

Tema 10 Acceso a datos con componentes.

1. Introducción.....	2
2. Creación del Proyecto, copia de la BD y creación del DataSet.....	3
3. Enlazar DataGridView con el Conjunto de Datos	9
4. Objetos BindingSource.....	11
Navegamos con el bindingSource.....	12
5. Controles enlazados con BindinSource.....	13
Propiedad DataBindings	13
6. Propiedades de BindingSource	15
7. Ejemplo Completo	17

1. Introducción.

Este tema vamos a ver de manera práctica cómo acceder a una BD con distintos **componentes** existentes en Visual Studio.

Copiaremos la BD al directorio App_Data de nuestro proyecto y crearemos un DataSet (conjunto de datos) a partir del cual trabajaremos en nuestra aplicación.

Veremos como enlazar un componente del tipo **DataGridView** al origen de datos que nos interese.

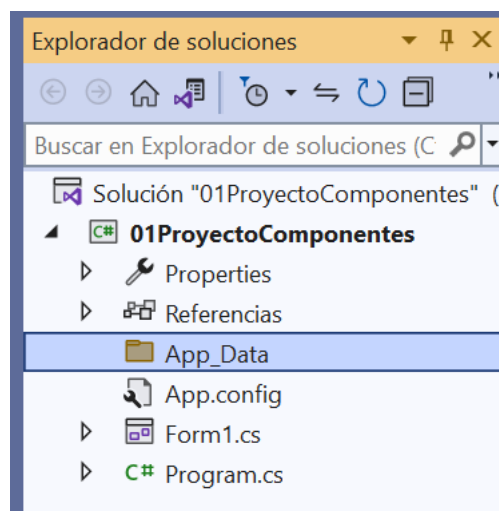
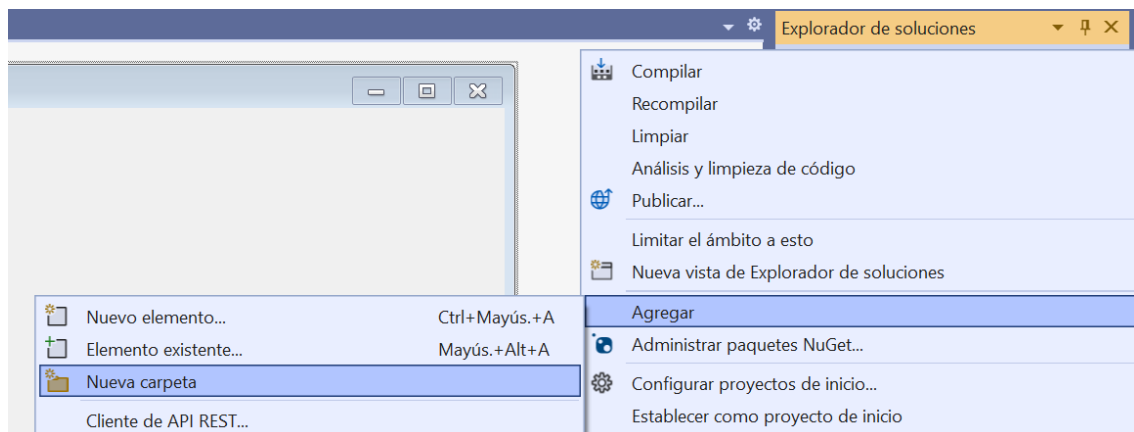
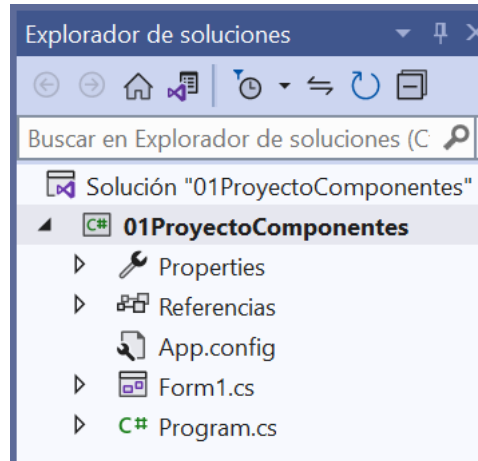
A continuación, estudiaremos el objeto **bindingSource** que actúa como intermediario entre el origen de datos y los controles que nos interesen.

Por último, veremos como enlazar los controles con el objeto **bindingSource**.

