TP2 - Laboratorio 04

1. Crear una ventana con grilla en la capa de presentación de escritorio

Objetivos

Crear en capa de presentación para una aplicación de escritorio un listado de usuarios. Utilizando las capas que ya creamos anteriormente.

Duración Aproximada

30 minutos

Pasos

1. Ir a la carpeta C:\Net\Labs y crear la carpeta TP2L04.

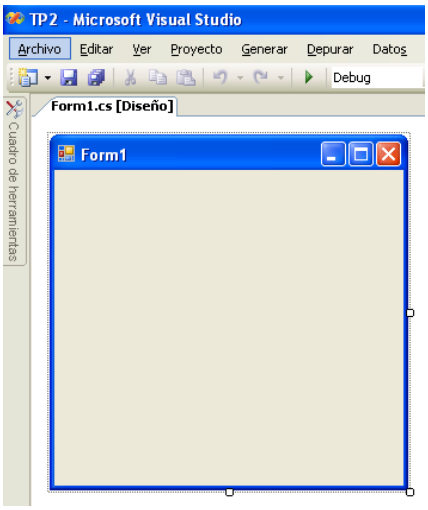
2. Copiar la carpeta TP2 que se encuentra en C:\Net\Labs\TP2L03\ a C:\Net\Labs\TP2L04\
   De esta forma modificaremos esta copia manteniendo la versión anterior intacta.

3. Dentro de la carpeta TP2 que acabamos de copiar extraemos el contenido del archivo Data.Database.zip que se encuentra junto con este enunciado. Este es un proyecto de tipo Librería de Clases que simulará una capa de datos. Podemos eliminar este archivo luego de extraerlo.

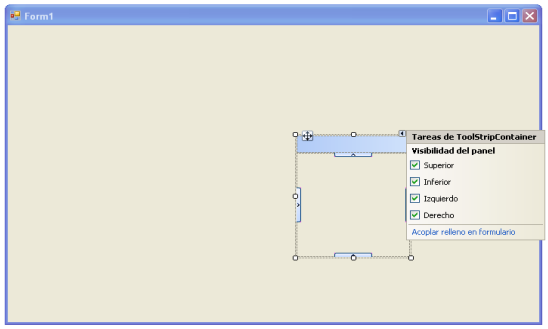
4. Hacemos doble clic sobre el archivo TP2.sln para abrir la solución

5. En el explorador de soluciones en el proyecto UI.Desktop expandimos las referencias y controlamos que existan referencias a los proyectos Business.Logic y Business.Entities. Si no existen agregarlas.

6. En el explorador de soluciones, dentro del proyecto UI.Desktop hacemos doble clic sobre el Form1.cs y se abrira el diseñador de formularios mostrandonos un Windows Form como el siguiente:



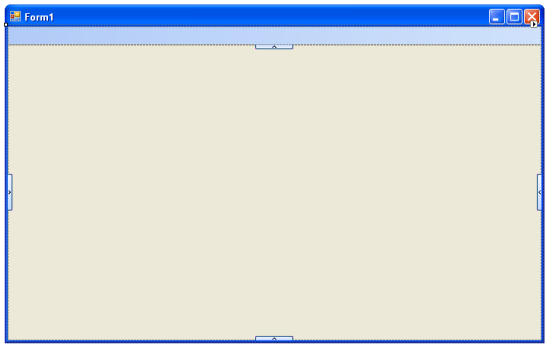
7. Ahora modificaremos el Formulario haciéndolo más grande y luego arrastraremos de la pestaña de la derecha (Cuadro de herramientas o Toolbox)un ToolStripContainer que nos permitirá agregar una barra de herramientas y contenido sin que estos se superpongan.



