Unidad 3 – Capítulo 1 - Laboratorio 1

1. Creación Formulario Login

Objetivos
Construir un login para una aplicación WinForm.

Sugerencia
Puede utilizar directamente el proyecto de la capa de presentación del TP2 para realizar este laboratorio.

Duración Aproximada

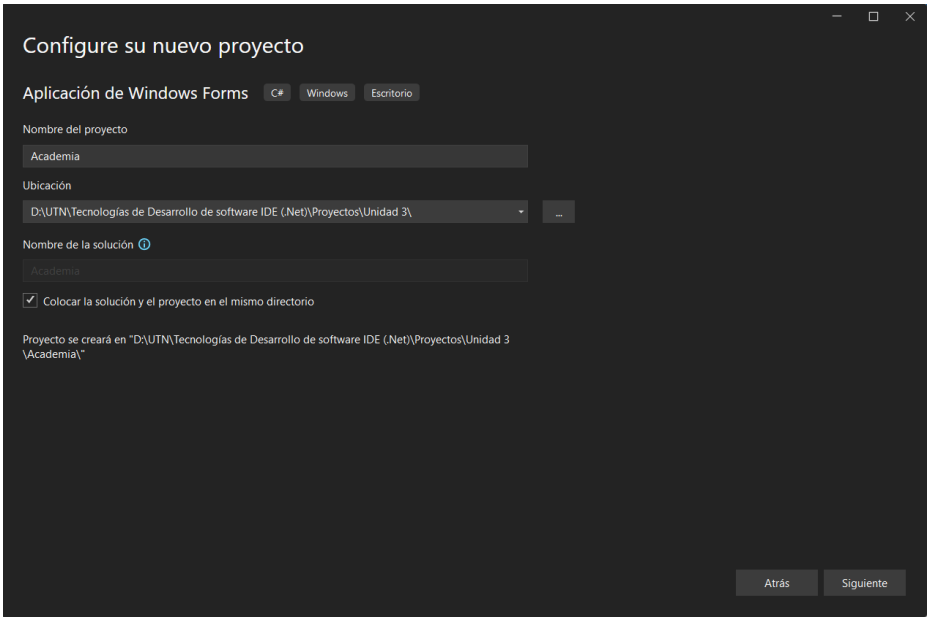
15 minutos

Pasos

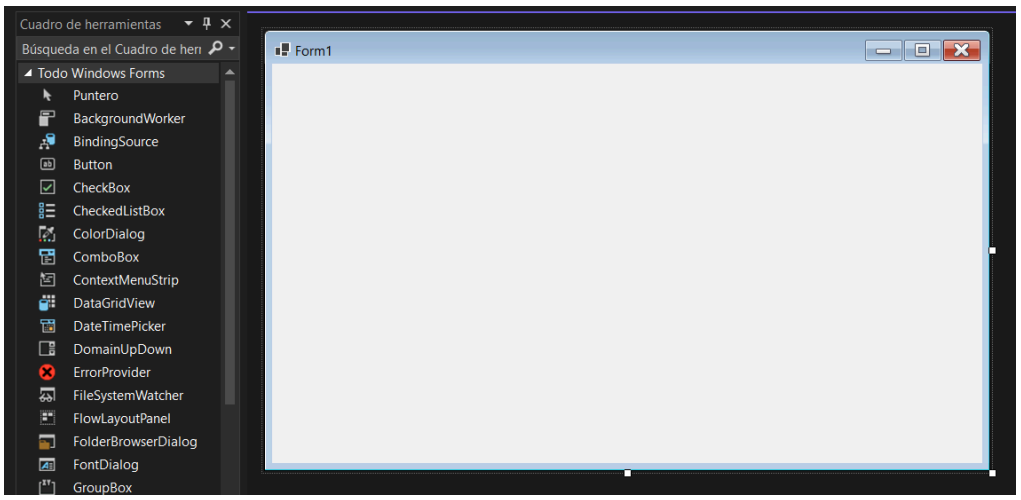
1) Abra el Visual Studio 2022 y elija crear un nuevo proyecto.

2) Elija un proyecto de la categoría C# -> Aplicación de Windows Forms.

   A) Completar los siguientes datos:
   - Nombre del Proyecto (Name): Academia
   - Ubicación de la solución (Location): C:\Proyectos\Unidades\Unidad03



3) Presione Siguiente. Aparecerá un formulario en blanco como el siguiente.