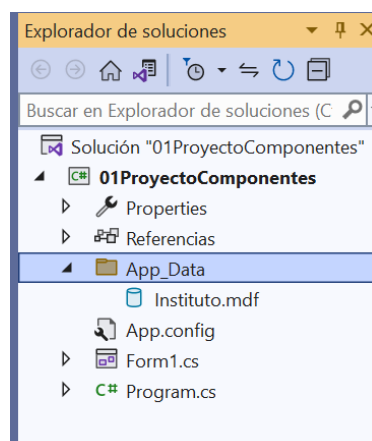
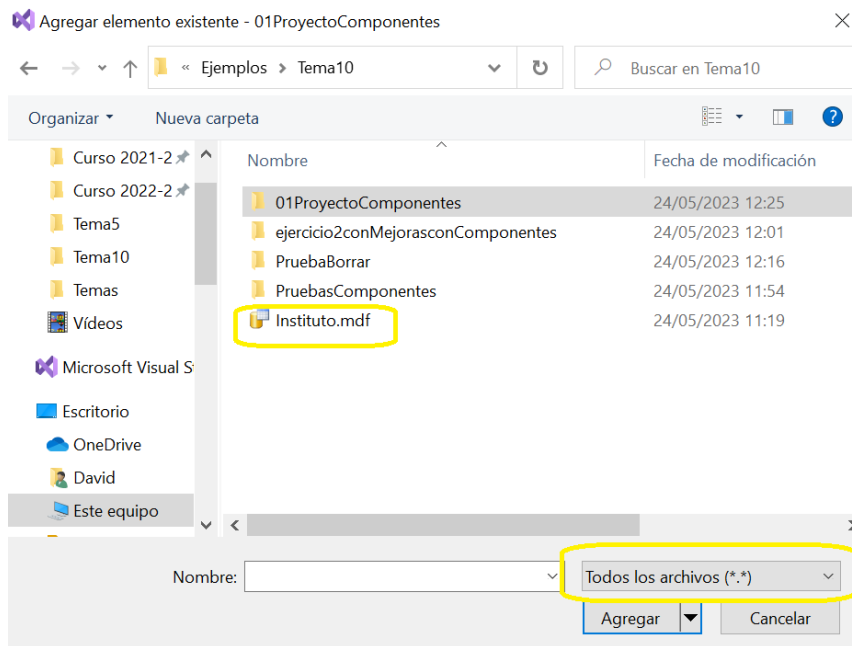
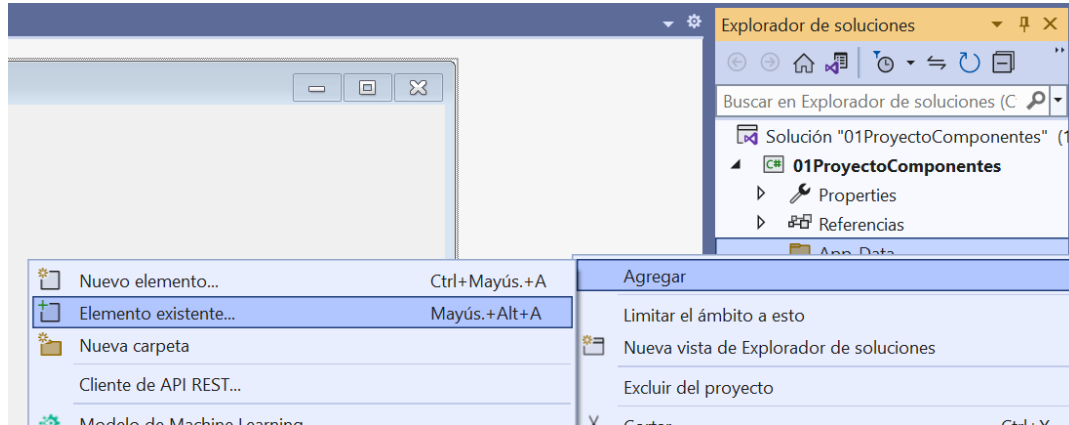
2. Creación del Proyecto, copia de la BD y creación del DataSet

Vamos a crear un nuevo proyecto.

Una vez lo tengamos vamos a crear la carpeta App_Data con el botón derecho en el Explorador de Soluciones:



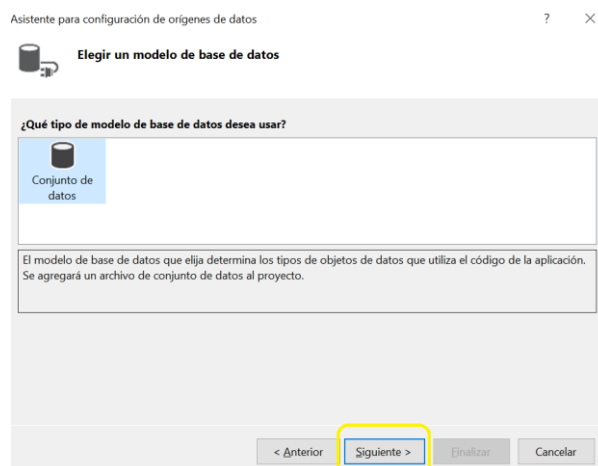
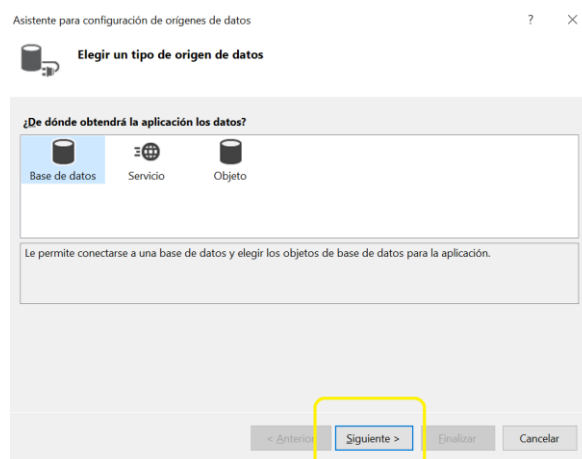
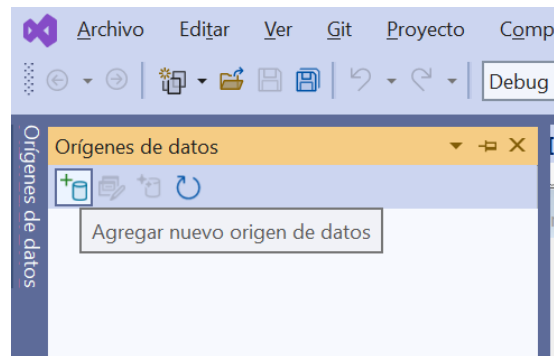
Pulsando el botón derecho sobre la carpeta App_Data en el explorador de soluciones, añadiremos el fichero Instituto.mdf de alguna de las localizaciones donde lo tengamos (así se copiará en nuestro proyecto):



Agregar el origen de datos y el dataSet.

