

GUIA PRÁCTICA SEMANA 03

Competencia:

El desarrollo de esta guía práctica te permitirá conocer la creación de objetos DataTable en memoria para hacer una simulación de un registro de datos en una agenda electrónica, incluyendo el manejo de las fotos como archivos asociados a las personas registradas. Así mismo, veremos una versión inicial de un formulario de Login para ingresar las credenciales (usuario y contraseña)

Descripción:

La guía se basa en el proyecto **ProyWinC_Sem03**. La guía se divide en 2 secciones:

- Un laboratorio guiado paso a paso con la solución del registro de personas en una agenda electrónica.
- Un laboratorio guiado paso a paso para el acceso mediante del formulario **frmLogin** al formulario de **MDIPrincipal**.

PARTE 1: REGISTRO DE DATOS PERSONALES

El escenario es crear una aplicación que registre solo en memoria datos de personas y luego se genere un archivo XML en una carpeta predeterminada, para posteriormente visualizar la información. Se incluye el registro de la foto de cada persona como parte del proceso.

Paso 1:

Recupera el proyecto **ProyWinC_Sem03**, ubique el formulario **frmRegistroPersonal**:

The screenshot shows a Windows form titled "Registro Personal". It contains several input fields and buttons, each marked with a red circle and a number:

- 1: DNI input field
- 2: Nombres input field
- 3: Apellidos input field
- 4: F.Registro date picker (showing 12/09/2021)
- 5: Sueldo input field
- 6: "Cargar Foto" button
- 7: Large empty rectangular area for the photo
- 8: Empty horizontal input field at the bottom
- 9: "Agregar registro" button
- 10: "Grabar y Cerrar" button

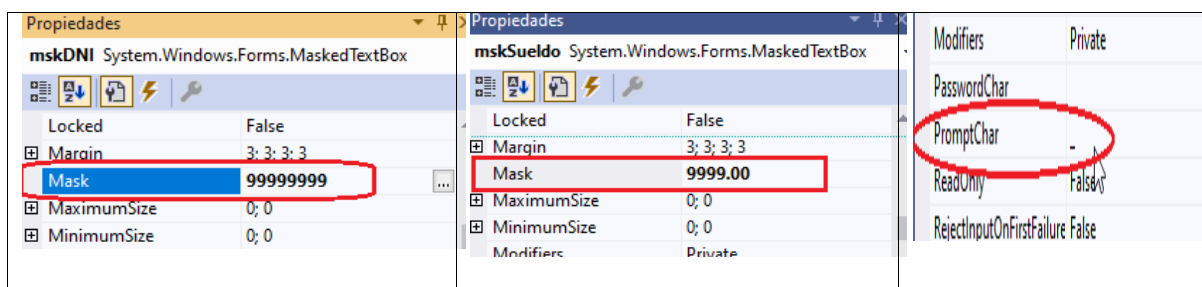
Los controles que se incluyen son los siguientes:

1	MaskedTextBox - mskDNI
2	TextBox - txtNombres
3	TextBox - txtApellidos
4	DateTimePicker - dtpFechaReg
5	MaskedTextBox - mskSueldo
6	Button - btnCargar
7	PictureBox - pcbFoto
8	Label - lblRuta
9	Button - btnAgregar
10	Button - btnGrabar

Vale la pena mencionar el empleo de los controles **MaskedTextBox** y **DateTimePicker** en este formulario

Como se puede apreciar los controles 1 y 5 son de tipo **MaskedTextBox**. Este tipo de control nos permite emplear mascararas para definir el formato del dato que se va a ingresar en el control. En el caso del control **mskDNI** vea que la propiedad **Mask** tiene el valor **99999999**. El caracter **9** representa solo dígitos, por lo que sin necesidad de hacer ninguna codificación adicional el control no permitirá tipear ningún carácter que no sea dígito. Así mismo el control **mskSueldo** en la propiedad **Mask** tiene el valor **9999.00** para establecer un formato de ingreso de sueldos que incluya 2 decimales. En ambos casos se define como propiedad **PromptChar** el carácter “_” que es lo que se visualiza como indicador al momento del ingreso de los datos para ambos controles.

NOTA: Si deseas revisar las opciones de la propiedad Mask del control MaskedTextBox, revisa el documento Guia_MaskedTextBox entregado en el material de la semana 3.