A la izquierda se distingue una solapa llamada Cuadro de herramientas. En esta aparecen todos los controles que podemos utilizar en el Formulario.
En este laboratorio utilizaremos los controles de tipo:

- Label (Etiquetas): Sirven para mostrar un texto, el cual, el usuario no puede editar.
- TextBox (Cajas de texto o Cuadros de Texto): Sirven para contener texto con el cual el usuario puede interactuar. El grado de interacción varía según las propiedades Enable y ReadOnly del TextBox
- Button (Botón): El usuario puede interactuar con este control haciendo clic sobre el mismo y así disparar la ejecución de una acción definida por el usuario.
- LinkLabel (Etiqueta de vínculo): Comúnmente se utiliza para contener un hipervínculo. Al posicionar el puntero del mouse sobre ellos la forma del puntero cambia a la de una mano con el dedo índice extendido. Puede utilizarse para disparar acciones.

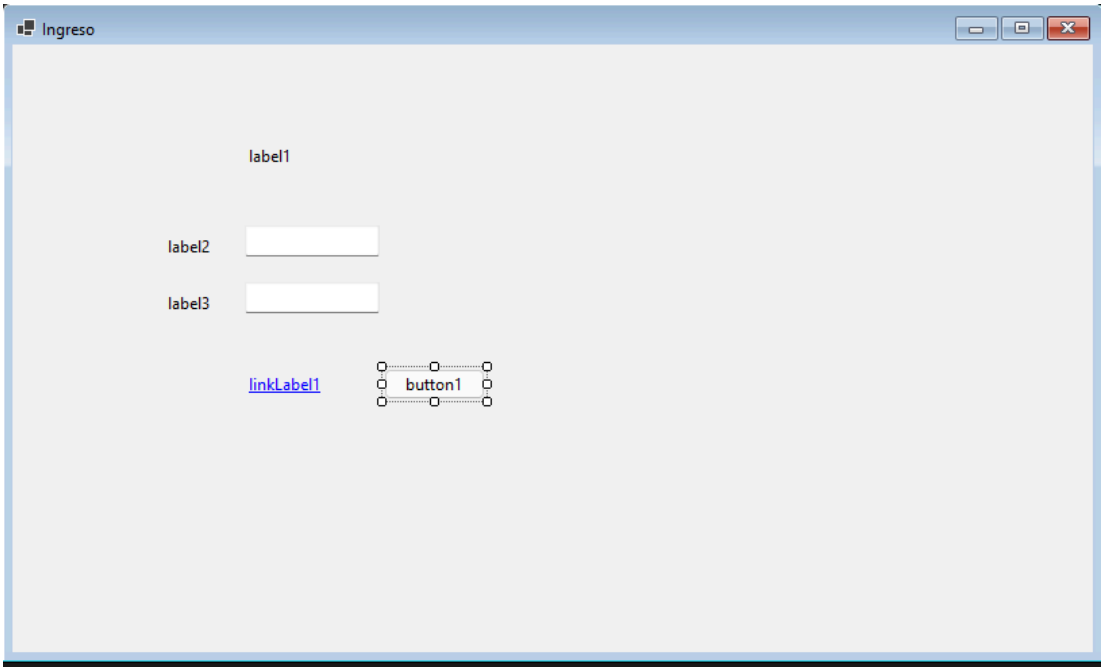
A la derecha aparece la ventana de propiedades en la cual podemos modificar las propiedades del objeto que se encuentra seleccionado.

4) Hacemos clic sobre el formulario entonces editamos las siguientes propiedades:

   - (Name) (Nombre del objeto): Lo seteamos a formLogin
   - Text (Título del formulario): escribimos Ingreso

   Luego en el explorador de soluciones hacemos clic con el botón derecho sobre Form1 y elegimos cambiar el nombre por formLogin

5) Luego del Cuadro de herramientas arrastramos 3 Labels, 2 TextBox, 1 Button y un LinkLabel y los ubicamos de la siguiente forma:

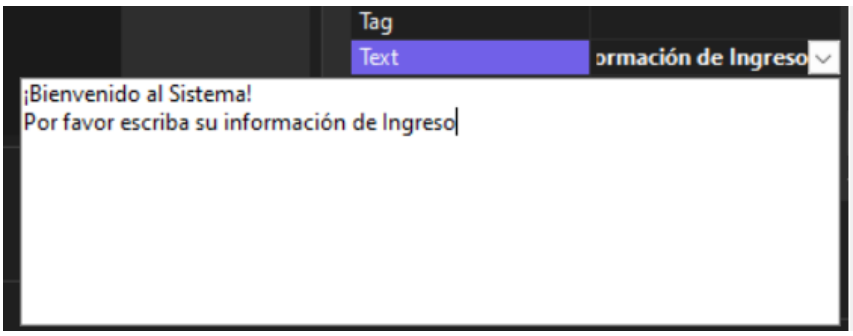


6) Ahora editamos cada control y sus propiedades de la siguiente forma

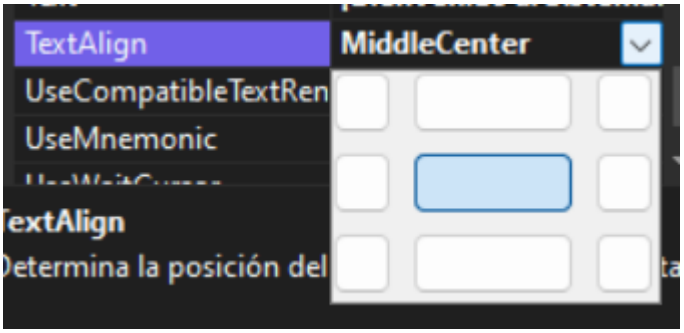
Control	Propiedad	Valor Modificado
label1	Text(*)	¡Bienvenido al Sistema! Por favor digite su información de Ingreso
	TextAlign(**)	MiddleCenter
label2	Text	Nombre de Usuario:
label3	Text	Contraseña:
linkLabel1	(Name)	lnkOlvidaPass
	Text	Olvidé mi contraseña
textBox1	(Name)	txtUsuario
textBox2	(Name)	txtPass
button1	(Name)	btnIngresar
	Text	Ingresar

ATENCIÓN: Al cambiar la propiedad (Name) el objeto dejará de llamarse como se indica en la columna de la izquierda y pasará a llamarse con el texto que nosotros ingresamos.

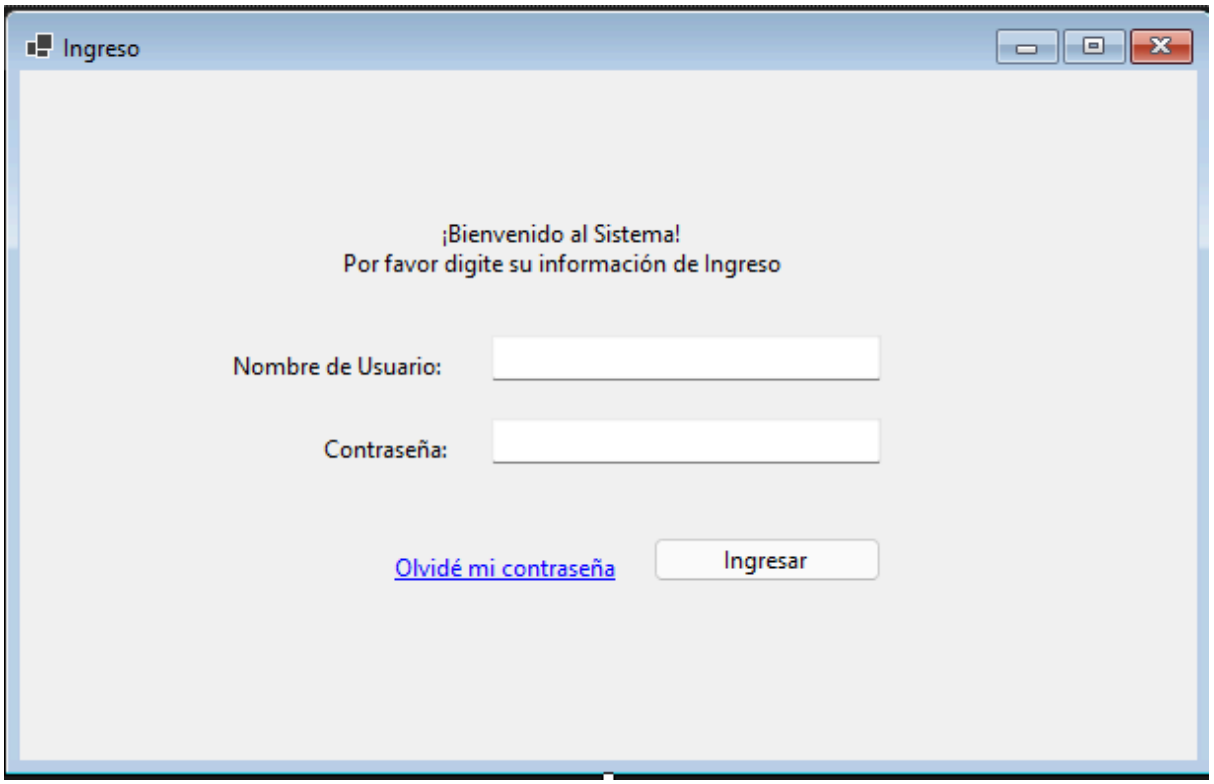
(*) Para ingresar varias líneas haga clic en el cuadro de la propiedad y luego en el botón desplegar.