En este momento vamos a crear de manera similar a como lo hacíamos en el tema anterior (aunque hay cambios) el conjunto de datos con el que va a trabajar nuestra aplicación.

Para ello nos vamos al menú **Proyecto->Agregar origen de datos**, o bien en **Orígenes de Datos (Ver->Otras ventanas->Orígenes de datos)**:





Elegir la conexión de datos

¿Qué conexión de datos debería usar la aplicación para conectarse a la base de datos?

Nueva conexión...

Esta cadena de conexión parece contener datos confidenciales (por ejemplo, una contraseña) que son necesarios para conectarse con la base de datos. Sin embargo, almacenar datos confidenciales en la cadena de conexión puede suponer un riesgo para la seguridad. ¿Desea incluir estos datos en la cadena de conexión?

☐ No, excluir los datos confidenciales de la cadena de conexión. Estableceré esta información en el código de mi aplicación.

☐ Sí, incluir datos confidenciales en la cadena de conexión.

☐ Mostrar la cadena de conexión que se guardará en la aplicación

Elegimos el fichero Instituto.mdf de App_Data:

Agregar conexión ? X

Especifique la información para conectarse al origen de datos seleccionado o haga clic en "Cambiar" para elegir otro origen o proveedor de datos.

Origen de datos:

Cambiar...

Nombre de archivo de base de datos (nuevo o existente):

Examinar...

Conexión con el servidor

☒ Usar autenticación de Windows

☐ Usar autenticación de SQL Server

Nombre de usuario:

Contraseña:

Seleccione el archivo de base de datos de SQL Server

← → ↑ ↓ 01ProyectoCo... > App_Data Buscar en App_Data

Organizar Nueva carpeta

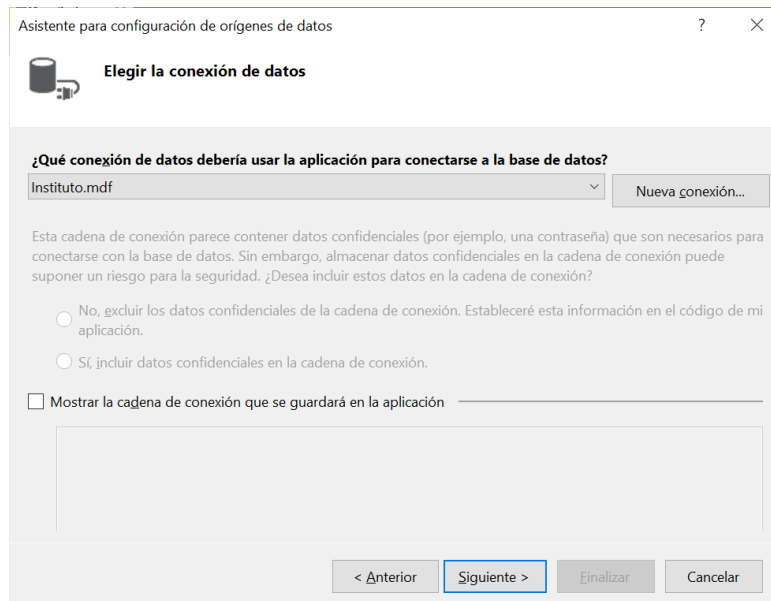
Nombre	Fecha de modificación
Instituto.mdf	24/05/2023 12:31

Nombre: Instituto.mdf Bases de datos de Microsoft SC

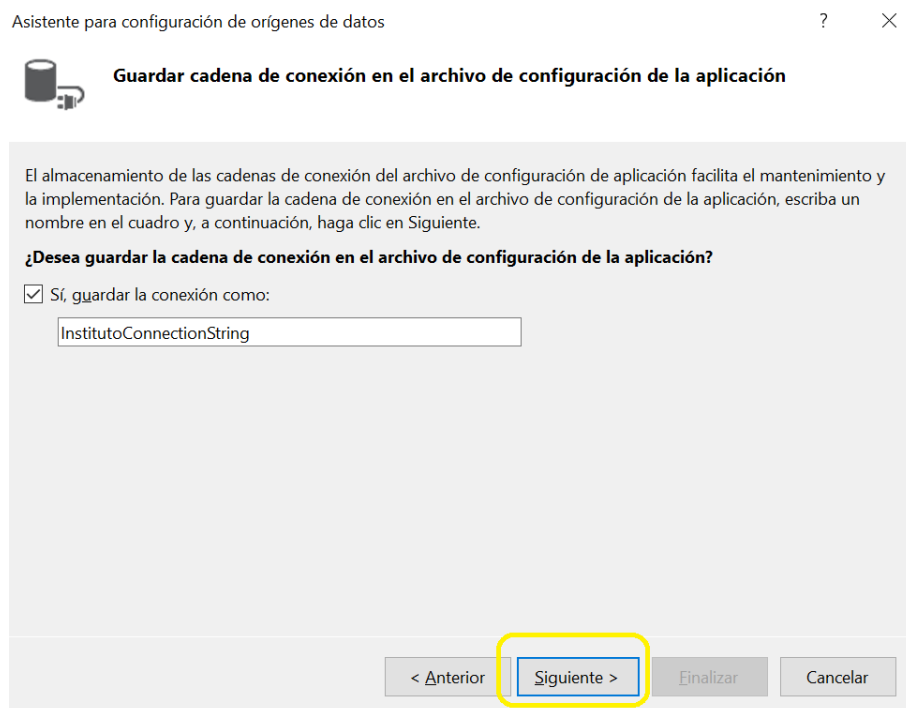
Abrir Cancelar

Aceptamos.