Para el caso de la fecha de registro (4) se emplea en control de tipo **DateTimePicker** llamado **dtpFechaReg**. Este control es el indicado para el ingreso de todo lo referente a fechas, manejando de manera fácil y sin necesidad de hacer validaciones este tipo de datos. En el formulario se aprecia que en la propiedad **Format** se la asigno el valor **Short** para que se establezca el formato típico de **dd-MM-yyyy**.

Paso 2:

Como la intencion es registrar los datos de personas y almacenarlos en primera instancia en una tabla en memoria, es necesario crear dicha tabla en memoria. En la zona de declaraciones vemos que se han instanciado los siguientes objetos :

```
public partial class frmRegistroPersonal : Form
{
    // Creamos las instancias de DataSet y DataTable
    DataSet dts = new DataSet();
    DataTable tabla = new DataTable();

    // Variables para las columnas y las filas
    DataColumn column;
    DataRow row;
```

Como adelanto a lo que se vera en las semanas proximas, vemos que se define una instancia llamada dts , de la clase **DataSet**. Un DataSet es un objeto que se contruye en memoria y sirve como un repositorio de tablas. Tambien vemos la creacion del objeto tabla de la clase **DataTable**, que sera la tabla en la que almacenaremos los datos. Dicha tabla se construira en base a las instancias column de la clase **DataColumn** y row de la clase **DataRow**. Todo este escenario es lo que se conoce como “**Entorno Desconectado**”, ya que no se requiere de una conexión a base de datos para establecer una tabla y alimentarla con datos, sino que toda la gestion se hace en memoria. Logicamente veremos como toda esa data en memoria se descargar en un archivo XML. Para poder trabajar con todas estas clases es necesario referenciar al ensamblado **System.Data** como se indica en la plantilla.

En el evento Load del formulario, se invoca al procedimiento CrearTabla (donde se crearara la estructura de la tabla en memoria) el cual debemos codificar de acuerdo a lo siguiente:

```
private void CrearTabla()
{
    // Creamos el DataTable en memoria
    // 1. Creamos las columnas
    // DNI...
    column = new DataColumn();
    column.DataType = Type.GetType("System.String");
    column.ColumnName = "DNI";
    tabla.Columns.Add(column);

    // Nombres...
    column = new DataColumn();
    column.DataType = Type.GetType("System.String");
    column.ColumnName = "Nombres";
    tabla.Columns.Add(column);

    //Apellidos...
    column = new DataColumn();
    column.DataType = Type.GetType("System.String");
    column.ColumnName = "Apellidos";
    tabla.Columns.Add(column);

    // Fecha de registro
    column = new DataColumn();
    column.DataType = Type.GetType("System.DateTime");
    column.ColumnName = "FechaReg";
    tabla.Columns.Add(column);

    // Sueldo
    column = new DataColumn();
    column.DataType = Type.GetType("System.Single");
    column.ColumnName = "Sueldo";
    tabla.Columns.Add(column);

    // 2. Aplicamos la PK al DataTable...
    tabla.PrimaryKey = new DataColumn[] { tabla.Columns ["DNI"] };

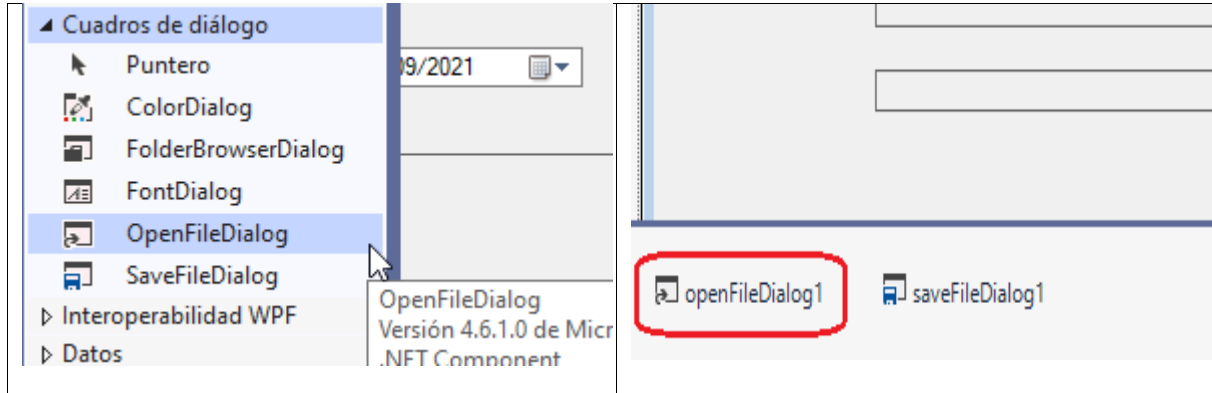
    // 3. Colocamos la tabla dentro de la coleccion de tablas del Dataset
    dts.Tables.Add(tabla);
}
```