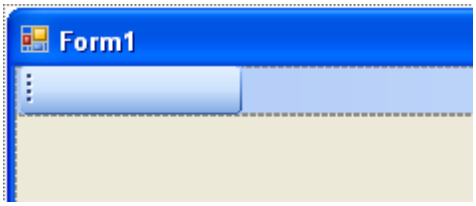
8. Teniendo seleccionado el ToolStripContainer vamos a la pestaña de la derecha llamada Propiedades y vemos que se despliega la lista de todas las propiedades que podemos editar desde el diseñador de formularios. A este objeto le modificaremos las siguientes propiedades

Propiedad	**Nuevo Valor**
(Name)	tcUsuarios
Dock	Fill

La propiedad name nos permite definir con que nombre haremos referencia al control dentro de nuestro código y la propiedad Dock nos permitirá determina como se ajusta el panel dentro del control que lo contiene (en este caso el formulario). Ahora Se verá de esta forma:

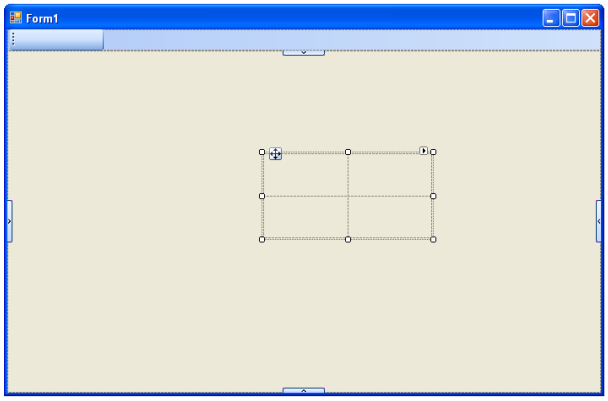


9. Arrastraremos un ToolStrip a la parte superior de tscUsuarios para que se vea así:



10. Al toolStrip1 le modificaremos su (Name) por tsUsuarios. Luego continuaremos editando este elemento.

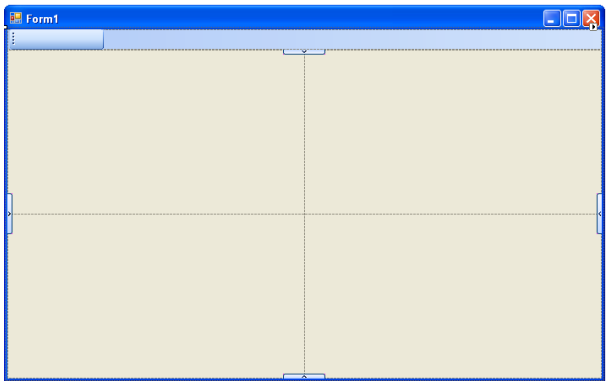
11. Arrastraremos un TableLayoutPanel dentro del tcUsuarios



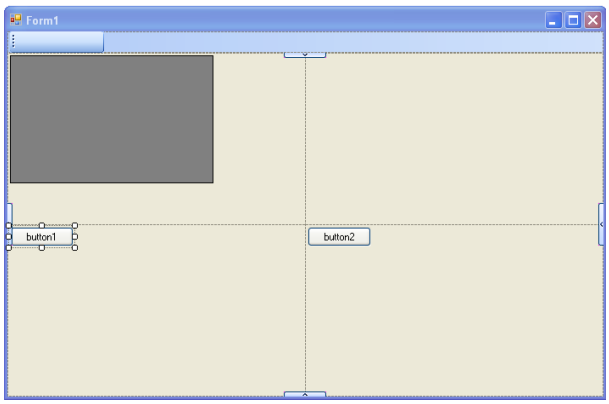
12. Allí modificaremos las siguientes propiedades al tableLayoutPanel1

Propiedad	**Nuevo Valor**
(Name)	tlUsuarios
Dock	Fill

Entonces se verá así:



13. Ahora arrastraremos una DataGridView a la zona superior izquierda y dos Button, uno a cada una de las regiones inferiores.