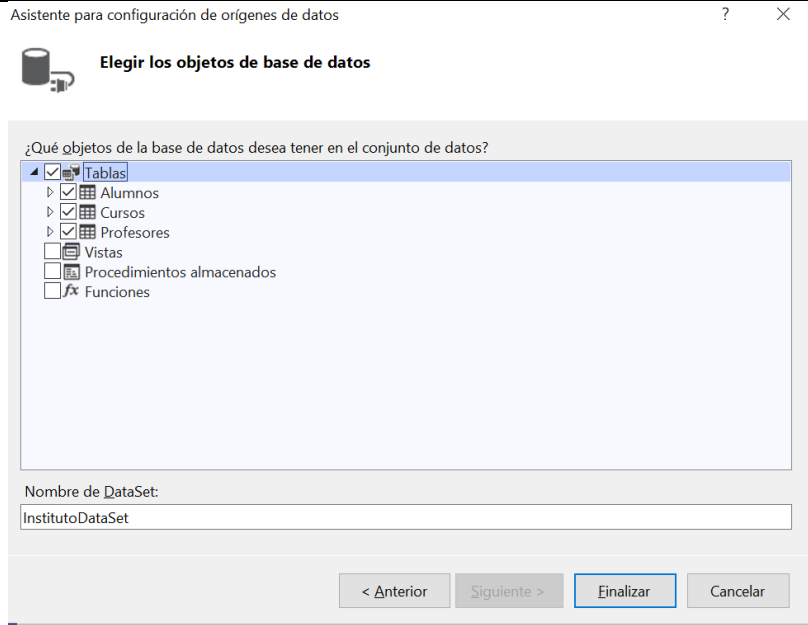
Volvemos a la pantalla de elegir la conexión de datos:



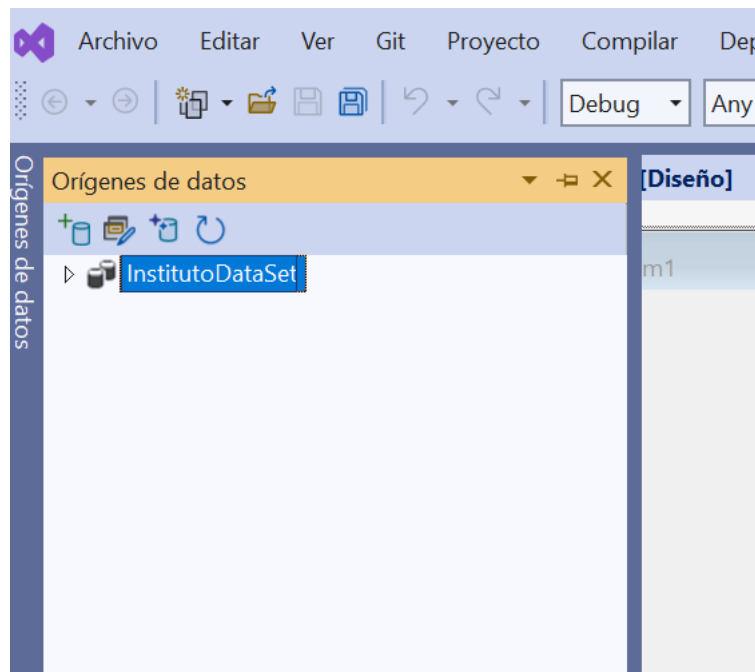
Y en esta ocasión **SÍ QUE PULSAMOS LA TECLA SIGUIENTE.**



Y de nuevo Siguiente, para pasar a la pantalla donde elegimos lo que queremos que forme parte de nuestro dataSet. Podemos elegir una sola tabla o las tres en este caso:



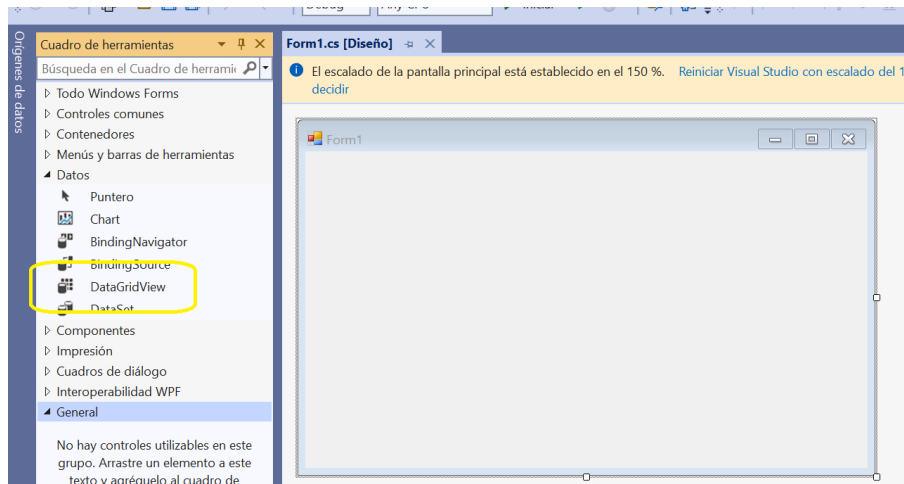
Pulsamos finalizar y nos encontraremos en Orígenes de datos con nuestro dataSet ya creado (en esta ocasión lo estamos creando de manera visual, no en el código):