Como apreciamos en la imagen superior, empleamos la instancia `column` para ir creando las columnas DNI, Nombres, Apellidos, FechaReg y Sueldo. Cada columna al momento de crearse especifica su tipo de dato por medio de la clase `Type` y el metodo `GetType`. Por ejemplo `Type.GetType("System.DateTime")` para la fecha de registro o `Type.GetType("System.String")` para el caso del DNI, nombres y apellidos o `Type.GetType("System.Single")` para el caso del sueldo. Siempre que defina el tipo de dato debe respetar escribirlo respetando las mayusculas y minusculas, es decir por ejemplo "System.String" y no "system.string".

Se aprecia tambien que se puede definir una llave primaria a la tabla en memoria (en este caso el DNI). La llave primaria en un DataTable se define en la propiedad `PrimaryKey` y se establece en un arreglo de tipo `DataColumn` , colocando el nombre del campo que sera llave principal. Por ultimo, creada la tabla y su PK , esta se agrega a la colección de tablas del DataSet con el conocido metodo `Add`, sobre la colección de tablas del DataSet `dts` definida por `Tables`.

Paso 3:

Ahora procederemos a codificar el evento **Click** del botón **btnCargar**. En este evento trabajaremos con el control **openFileDialog1**. Este control está alojado en la parte inferior del formulario y es un cuadro de dialogo que invoca al conocido cuadro de dialogo Abrir, que en este caso permitirá seleccionar la ubicación del archivo donde está la foto de la persona y asignarla al **pictureBox** del formulario llamado **pcbFoto** en tiempo de ejecución. El control **OpenFileDialog** está en el cuadro de herramientas, en el grupo de **Cuadros de Dialogo**



Ingreseemos el siguiente código en el evento **Click** del botón **btnCargar**:

```
private void btnCargar_Click(object sender, EventArgs e)
{
    try
    {
        // Mostraremos el cuadro de dialogo openFileDialog1 para poder cargar la foto

        // Establecemos propiedades del openFileDialog
        openFileDialog1.Multiselect = false;
        openFileDialog1.FileName = "";
        openFileDialog1.Filter = "Fotos (Solo jpg) | *.jpg"; // El caracter | se obtiene con Alt + 124
        openFileDialog1.ShowDialog();

        // Elegida la foto se muestra la ruta de la foto en el label y la foto en pcbFoto
        lblRuta.Text = openFileDialog1.FileName;
        pcbFoto.Image = Image.FromFile(openFileDialog1.FileName);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error");
    }
}
```

El código mostrado establece inicialmente las propiedades **Multiselect** en false, para que solo se pueda elegir un archivo, **FileName** en "" para que no se establezca ningún nombre de archivo por defecto y la más importante propiedad en este caso sería **Filter**, donde especificamos que tipo de archivos serán seleccionables desde el openFileDialog1. El formato de la propiedad **Filter** es una cadena compuesta por: <<Texto de Referencia>> | <<Filtro>>. Para nuestro ejercicio hacemos referencia solo archivos JPG. Definidas las propiedades lanzamos el **openFileDialog1** con el

método **ShowDialog()** (recuerde que todos las ventanas de cuadros de dialogo se muestran de forma modal).

Tras seleccionar el archivo y pulsar Abrir en el cuadro de dialogo en tiempo de ejecución se mostrará la ruta completa del archivo de la foto seleccionada en el **label lblRuta** y la imagen se cargará en el **pictureBox pcbFoto**.