14. Ahora hacemos clic sobre el botón de la izquierda (button1) y modificamos las propiedades de la siguiente forma

Propiedad	Nuevo Valor
(Name)	btnActualizar
Text	Actualizar

15. Hacemos clic en el botón de la derecha (button2) y en la ventana de propiedades las modificamos de la siguiente forma:

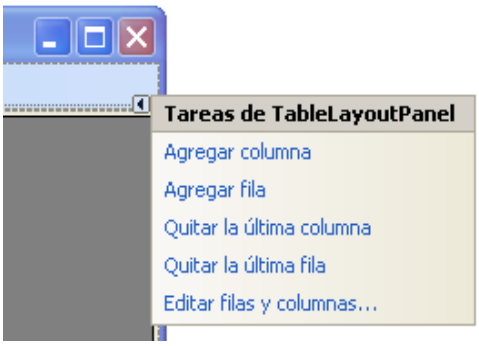
Propiedad	Nuevo Valor
(Name)	btnSalir
Text	Salir

16. Luego hacemos clic sobre la grilla y modificamos las propiedades para que queden de esta forma:

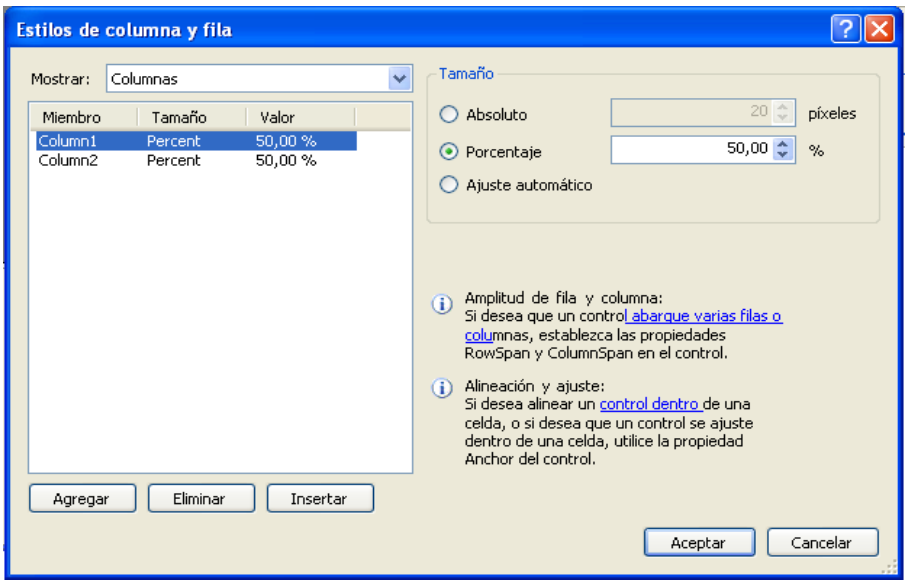
Propiedad	Nuevo Valor
(Name)	dgvUsuarios
Dock	Fill
ColumnSpan	2

Al setear el Dock en Fill vemos que la grilla ocupa todo el espacio que le queda asignado dentro del TableLayoutPanel. Por defecto es 1 fila (RowSpan) y 1 columna (ColumnSpan). Sin embargo como pretendemos que la grilla ocupe todo el ancho indicamos que ocupe 2 columnas de ancho y la propiedad Dock en Fill hará que esta rellene ambas columnas de la fila.

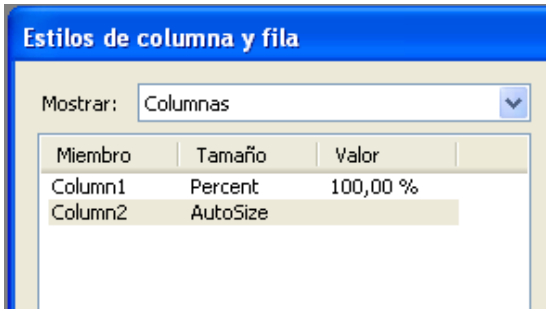
17. Ahora modificaremos las columnas y filas del TableLayoutPanel (tlUsuarios) para mejorar la estética y permitir que al redimiensionar el formulario se siga viendo correctamente la pantalla. Hacemos clic sobre el tlUsuario y y en la parte superior derecha de tlUsuarios se encuentra un pequeño cuadrado blanco con una punta flecha negra. Esto se llama smart tag hacemos clic en él (cuidado que no estemos haciendo clic sobre el smart tag de la grilla, para ello tener en cuenta que al desplegar el smart tag diga Tareas de TableLayoutPanel en negrita) y se desplegarán varias opciones.