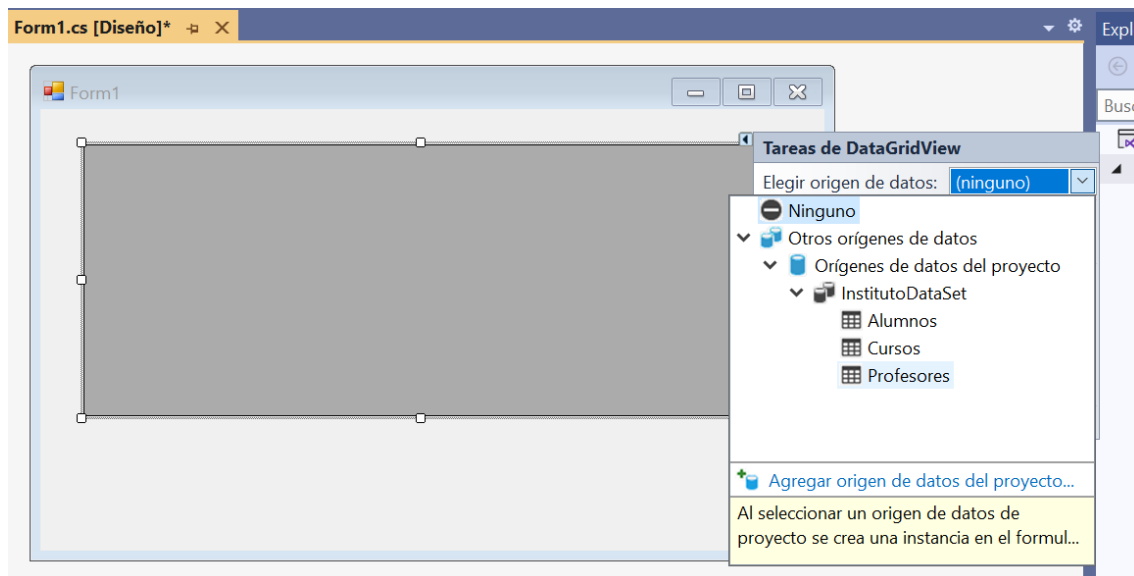
3. Enlazar DataGridView con el Conjunto de Datos

DataGridView es un componente personalizable de Visual Studio que nos permite mostrar datos en una cuadrícula.

En primer lugar, arrastraremos el componente desde el cuadro de herramientas hasta nuestro formulario:

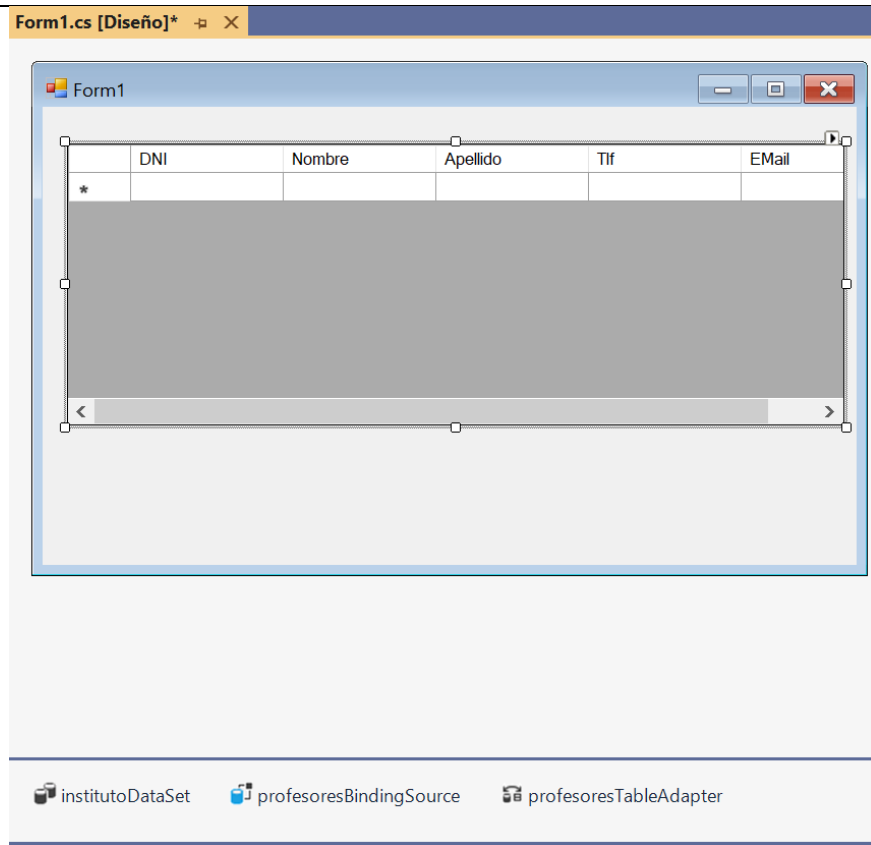


Eligiendo a continuación en Tareas de DataGridView el origen de datos se mostrará. En este caso, la tabla Profesores del dataSet que creamos anteriormente:



Fijémonos que en ese momento se añade a nuestro formulario los componentes institutoDataSet, profesoresBindingSource y profesoresTableAdapter:

Form1.cs [Diseño]*

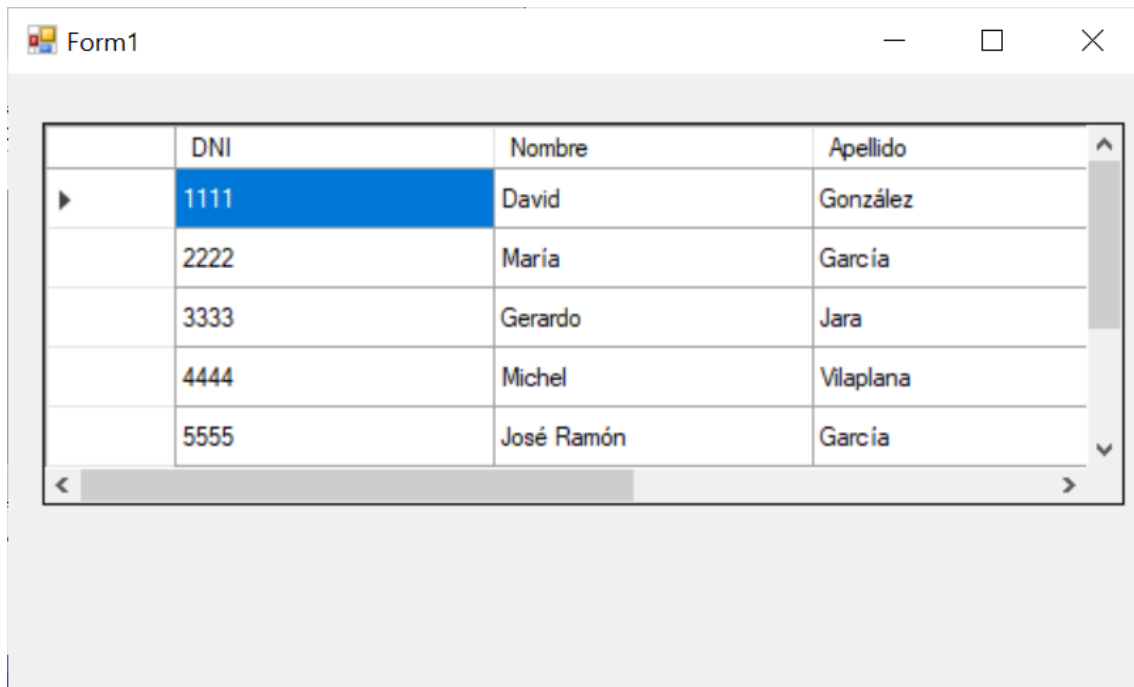


Form1

	DNI	Nombre	Apellido	Tlf	EMail
*					

institutoDataSet profesoresBindingSource profesoresTableAdapter

Si ejecutamos nuestro programa nos aparece el formulario con la tabla correspondiente a los profesores de la BD.



Form1

	DNI	Nombre	Apellido
▶	1111	David	González
	2222	María	García
	3333	Gerardo	Jara
	4444	Michel	Vilaplana
	5555	José Ramón	García

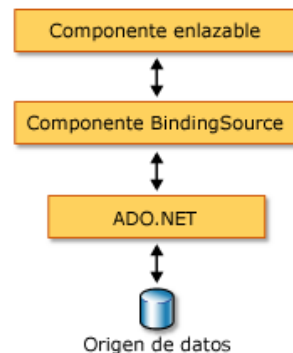
4. Objetos BindingSource

El objeto *bindingSource* se utiliza para **encapsular un origen de datos** (normalmente, un **DataSet**) **para enlazarlo a los controles del formulario**.

El componente *bindingSource* **actúa como capa intermedia entre los datos y los controles del formulario**. A partir de aquí, toda interacción con los datos (navegación, ordenación, filtrado, actualización) se realiza con el componente *bindingSource*.

El componente *bindingSource* **se enlaza al origen de datos** mediante sus propiedades **DataSource** (hace referencia a la **base de datos**) y **DataMember** (hace referencia a la **tabla**).

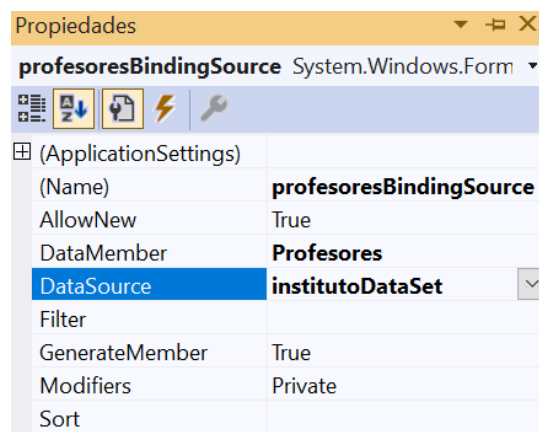
Un **control se enlaza al componente *bindingSource*** mediante su propiedad **DataBindings** (se estudiará más adelante).



Más adelante, veremos cómo crear un *bindingSource* y las distintas propiedades que nos podemos encontrar y utilizar con el mismo.

