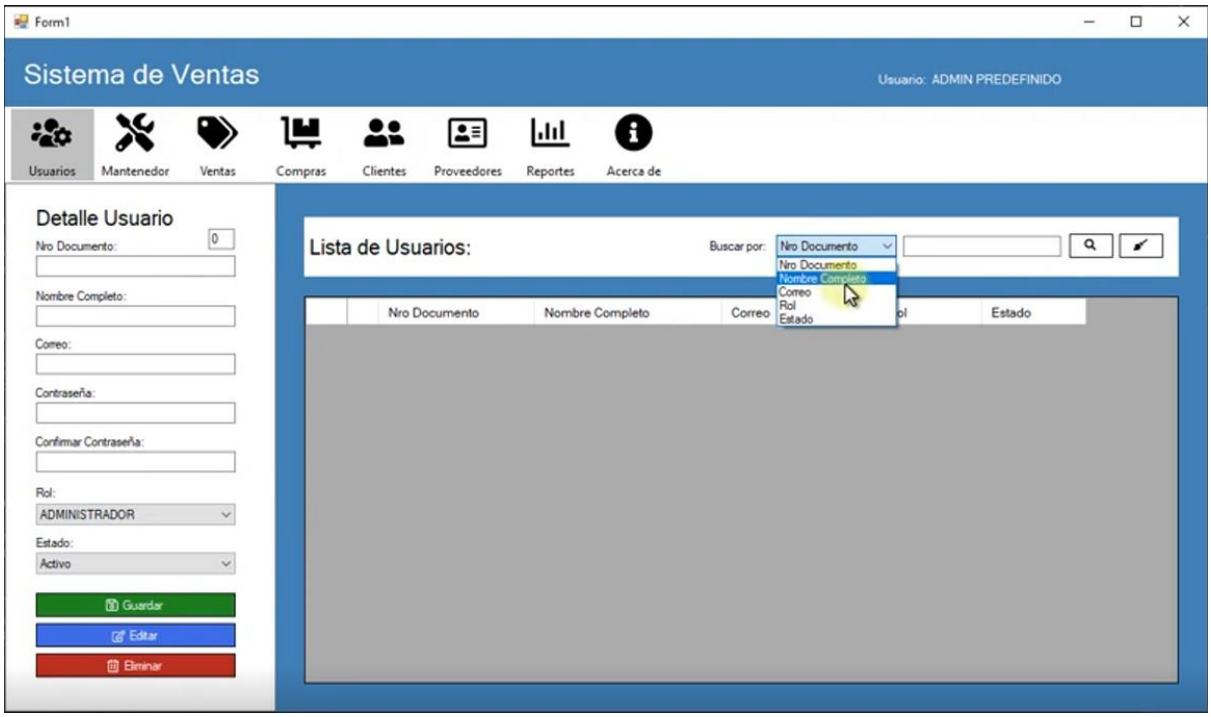
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using CapaPresentacion.Utilidades;

using CapaEntidad;
using CapaNegocio;

namespace CapaPresentacion
{
    public partial class frmUsuarios : Form
    {
        public frmUsuarios()
        {
            InitializeComponent();
        }


        private void frmUsuarios_Load(object sender, EventArgs e)
        {
            cboestado.Items.Add(new OpcionCombo() { Valor = 1 , Texto = "Activo" });
            cboestado.Items.Add(new OpcionCombo() { Valor = 0 , Texto = "No Activo" });
            cboestado.DisplayMember = "Texto";
            cboestado.ValueMember = "Valor";
            cboestado.SelectedIndex = 0;


            List<Rol> listaRol = new CN_Rol().Listar();

            foreach (Rol item in listaRol) {
                cborol.Items.Add(new OpcionCombo() { Valor = item.IdRol, Texto = item.Descripcion });
            }
            cborol.DisplayMember = "Texto";
            cborol.ValueMember = "Valor";
            cborol.SelectedIndex = 0;


        }


    }
}
```

Para que nos pinte los datos introducidos en el DataGrid al pinchar el botón Guardar:

```
private void btnguardar_Click(object sender, EventArgs e)
{
    dgvdata.Rows.Add(new object[] {"",txtid.Text,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();
}
```

```
private void Limpiar() {

    txtindice.Text = "-1";
    txtid.Text = "0";
    txtdocumento.Text = "";
    txtnombrecompleto.Text = "";
    txtcorreo.Text = "";
    txtclave.Text = "";
    txtconfirmarclave.Text = "";
    cborol.SelectedIndex = 0;
    cboestado.SelectedIndex = 0;

    txtdocumento.Select();

}
```

Para poder filtrar dentro del data grid vamos a incorporar dos elementos nuevos a nuestro formulario de usuario, un combo llamado cbobusqueda que va a contener los nombres de nuestras columnas (Headertext) del DataGridView y un textbox txtbusqueda y dos botones btnbuscar y btnlimpiarbuscador con IconChar Search y Broom, TextImageRelation=Overlay, back color=blanco, IconColor=black, Flat Style=flat).



Cargamos el combo cbobusqueda:

```
private void frmUsuarios_Load(object sender, EventArgs e)
{
    cboestado.Items.Add(new OpcionCombo() { Valor = 1 , Texto = "Activo" });
    cboestado.Items.Add(new OpcionCombo() { Valor = 0 , Texto = "No Activo" });
    cboestado.DisplayMember = "Texto";
    cboestado.ValueMember = "Valor";
    cboestado.SelectedIndex = 0;


    List<Rol> listaRol = new CN_Rol().Listar();

    foreach (Rol item in listaRol) {
        cborol.Items.Add(new OpcionCombo() { Valor = item.IdRol, Texto = item.Descripcion });
    }
    cborol.DisplayMember = "Texto";
    cborol.ValueMember = "Valor";
    cborol.SelectedIndex = 0;


    foreach (DataGridViewColumn columna in dgvdata.Columns) {

        if (columna.Visible == true && columna.Name != "btnseleccionar") {
            cbobusqueda.Items.Add(new OpcionCombo() { Valor = columna.Name, Texto = columna.HeaderText});
        }
    }
    cbobusqueda.DisplayMember = "Texto";
    cbobusqueda.ValueMember = "Valor";
    cbobusqueda.SelectedIndex = 0;


}
```