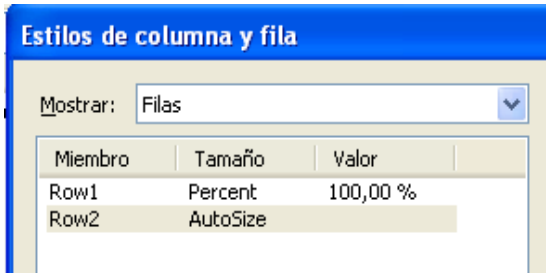
18. Entonces hacemos clic sobre Editar filas y columnas... y se abre la siguiente ventana



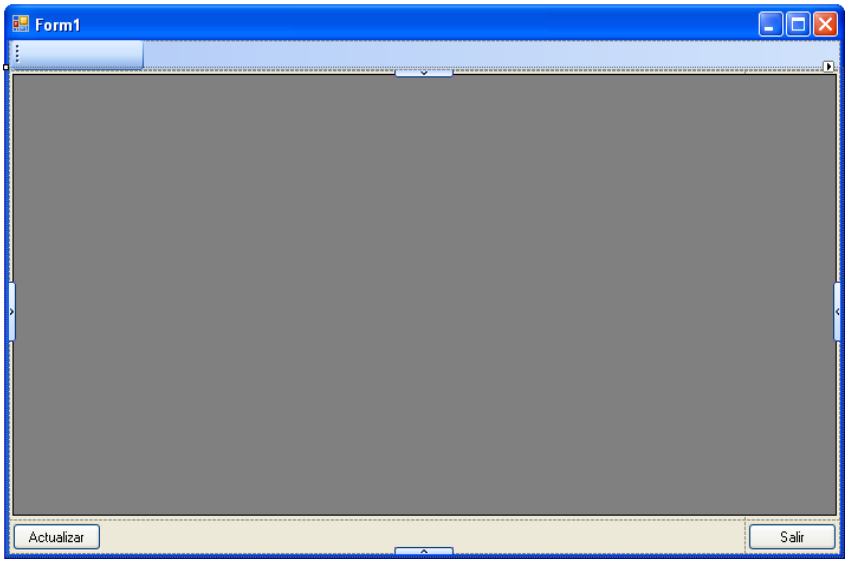
19. Lo que pretendemos es que la segunda columna se adapte automaticamente al tamaño de los controles que contenga (en este caso el botón Salir y que la columna 1 se expanda para cubrir todo el ancho. Para ello, teniendo seleccionada Column1 en la lista de la izquierda vamos al sector de la derecha y donde dice Porcentaje escribimos 100%. Luego hacemos clic en Column2 y en la parte de la derecha elegimos ajuste automático. Debería quedar así:



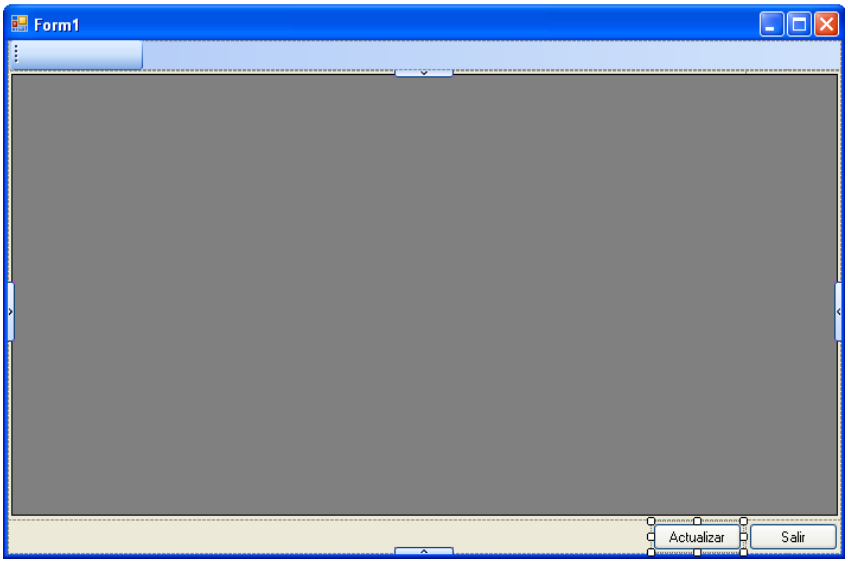
20. Luego en el combo de arriba seleccionamos Filas y las modificaremos igual que las columnas para que queden así:



21. Ahora presionamos Aceptar y el formulario debería verse así:



22. Para que el botón Actualizar quede alineado a la derecha. Hacemos clic sobre actualizar. En la ventana de propiedades vamos a donde dice Anchor. Hacemos clic sobre la flecha que se despliega y dejamos que se quede fijado a la parte superior y a la derecha. Al presionar enter debera decir Top, Right y verse así:



23. En el explorador de soluciones hacemos clic con el botón derecho sobre UI.Desktop y lo establecemos como proyecto de inicio. Luego ejecutamos y podremos ver como se ajustan los controles al cambiar el tamaño del formulario.

24. Ahora haremos clic con el botón derecho sobre el formulario y luego clic en Ver Código.

25. Se abrirá el editor de código del formulario. Agregamos los using a Business.Entities y Business.Logic.

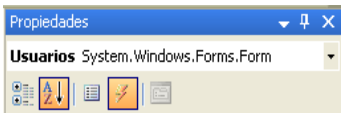
26. Como podremos observar el nombre de la clase es Form1. Entonces procederemos a cambiarlo. Primero nos aseguraremos que el proyecto compile, entonces hacemos clic con el botón derecho sobre Form1, allí nos dirigimos a Refactor y luego clic sobre Cambiar Nombre y la llamaremos Usuarios. Entonces cambiará el nombre de la clase.
    Sin embargo si vamos al explorador de soluciones vemos que el archivo sigue llamándose Form1.

27. Para renombrar el archivo hacemos clic con el botón derecho sobre el archivo en el explorador de soluciones y clic en Cambiar nombre y lo llamaremos Usuarios.cs (Cuidando de no olvidar la extensión del archivo).

28. A continuación crearemos el método public void Listar(). En este método crearemos una nueva instancia de UsuarioLogic, a continuación invocaremos el método GetAll() sobre dicha instancia y asignaremos el resultado a la propiedad DataSource de la grilla. Quedando el código como el siguiente:

```
public void Listar()
{
    UsuarioLogic ul = new UsuarioLogic();
    this.dgvUsuarios.DataSource = ul.GetAll();
}
```

29. Acto seguido volvemos al diseñador de formularios. Seleccionamos el formulario y en la ventana de propiedades cambiamos la propiedad Text por Usuarios. Y luego en la parte superior hacemos clic sobre el botón de eventos. Ahora en lugar de ver la lista de propiedades, estamos visualizando la listas de eventos del control seleccionado.



30. Buscamos el Evento Load y hacemos doble clic sobre el. Automáticamente generará el manejador del evento y el método Usuarios_Load para manejarlo.