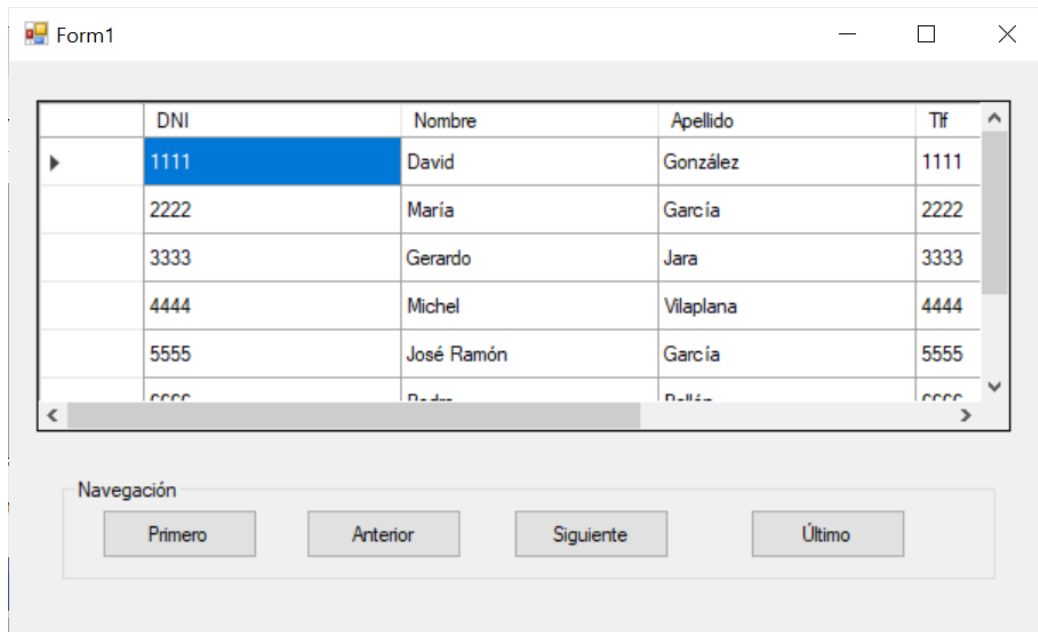
De momento vamos a utilizar en nuestro formulario *profesoresBindingSource* que se creó automáticamente poner el origen de datos a nuestro *DataGridView*.

Si seleccionamos sus propiedades podemos ver su *DataSource* (hace referencia al *dataSet*), *DataMember* (hace referencia a la tabla):



Navegamos con el bindingSource

Vamos a utilizar los métodos **MoveFirst**, **MoveNext**, **MoveLast** y **MovePrevious** para “navegar” por los registros de la BD. Para ello en nuestro formulario creamos los botones Primero, Anterior, Último y Siguiente de manera similar a como lo hicimos en el tema anterior:



	DNI	Nombre	Apellido	Tlf	
▶	1111	David	González	1111	^
	2222	María	García	2222	
	3333	Gerardo	Jara	3333	
	4444	Michel	Vilaplana	4444	
	5555	José Ramón	García	5555	
<	6666	David	González	6666	>

Navegación

Primero Anterior Siguiente Último

Y tendremos este código que nos permite navegar por los registros, evitando además errores al darle a Siguiente en el último registro:

```
private void btnPrimero_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveFirst();
}

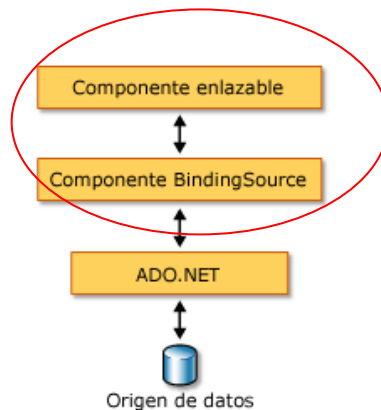
private void btnAnterior_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MovePrevious();
}

private void btnSiguiente_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveNext();
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveLast();
}
```

5. Controles enlazados con BindinSource.

Una vez enlazado el origen de datos al objeto *BindingSource* llega el momento de **enlazar los controles al objeto *BindingSource*** para, de este modo, completar la cadena:



Un **control enlazado** es aquel que contiene **alguna de sus propiedades vinculadas a una fuente de datos**. Aunque la vinculación se puede realizar directamente entre el control y el origen de datos (*DataSet*), lo más común es **utilizar el objeto *BindingSource*** como intermediario, ya que éste **proporciona métodos y propiedades para facilitar la navegación, edición y modificación de los datos**.

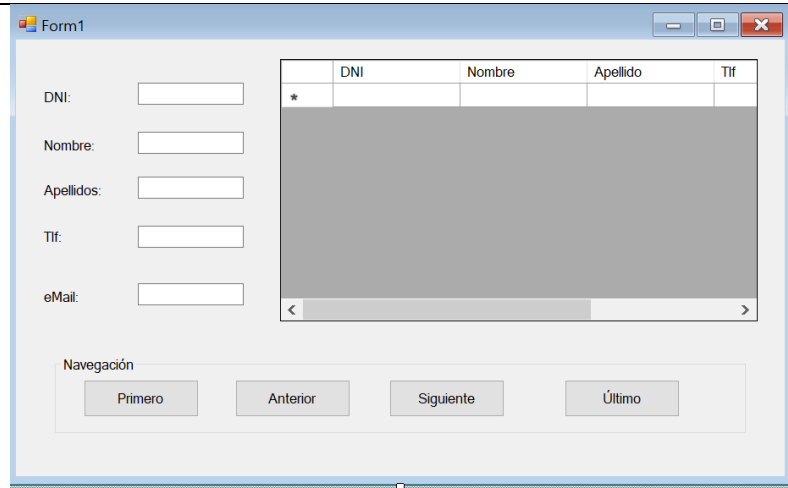
Los controles que más comúnmente se utilizan para enlazar son: *TextBox*, *ComboBox*, *ListBox*, *CheckBox*, *RadioButton*, *DataGridView* y *PictureBox*.