Paso 4:

A continuación, se codificará el evento **Click** del botón **btnAgregar**. El propósito es que al hacer click en este botón se agregue un registro al **DataTable Tabla**. Se debe validar que la máscara del control **mskDNI** se llene, que se haya seleccionado una foto y que el sueldo asignado este en el rango de 1000 a 28000. En caso que se rompa una de estas reglas de negocio se provocara una excepción con el respectivo mensaje. Veamos el código que debe ingresar:

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    try
    {
        // Validamos segun reglas de negocio

        // Se obliga a que se llene la mascara del DNI
        if (mskDNI.MaskFull == false)
        {
            throw new Exception("El DNI debe tener 8 caracteres");
        }

        // Foto obligatoria
        if (lblRuta.Text == String.Empty)
        {
            throw new Exception("La foto es obligatoria");
        }

        // Sueldo entre 1000 y 2800
        Single sngSueldo = Convert.ToSingle(mskSueldo.Text);

        if (sngSueldo < 1000 | sngSueldo > 2800)
        {
            throw new Exception("Sueldo debe estar entre 1000 y 2800 soles.");
        }

        // Si todo esta OK , se adiciona la el registro de la nueva persona al DataTable ...

        // Se instancia una nueva fila del DataTable...
        row = tabla.NewRow();
    }
}
```

```
// Llenamos la nueva fila con todo los datos de la persona
row["DNI"] = mskDNI.Text;
row["Nombres"] = txtNombres.Text.Trim();
row["Apellidos"] = txtApellidos.Text.Trim();
row["FechaReg"] = dtpFechaReg.Value.Date;
row["Sueldo"] = sngSueldo ;

// Agregamos la nueva fila ya llenada a la coleccion de filas de la tabla...
tabla.Rows.Add(row);

// Se copia la foto a la carpeta destino
// NOTA: Debe crear la carpeta previamente. Si Ud. desea puede definir otra ruta
// para lo cual haga la modificacion respectiva.

File.Copy(lblRuta.Text, @"C:\Aleon\RepositorioFotos\" + mskDNI.Text + ".jpg",true);

// Se envia el mensaje de conformidad y se invoca al metodo Limpiar
// dejando todo listo para un siguiente ingreso si es necesario

MessageBox.Show("Registro almacenado EN MEMORIA con exito");
Limpiar();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Error");
}
}
```

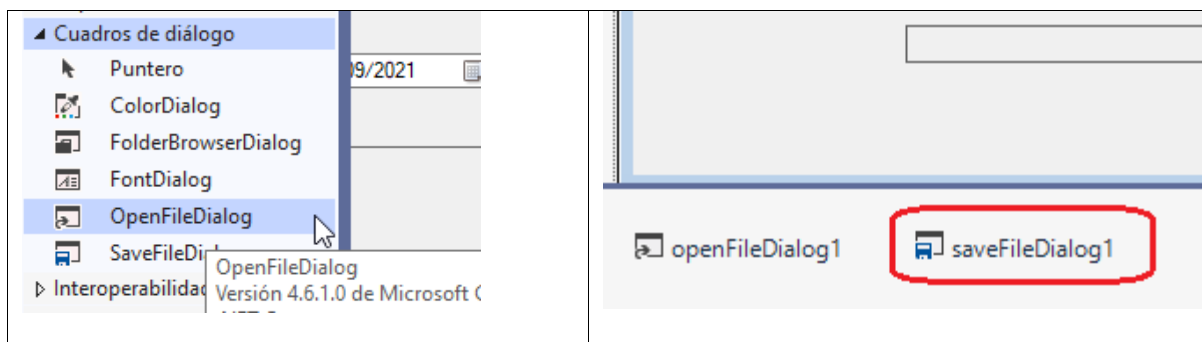
En el código se puede ver que, tras las validaciones de las reglas de negocio, de estar todo bien, se crea una instancia de nueva fila de la tabla **Tabla** (row=Tabla.NewRow) con el objeto **row**. Con la nueva instancia de fila creada, se cargan a cada una de las columnas los datos que provienen desde los controles del formulario y, llenada la fila, esta se agrega a la colección de filas del **DataTable Tabla** (Tabla.Rows.Add(row)).

Para el tema de la foto, lo que se hace es copiar el archivo definido en el **label lblRuta** (que se seleccionó desde el **openFileDialog1**) hacia la carpeta del repositorio de fotos, estableciendo como nombre de la copia el propio DNI de la persona. El parámetro true al final de la expresión indica que si hubiera ya un archivo con ese nombre se reemplace. Para ello se aprecia que se emplea la clase **File** con el método **Copy**. Recuerde que la carpeta de destino debe crearse para poder ejecutar correctamente el ejercicio. Si deseas puedes establecer otra ruta, modificando el código lógicamente.

Por último, tras agregarse el registro se envía un mensaje de confirmación y se invoca al procedimiento **Limpiar** (que ya está codificado en la plantilla) para que todos controles vuelvan a su estado original para un nuevo ingreso.

Paso 5:

Para generar el archivo XML con todos los datos de los registros en memoria, haremos clic en el botón **btnGrabar**. Para especificar la ruta y nombre de archivo donde se guardará el archivo XML emplearemos el control de tipo cuadro de dialogo **SaveFileDialog**, el cual nos mostrará la conocida ventana **Guardar**. El control **saveFileDialog1** está ubicado se ha incluido en el formulario y se aloja en la parte baja al costado del **openFileDialog1**.



Ingresemos el siguiente código en el evento **Click** del botón **btnGrabar**:

```
private void btnGrabar_Click(object sender, EventArgs e)
{
    try
    {
        // Validamos que la tabla tenga filas
        if (tabla.Rows.Count == 0)
        {
            throw new Exception("La tabla no tiene registros");
        }

        // Configuramos el savefiledialog para que solo admita guardar archivos XML
        saveFileDialog1.Filter = "XML (Solo xml)| *.xml"; // Alt + 124 |
        saveFileDialog1.ShowDialog();

        // Se obtiene la ruta seleccionada y nombre del archivo y
        // grabamos la tabla en formato XML en dicha ruta...

        String strRuta = saveFileDialog1.FileName;
        dts.WriteXml(strRuta);

        // Mandamos mensaje de conformidad y cerramos el formulario...

        MessageBox.Show("Archivo grabado con éxito!!!", "Mensaje");
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error");
    }
}
```

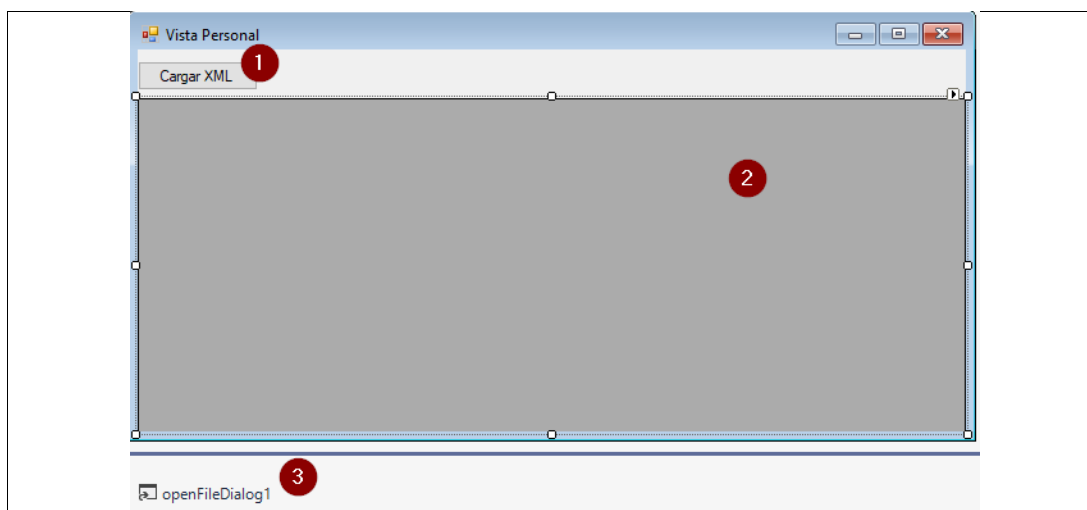
En el código mostrado se establece en primer lugar una validación para verificar que existen registros en el **DataTable Tabla**. Si no lo hubiera se provoca una excepción con el respectivo mensaje.

Si el **DataTable Tabla** tiene registros se establece la propiedad **Filter** del control **saveFileDialog1** (como se hizo con el **openFileDialog1**), pero en este caso para definir solo archivos XML, para luego mostrar el cuadro de dialogo del **saveFileDialog1**. La ruta y nombre de archivo donde el usuario decida grabar el archivo XML se asigna a la variable **strRuta**, y dicha variable sirve de parámetro para el método **WriteXML** del **DataSet dts**, generándose así el archivo XML con todos

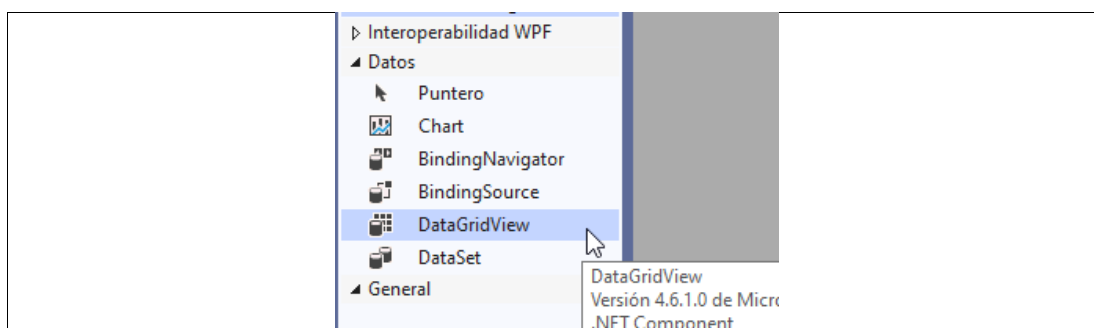
los datos de las personas registradas. Finalmente se envía el mensaje respectivo y se cierra el formulario, terminando el proceso.

PARTE 2: RECUPERANDO EL ARCHIVO XML CON LOS DATOS PERSONALES Y MOSTRARLO EN UN DATAGRIDVIEW

Programaremos ahora en el formulario **frmVistaPersonal** para que podamos mostrar los datos de las personas registradas en el archivo XML de la parte 1. El diseño del formulario es el siguiente:



Este formulario incluye el botón **btnCargar** (1), el control **DataGridView dtgPersonal** (2) y el control **OpenFileDialog openFileDialog1**. El control **DataGridView** se ubica en el cuadro de herramientas dentro de la categoría Datos.



Precisamente el control **DataGridView dtgPersonal** nos permitirá visualizar en filas y columnas los registros provenientes del archivo XML. Se le han asignado una serie de propiedades que se van explicar más adelante. La idea es que al hacer clic en el botón **btnCargar**, se muestre el cuadro de dialogo openFileDialog1 para poder seleccionar el archivo XML donde están los datos grabados en la parte 1 y mostrarlos en el **dtgPersonal**.

Paso 1:

En el evento **Click** del botón **btnCargar** codifique lo siguiente:

```
private void btnCargar_Click(object sender, EventArgs e)
{
    try
    {
        // Mostramos el openFileDialog1 para que se elija la carpeta desde donde se cargara el XML
        // Establecemos propiedades del openFileDialog
        openFileDialog1.Multiselect = false;
        openFileDialog1.FileName = "";
        openFileDialog1.Filter = "Solo XML | *.xml"; // Alt + 124 |
        openFileDialog1.ShowDialog();

        // Obtenemos la ruta seleccionada
        String strRuta = openFileDialog1.FileName;

        // Si no se eligio ruta provocamos la excepcion...
        if (strRuta == "")
        {
            throw new Exception("Debe elegir un archivo XML para cargar.");
        }

        // Si se eligio un archivo XML , este de carga al DataSet y se enlaza el datagridView
        // dtgPersonal a la tabla 0 del DataSet
        DataSet dts = new DataSet();
        dts.ReadXml(strRuta);
        dtgPersonal.DataSource = dts.Tables[0];
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Como ya se vio en la parte 1, se configuran las propiedades del **openFileDialog1** y se asigna a la variable **strRuta** la ruta y nombre del archivo seleccionado con la expresión **String strRuta=openFileDialog1.FileName**.

Si no se seleccionó nombre alguno se genera una excepción indicando la necesidad de seleccionar un archivo XML para poder cargarlo.

En caso se haya seleccionado un archivo, se construye el **DataSet dts** para que aplicando el método **ReadXML** y estableciendo la variable **strRuta** como parámetro de dicho método se cargue un **DataTable** al **DataSet dts** y se enlace al **DataGridView dtgPersonal** con la propiedad **DataSource** la primera y única tabla que contiene con la expresión **dtgPersonal.DataSource=dts.Tables[0]**.

*NOTA: Recuerde que **Tables** es la colección de tablas de un **DataSet**. Las colecciones siempre se indexan desde 0.*

Pruebe la aplicación, seleccionando el archivo XML creado en la parte 1 y vera como se muestran los registros ingresados.