31. En el evento Load invocaremos al método Listar().

32. Luego en el diseñador de formularios seleccionamos el botón actualizar y en la lista de eventos hacemos doble clic sobre el evento Click. Automáticamente se generará el manejador del evento y el método Actualizar_Click que será invocado por dicho manejador. En este método también invocaremos a Listar().
    Aclaración: Actualmente los sistemas son multiusuario y por lo tanto los datos pueden ser accedidos y modificados por varios usuarios en forma simultanea. Esto hace necesario que resulte conveniente contar con un mecanismo para que un usuario pueda actualizar la información que ve en pantalla, ya que la misma pudo haber sufrido cambios a mano de otro usuario.

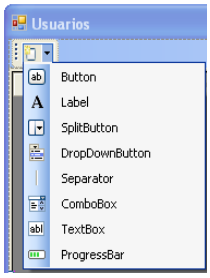
Para esto pueden utilizase diversos mecanismos. Desde los más simples como cerrar la ventana y volverla a abrir, hacerlo mediante un botón de actualizar o que el sistema lo haga en forma automática ya sea por tiempo o al ocurrir un evento. Aquí se eligió este mecanismo por ser fácil de implementar y comprender.

33. Luego en el diseñador de formularios en el botón Salir agregamos el evento Click y en el código del mismo escribimos this.Close();

34. Utilizando lo aprendido en el laboratorio de la unidad 5 de manipulación de la grilla. Modificamos las propiedades de la misma para que:
    - Sea de solo lectura
    - No permita agregar ni eliminar filas manualmente
    - No agregue columnas automáticamente (propiedad AutoGenerateColumns)
    - Cuente con las siguientes columnas:

Nombre Columna	Cabecera	Origen de datos	Tipo Columna
id	ID	ID	Texto
nombre	Nombre	Nombre	Texto
apellido	Apellido	Apellido	Texto
usuario	Usuario	NombreUsuario	Texto
email	EMail	Email	Texto
habilitado	Habilitado	Habilitado	Check

35. Hacemos clic sobre el ToolStrip tsUsuarios. Aparecerá un botón con una lista desplegable, allí elegimos Button



36. Aparecerá un botón con un ícono genérico. En el mismo modificar las siguientes propiedades.

Propiedad	Nuevo Valor
(Name)	tsbNuevo
ToolTipText	Nuevo

ToolTipText es el texto de ayuda que aparece sobre el control cuando ponemos el mouse sobre él.

37. Reemplazar la imagen por otra acorde a la acción utilizando la propiedad Image

38. Repetir la operación para los botones Editar y Eliminar.

2. Crear un formulario de ABM para el escritorio

Objetivos

Crear una capa de presentación para una aplicación de escritorio utilizando las mimas capas de negocio, datos y entidades que ya creamos.

Duración Aproximada

60 minutos

Pasos

1. Crear un nuevo windows form llamado ApplicationForm en el proyecto UI.Desktop. Este será el formulario básico del cuál heredarán todos los formularios que usemos para altas, bajas y modificaciones. En el Pondremos todo el comportamiento común a este tipo de formularios.

2. Definimos una enumeración publica llamada ModoForm con los valores: Alta, Baja, Modificacion y Consulta. Y una propiedad también pública Modo del tipo ModoForm.

3. Allí agregamos los métodos:

```
public virtual void MapearDeDatos(){}
public virtual void MapearADatos(){}
public virtual void GuardarCambios(){}
public virtual bool Validar() { return false; }

public void Notificar(string titulo, string mensaje, MessageBoxButtons
botones, MessageBoxIcon icono)
    {
         MessageBox.Show(mensaje, titulo, botones, icono);
     }

public void Notificar(string mensaje, MessageBoxButtons botones,
MessageBoxIcon icono)
     {
         this.Notificar(this.Text, mensaje, botones, icono);
     }
```

   MapearDeDatos va a ser utilizado en cada formulario para copiar la información de las entidades a los controles del formulario (TextBox, ComboBox, etc) para mostrar la infromación de cada entidad

   MapearADatos se va a utilizar para pasar la información de los controles a una entidad para luego enviarla a las capas inferiores

   GuardarCambios es el método que se encargará de invocar al método correspondiente de la capa de negocio según sea el ModoForm en que se encuentre el formulario
   Validar será el método que devuelva si los datos son válidos para poder registrar los cambios realizados.