(**) Seleccionando como se muestra en la imagen



7) Arrastramos el borde del formulario para agrandar el formulario y reorganizamos los controles para que el formulario formLogin luzca así:



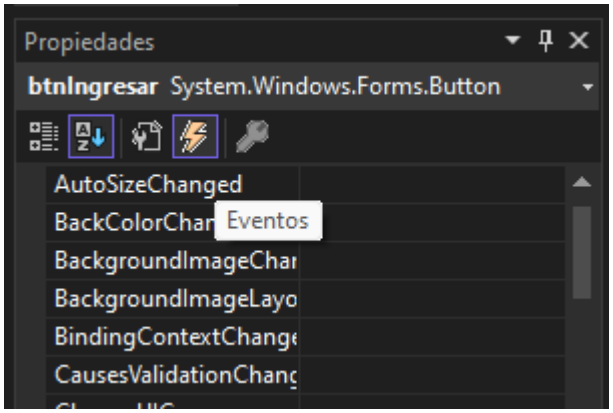
2. Eventos

Objetivos
Definir eventos del formulario Login

Duración Aproximada
10 minutos

Pasos

1) Haga clic sobre el botón btnIngresar luego en la ventana de propiedades haga clic en el ícono del relámpago para editar los eventos.



Entonces la ventana de propiedades cambiará y en lugar de mostrarnos las propiedades del objeto nos mostrará todos los eventos que podemos detectar en el control y así disparar la ejecución de un método.
En este caso nos interesa el evento llamado click

2) Hacemos doble clic en el cuadrado de la derecha (del evento click). Automáticamente se creará un EventHandler (Manejador de evento, del evento click y un nuevo método que se ejecutará cada vez que este evento suceda el nombre del manejador será de la forma nombreControl_nombreEvento) seremos redirigidos a al código del formulario donde tenemos la clase formLogin, su constructor por defecto y el recién agregado método que se ejecutará durante el evento Click.

```
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;

namespace Academia
{
	public partial class formLogin : Form
	{
		public formLogin()
		{
			InitializeComponent();
		}

		private void btnIngresar_Click(object sender, EventArgs e)
		{
		}
	}
}
```

3) Modificamos el handler del evento click de la siguiente forma:

```
private void btnIngresar_Click(object sender, EventArgs e)
{
    //la propiedad Text de los TextBox contiene el texto escrito en ellos
    if (this.txtUsuario.Text == "Admin" && this.txtPass.Text == "admin")
    {
          MessageBox.Show("Usted ha ingresado al sistema correctamente."
          , "Login", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("Usuario y/o contraseña incorrectos", "Login"
          , MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

La clase MessageBox nos permite mostrar un formulario modal simple para informarle al usuario y en caso de ser necesario una respuesta de acuerdo al botón que presione (OK, Cancel, Yes, No, Abort, Retry, Ignore y None).

4) Hacemos entonces lo mismo con el lnkOlvidaPass pero para el evento LinkClicked y agregamos el siguiente código:

```
private void lnkOlvidaPass_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
      MessageBox.Show("Es Ud. un usuario muy descuidado, haga memoria",
            "Olvidé mi contraseña",
      MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

5) Entonces ejecutamos la aplicación y probamos que todo funcione correctamente.

6) Como habrán notado la contraseña es visible, para proteger la contraseña debemos volver a hacer clic en el txtPass y en la ventana de propiedades hacer clic en el ícono de la izquierda del relámpago llamado Propiedades para así volver a visualizar las propiedades del control de texto. Entonces en la propiedad PasswordChar escribimos: *

7) Al hacer esto no sólo ocultamos la contraseña sino que también aparece un cartel de advertencia cuando tenemos presionado el CapsLock (BloqMayus).