Paso 2:

Por último, lo que se requiere es que se seleccione un registro con hacer **Click** en cualquier fila del **DataGridView dtgPersona** y se muestre el formulario **frmFoto** con la respectiva foto de la persona seleccionada. Codificaremos lo siguiente en el evento **Click** del control **dtgPersonal**:

```
private void dtgPersonal_Click(object sender, EventArgs e)
{
    try
    {
        // Guardamos el DNI del registro donde hacemos clic en el datagrid , columna 0
        String strDNI = dtgPersonal.CurrentRow.Cells[0].Value.ToString();

        // Creamos una instancia del formulario foto, asignamos a la propiedad DNI de
        // dicho formulario (que ya esta definida) y mostramos modalmente el formulario.
        frmFoto frmF = new frmFoto();
        frmF.DNI = strDNI;
        frmF.ShowDialog();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

El código arriba indicado guarda en la variable **strDNI** el DNI del registro seleccionado en el **dtgPersonal (CurrentRow)** y de ese registro seleccionado se toma el valor de la celda 0 , que es la primera columna del **DataGridView**, precisamente donde está el DNI. Recuerde que en toda colección el primer valor se refiere a la posición 0 de la colección.

Luego se crea una instancia del formulario **frmFoto** llamada **frmF** y se le asigna a la propiedad DNI de dicha instancia el valor de la variable **strDNI**, para luego mostrar dicha instancia como cuadro de dialogo (modalmente) con el método **ShowDialog()**.

Es importante indicar que la propiedad DNI del formulario **frmFoto** ya está definida en la plantilla como se observa a continuación:

```
namespace ProyWinC_Sem03
{
    public partial class frmFoto : Form
    {
        // Propiedad para asignar desde el formulario frmVistaPersonal el
        // DNI de la persona seleccionada y se pueda mostrar su foto
        public String DNI { get; set; }

        public frmFoto()
        {
            InitializeComponent();
        }
    }
}
```

Ahora solo falta codificar el evento Load del formulario **frmFoto**, para que cuando este se cargue muestren en el control **pcbFoto** (incluido en el formulario **frmFoto**) la foto de la persona

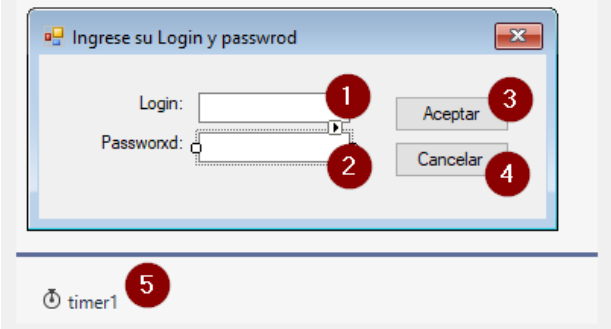
seleccionada desde el formulario **frmVistaPersonal**. En el evento Load del formulario **frmFoto** codifique lo siguiente:

```
private void frmFoto_Load(object sender, EventArgs e)
{
    try
    {
        // En base al DNI asignado en la propiedad del mismo nombre se obtiene la foto
        // desde la carpeta del repositorio de fotos.
        //NOTA: Si Ud. ha definido otra ruta para el repositorio, modifiquela segun lo que haya definido como ruta
        pcbFoto.Image = Image.FromFile(@"C:\Aleon\RepositorioFotos\" + this.DNI + ".jpg");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Como se puede observar, se asigna a la propiedad **Image** del **pictureBox pcbFoto** la ruta donde se encuentra la imagen cuyo nombre coincida con el DNI asignado a la propiedad DNI explicada líneas arriba. Para cargar una imagen en tiempo de ejecución se emplea la clase **Image** y el método **FromFile**, el cual demanda de la ruta donde está el archivo a mostrar. Pruebe el funcionamiento desde el formulario **frmVistaPersonal**.

PARTE 3: El formulario frmLogin

Como es conocido, todo aplicativo tiene un formulario donde el usuario debe ingresar sus credenciales para luego acceder al menú principal. En nuestro caso lo haremos desde el formulario **frmLogin**, cuyo diseño se muestra a continuación (este formulario debe agregarlo y diseñarlo Ud.):



1	TextBox - txtLogin
2	TextBox – txtPassword (PasswordChar=*)
3	Button - btnAceptar
4	Button - btnCancelar
5	Timer – timer1 (Enabled=true, Interval=1000)

Recuerde que el formulario **frmLogin** debe tener los bordes fijos, sin los botones de maximizar y minimizar y mostrarse al centro de la pantalla.