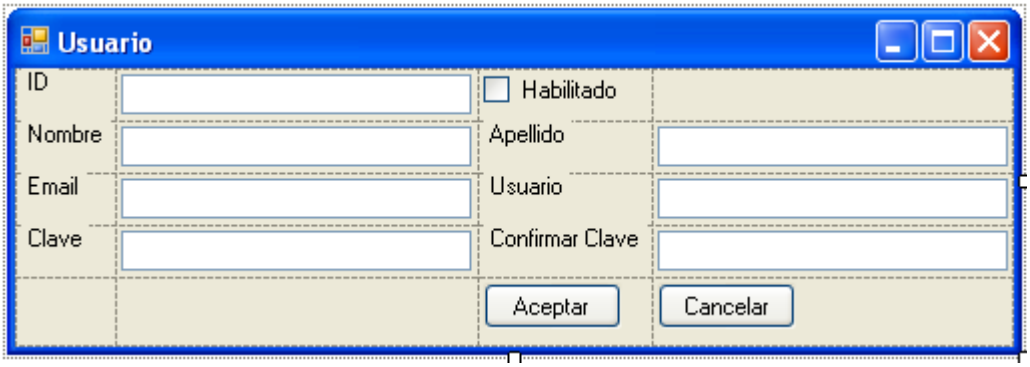
Notificar es el método que utilizaremos para unificar el mecanismo de avisos al usuario y en caso de tener que modificar la forma en que se realizan los avisos al usuario sólo se debe modificar este método, en lugar de tener que reemplazarlo en toda la aplicación.

4. Ahora crearemos otro formulario llamado UsuarioDesktop en UI.Desktop. Luego de crearlo modificamos el código para que herede de ApplicationForm y quede como sigue:
   public partial class UsuarioDesktop : ApplicationForm

5. Crear en UsuarioDesktop los métodos definidos ApplicationForm y dejarlos sin codificar por ahora.

6. Agregar la directivas using a Business.Logic y Business.Entities.

7. Ahora agregar un TableLayout y dentro del mismo los controles necesarios para que el formulario se vea así:



   Los nombres de los controles son: txtID, chkHabilitado, txtNombre, txtApellido, txtEmail, txtUsuario, txtClave, txtConfirmarClave, btnAceptar, btnCancelar.

8. En las propiedades de txtID y setear ReadOnly en true. El cuadro de texto se pondrá de color gris.

9. En el código del formulario definir una propiedad publica que se llame UsuarioActual de la clase Usuario de la capa de entidades.

10. Crear un constructor para UsuarioDesktop que reciba un parámetro con el Modo del formulario de la siguiente forma:
    public UsuarioDesktop(ModoForm modo):this()
    Internamete debe setear a ModoForm en el modo enviado, este constructor servirá para las altas.

11. Crear un tercer constructor para UsuarioDesktop que reciba un entero que represente el ID del usuario y el Modo en que estará el Formulario:
    public UsuarioDesktop(int ID, ModoForm modo):this()

12. En este nuevo constructor seteamos el modo que ha sido especificado en el parámetro e instanciamos un nuevo objeto de UsuarioLogic y

utilizamos el método GetOne para recuperar la entidad Usuario. Entonces la asignamos a la propiedad UsuarioActual e invocaremos al método MapearDeDatos.