8) Ejecutamos nuevamente la aplicación y probamos.

IMPORTANTE:
El código utilizado en el handler del evento click del btnIngresar debe ser reemplazado por un método que solicite a la capa de negocio recupere el usuario con nombre igual al ingresado en el txtUsuario y si existe invocar a un método que valide si su contraseña coincide con la ingresada en txtPass.

9) Ejercicios adicionales sugeridos (no necesariamente tiene que hacerlos ahora)

A. Cuando complete la persistencia y acceso a la fuente de datos, reemplace el código del punto #3 para lograr una autenticación sobre los datos existentes persistidos.
B. El punto #4 puede reemplazar por el uso de un servicio que envíe un mail con algún mecanismo que permita recuperar/resetear la clave.
Nota: puede tomar de referencia lo planteado en por el sitio web oficial de Microsoft https://learn.microsoft.com/es-es/dotnet/api/system.net.mail.smtpclient?view=net-7.0

3. Formulario Modal

Objetivos
Incluir el formLogin en una aplicación con un formulario MDI y convertirlo en un formulario modal

Sugerencia
Puede utilizar directamente el proyecto de la capa de presentación del TPI para realizar este laboratorio.

Duración Aproximada
20 minutos

Pasos

1) En el proyecto Academia agregar un nuevo formulario, al cual llamaremos formMain, el cual será el formulario principal de la aplicación.

Nota: En un próximo laboratorio trabajaremos sobre este formulario en detalle. Ahora sólo configuraremos las principales propiedades.

2) En el formMain editamos las propiedades:
   - Text: Academia
   - WindowState: Maximized
   - IsMdiContainer: True

Al modificar esta última propiedad notaremos que el color de fondo cambiará.

3) En el formMain agregamos un MenuStrip. En el mismo cambiamos la propiedad (Name) por mnsPrincipal.

4) En la zona donde dice Escriba aquí escribimos Archivo y en la que aparece debajo escribimos Salir.

Esto crea 2 objetos archivoToolStripMenuItem y salirToolStripMenuItem, cuyas propiedades cambiaremos por mnuArchivo y mnuSalir respectivamente (haciendo click derecho sobre el nombre Cambiar nombre)

5) En el mnuSalir agregamos un handler para el evento Click como lo hicimos con el btnIngresar del formLogin.

6) En el método que se crea escribimos:

```
private void mnuSalir_Click(object sender, EventArgs e)
{
        this.Dispose();
}
```

7) Entonces en el formMain agregaremos un handler para el evento Shown. Este evento se ejecuta la primera vez que se muestra el formulario.

```
private void formMain_Shown(object sender, EventArgs e)
{
        formLogin appLogin = new formLogin();
        appLogin.Show();
}
```

8) Luego nos dirigimos a la clase Program y en el método Main cambiamos el formulario que se ejecuta al inicio del formLogin a formMain quedando así:

```
using System;
```

```
using System.Collections.Generic;
using System.Windows.Forms;

namespace Academia
{
	static class Program
	{
	/// <summary>
	/// The main entry point for the application.
	/// </summary>
	[STAThread]
	 static void Main()
	{
		Application.EnableVisualStyles();
		Application.SetCompatibleTextRenderingDefault(false);
		//Application.Run(new formLogin());
		Application.Run(new formMain());
	}
    }
}
```

9) Ejecutamos la aplicación y probamos.

Luego de ejecutar podemos notar algunos inconvenientes:

- A pesar de que el formLogin aparece delante del formMain podemos hacer clic en el formMain e interactuar con el mismo haciendo clic en el menú salir. Si además del menú salir hubiese otro menú podríamos ejecutarlo sin necesidad de tener un usuario y una contraseña.
- El formLogin puede modificar su tamaño, ser minimizado o maximizado y no aparece centrado.
- Al presionar la tecla Enter en el formLogin no sucede nada. La única forma de tratar de ingresar es haciendo clic en el botón btnIngresar
- No hay conexión ni forma de saber si el usuario se registro correctamente o no para permitir el acceso o no

Ahora procederemos a explicar porqué sucede eso y resolveremos estos cuatro inconvenientes.

10) Primero: evitar que se pueda interactuar con el frmMain hasta tanto se cierre el formLogin.