Las cajas de texto **txtLogin** y **txtPassword** permitirán ingresar las credenciales. En el caso de la caja de texto **txtPassword** se ha asignado a la propiedad **PasswordChar** el carácter *, de tal manera que al tipear la contraseña no se pueda ver su contenido, solo dicho carácter indicado. El control **timer1** estará programado para que ejecute el evento **Tick** cada segundo con la propiedad **Interval** en **1000**. Al hacer **Click** en el botón **btnAceptar** se deberá validar si el usuario y contraseña coinciden con las credenciales correctas. Dado que aun o se ha revisado el tema de acceso a base de datos, se asumirá que el usuario será **ISIL** y la contraseña será la cadena **12345**. Además, se le dará 20 segundos al usuario para que se registre, y si en ese tiempo no logra hacerlo o ha sobrepasado 3 intentos errados, el formulario se cerrara automáticamente indicándole con un mensaje que sobrepaso el tiempo o los intentos permitidos. En la zona de declaraciones debe definir 2 variables para manejar tanto el tiempo y los intentos:

```
namespace ProyWinC_Sem03
{
    public partial class frmLogin : Form
    {
        int intentos = 0; // Para cantidad de intentos fallidos
        int tiempo = 20; // Para cantidad de segundos como limite de tiempo para ingresar credenciales
        public frmLogin()
        {
            InitializeComponent();
        }

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

Paso 1:

Codifique el evento **Click** del botón **btnAceptar** con las siguientes líneas, siendo la que validan las credenciales las que están encerradas en el recuadro en rojo:

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    if(txtLogin.Text.Trim() != "" & txtPassword.Text.Trim() != "")
    {
        // Validamos las credenciales
        if (txtLogin.Text == "ISIL" & txtPassword.Text == "12345")
        {
            // Si son correctas mostramos el formulario MDIPrincipal
            this.Hide();// Ocultamos el Login
            timer1.Enabled = false;
            MDIPrincipal mdi = new MDIPrincipal();
            mdi.ShowDialog();
        }
        else // De lo contrario se contabiliza un intento fallido
        {
            MessageBox.Show("Usuario o Password incorrectos",
                "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Error);
            intentos +=1;
        }
    }
    else
    {
        MessageBox.Show("Usuario o Password obligatorios",
            "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Error);
        intentos +=1;
    }

    if (intentos == 3)
    {
        MessageBox.Show("Lo sentimos, sobrepaso el numero de intentos",
            "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Error);
        Application.Exit();
    }
}
```

Tal como se observa hay 2 condicionales anidadas:

La primera para validar que se hayan ingresado datos para el usuario y contraseña y la segunda (la marcada dentro del recuadro rojo) para validar si las credenciales proporcionadas son las correctas. Si la primera condicional es cierta se evalúa la segunda condicional para verificar si las credenciales proporcionadas son las correctas. Si las credenciales registradas son correctas se oculta el formulario **frmLogin** con el método **Hide**, el control **timer1** se desactiva, se crea una instancia del formulario **MDIPrincipal** llamada **mdi** y dicha instancia se muestra de manera modal. Si las credenciales proporcionadas no son correctas se envía un mensaje advirtiéndolo y se incrementa el contador de intentos en 1.

Si la primera condicional no es cierta se envía un mensaje indicando la obligatoriedad del ingreso de ambos datos, y también se incrementa en uno el contador de intentos.

Al final del procedimiento se evalúa si la cantidad de intentos llegó a 3, y de ser así se envía el mensaje indicando que se sobrepasó el límite de intentos errados, saliendo de la aplicación.

Paso 2:

Por último, para contabilizar el tiempo para que pueda ingresar sus credenciales correctamente en el evento **Tick** del control **timer1** codifique lo siguiente:

```
private void timer1_Tick(object sender, EventArgs e)
{
    // Cada segundo se decrementa la variable tiempo en menos 1, reflejando esto en el texto del formulario
    // para que el usuario sepa cuantos segundos le van restando para ingresar sus credenciales.
    tiempo -= 1;
    this.Text = "Ingrese su Usuario y contraseña. Le quedan..." + tiempo;

    // Si la variable tiempo llega a 0 , se le indica que el tiempo expiro y saldremos de la aplicacion.
    if (tiempo == 0)
    {
        timer1.Enabled = false;
        MessageBox.Show("Lo sentimos, sobrepaso el tiempo de espera.", "Mensaje",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        Application.Exit();
    }
}
```

Para probar el funcionamiento indique en la clase **Program** que la ejecución iniciara por el formulario **frmLogin**.

!!!Buen trabajo!!!!