13. En código del método MapearDeDatos copiar el contenido de cada una de las propiedades de UsuarioActual al control correspondiente. Por ej.:
```
this.txtID.Text = this.UsuarioActual.ID.ToString();
this.chkHabilitado.Checked = this.UsuarioActual.Habilitado;
this.txtNombre.Text = this.UsuarioActual.Nombre;
//completar el mapeo de los demás controles.
```

14. Dentro del mismo método setearemos el texto del botón Aceptar en función del Modo del formulario de esta forma:

Modo	**Texto del botón**
Alta o Modificación	Guardar
Baja	Eliminar
Consulta	Aceptar

15. En MapearADatos si el Modo del formulario es Alta crearemos un nuevo objeto de Usuario y lo asginaremos a UsuarioActual.
Luego, ya sea Alta o Modificación copiaremos el valor de cada uno de los controles a las propiedades respectivas de UsuarioActual. Tendremos que tomar precauciones para el caso del Alta no tendrá ningún ID.

16. Luego, en el MapearADatos Estableceremos la propiedad State de UsuarioActual de acuerdo al Modo del formulario.

17. En el método Validar debemos controlar los contenidos de los controles del no estén vacíos, que la clave coincida con la confirmación, tenga al menos 8 caracteres y el email sea válido. En caso de ser algo inválido debe retornar false e informar al usuario utilizando el método de Notificar que definimos anteriormente, y si es todo válido debe llamar retornar true.

18. En el método GuardarCambios debemos utilizar el método MapearADatos y luego crearemos una nueva instancia de la clase UsuarioLogic y llamaremos al método Save de esa instancia pasándole por parámetro UsuarioActual.

    Nota: Aquí se realiza la eliminación de la entidad mediante el método Save y enviándole la entidad como parámetro por simplicidad, pero podría haberse invocado al método Delete con el ID, esto puede ser particularmente útil cuando se esté realizando una eliminación sin tener toda la entidad usuario si no que sólo con us ID o en algún esquema web donde la performance puede mejorar al sólo tener que enviar el ID.

19. Hacemos doble clic en el botón Aceptar para agregar un manejador del evento clic y allí determinamos si los datos son válidos invocando al

método Validar y si lo son entonces ejecutamos el método GuardarCambios y luego cerramos el formulario con el método Close.

20. También creamos el manejador del evento clic para el botón salir y allí utilizamos el método

21. Volvemos al Windows Form Usuarios y en el ToolStripButton tbsNuevo hacemos doble clic para manejar el evento clic.

22. Allí creamos una variable de tipo UsuarioDesktop, la instanciamos invocando al constructor con un parámetro y pasanadole el modo alta. Luego mostramos el formulario con ShowDialog y a continuación refrescamos la grilla llamando al método Listar, para que al finalizar la ejecución de la nueva alta si se agregó un usuario este se vea en la grilla. El código será similar a este:

```
private void tsbNuevo_Click(object sender, EventArgs e)
{
    UsuarioDesktop formUsuario = new UsuarioDesktop(ApplicationForm.ModoForm.Alta);
    formUsuario.ShowDialog();
    this.Listar();
}
```

23. Compilar y probar que funcione el alta.

24. De manera similar llamar al Formulario crear los manejadores de eventos para el Editar y el Eliminar. Utilizando el constructor de UsuarioDesktop que requiere enviar el ID y el Modo
    Para obtener el ID podemos hacerlo de la siguiente forma

```
int ID = ((Business.Entities.Usuario)this.dgvUsuarios.SelectedRows[0].DataBoundItem).ID;
```

Para poder utilizar dicha línea de código debemos:

- Asegurarnos que haya una fila seleccionada (controlando que this.dgvUsuarios.SelectedRows tenga elementos dentro)
- Permitir que se selecciones una única fila (Propiedad MultiSelect de la grilla en false)
- Que al hacer clic en una celda se seleccione una fila entera (Propiedad SelectionMode de la grilla en FullRowSelect)

Historia de Versiones

Fecha	Versión	Autor	Descripción
29/05/2011	0.1	AM	Versión beta. Pendiente: Validar, Guardar cambios, vincular formularios y refrescar grilla.
13/06/2011	1	AM	Versión inicial