El problema sucede porque utilizamos el método Show para mostrar el formLogin. Este método sólo muestra el formulario pero no crea ningún tipo de restricción en cuanto a que el formulario deje de ser el formulario activo. Por ello lo cambiaremos por el método ShowDialog.
Este método permite mostrar la ventana formLogin como una ventana modal. Esto significa que no se podrá interactuar con el resto de la aplicación salvo que se haya cerrado la ventana que se mostró con ShowDialog. Entonces el método formMain_Shown queda así:

```
private void formMain_Shown(object sender, EventArgs e)
{
	formLogin appLogin = new formLogin();
	appLogin.ShowDialog();
}
```

11) Ejecutamos y probamos. Ahora la ventana actúa en forma modal pero no se cierra automáticamente. Para ello modificamos el método btnIngresar_Click del formLogin así:

```
private void btnIngresar_Click(object sender, EventArgs e)
{

	//la propiedad Text de los TextBox contiene el texto escrito en ellos
```

```
        if (this.txtUsuario.Text == "Admin" && this.txtPass.Text == "admin")
        {
            this.DialogResult = DialogResult.OK;
        }
        else
        {
            MessageBox.Show("Usuario y/o contraseña incorrectos", "Login",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
```

12) Ahora cuando el usuario se registra correctamente se setea la propiedad DialogResult del formulario. Cuando se setea dicha propiedad en un formulario que se mostró con el método DialogResult el formulario se cierra automáticamente. Adicionalmente el método ShowDialog con el cual se mostró el formLogin devuelve el valor que se le asigna a la propiedad DialogResult del formulario. Por defecto al cerrar el formulario desde la cruz la respuesta es DialogResult.Cancel.

13) El segundo inconveniente es evitar que el formLogin pueda ser redimensionado, maximizado o minimizado y que aparezca centrado

    Para ello editamos las siguientes propiedades del formLogin:
    - MaximizeBox (Botón de maximizar): False
    - MinimizeBox (Botón de minimizar): False
    - FormBorderStyle (Estilo del borde del formulario): FixedSingle
    - StartPosition (Posición inicial): CenterParent

14) Tercer objetivo lograr que el Enter ejecute la acción por defecto deseada en este caso lo mismo que clic en el btnIngresar.

    Para ello en las propiedades del frmLogin en AcceptButton elegimos el botón btnIngresar. Ahora el botón btnIngresar se convierte en el botón por defecto del frmLogin y al hacer Enter es como hacer clic en el botón.

15) El cuarto y último inconveniente es que en caso que se cierre el frmLogin sin registrarse correctamente se cierre la aplicación.

    Para ello modificamos el método fromMain_Shown de la siguiente forma:

```
private void formMain_Shown(object sender, EventArgs e)
{
    formLogin appLogin = new formLogin();
    if (appLogin.ShowDialog() != DialogResult.OK)
    {
        this.Dispose();
    }
}
```

    En el método btnIngresar_Click del formLogin habíamos hecho que al registrarse correctamente el resultado del formulario (DialogResult) fuera OK entonces cualquier otro resultado significa que el usuario no se ha registrado correctamente.

16) Ejecutamos y probamos.

Código Fuente Resultante

En esta sección está el código resultado generado en los puntos previos.

Clases

Clase Program

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Academia
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            //Application.Run(new formLogin());
            Application.Run(new formMain());
        }
    }
}
```

Clase formLogin

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

namespace Academia
{
    public partial class formLogin : Form
    {
        public formLogin()
        {
            InitializeComponent();
        }

        private void btnIngresar_Click(object sender, EventArgs e)
        {
            //la propiedad Text de los TextBox contiene el texto escrito en ellos
            if (this.txtUsuario.Text == "Admin" && this.txtPass.Text == "admin")
            {
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Usuario y/o contraseña incorrectos", "Login"
                    , MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void lnkOlvidaPass_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            MessageBox.Show("Es Ud. un usuario muy descuidado, haga memoria", "Olvidé mi contraseña",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
}
```

Clase formMain

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

namespace Academia
{
    public partial class formMain : Form
    {
        public formMain()
        {
            InitializeComponent();
        }

        private void mnuSalir_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void formMain_Shown(object sender, EventArgs e)
        {
            formLogin appLogin = new formLogin();
            if (appLogin.ShowDialog() != DialogResult.OK)
            {
                this.Dispose();
            }
        }
    }
}
```