Propiedad DataBindings

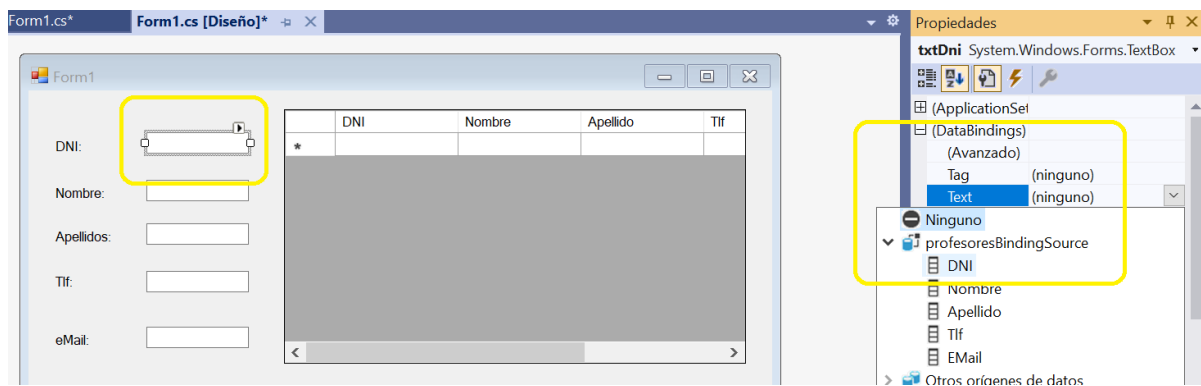
En Visual.NET se puede **enlazar cualquier propiedad de un control a cualquier estructura que contenga datos**. Aunque lo más normal es que se utilice sólo la **propiedad de presentación** del control para ello. Por ejemplo, en el caso de un *TextBox*, la propiedad más útil para enlazar es su propiedad *Text*, que es la que muestra el texto que aparece en su interior.

Para definir qué propiedad del control se enlaza a datos y qué datos son los que van a aparecer en dicha propiedad se utiliza la propiedad **DataBindings** del control. Esta propiedad muestra directamente la propiedad de presentación del control (la más común) y un menú desplegable a su derecha con la lista de orígenes de datos a los que se puede enlazar. Como ya hemos comentado, se elegirá un dato disponible a través del objeto *BindingSource* de la aplicación.

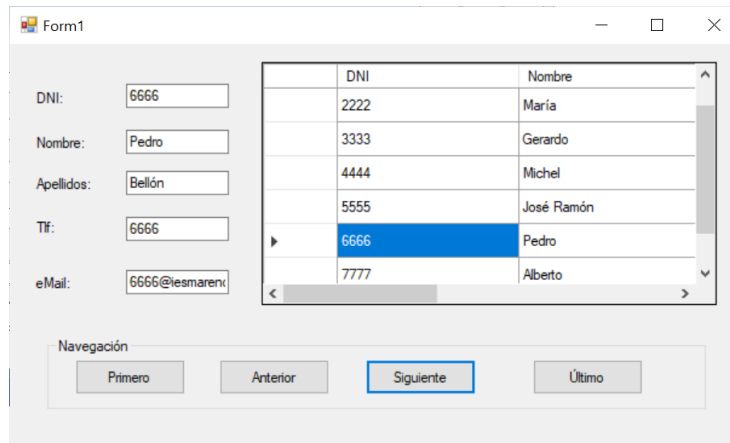
En nuestro caso, vamos a añadir a nuestro formulario una serie de *TextBox* donde se representarán los campos de la tabla:



Lo que hacemos ahora es enlazar mediante la propiedad Databindings del control que nos interese y su propiedad Text con el origen de datos (en nuestro caso profesoresBindingSource) y el campo que queremos:



Al hacer esto en todos los textBox y con lo que tenemos implementado, podemos navegar por nuestro conjunto de datos, bien a través de los botones, bien a través del dataGridView:



6. Propiedades de BindingSource

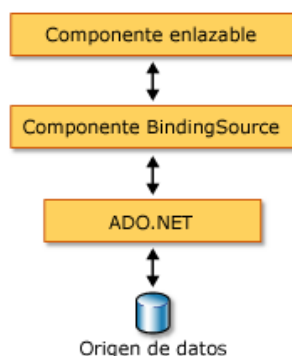
Hacemos un pequeño resumen de lo visto hasta aquí.

El objeto *bindingSource* se utiliza para **encapsular un origen de datos** (normalmente, un **DataSet**) **para enlazarlo a los controles del formulario**.

El componente *bindingSource* **actúa como capa intermedia entre los datos y los controles del formulario**. A partir de aquí, toda interacción con los datos (navegación, ordenación, filtrado, actualización) se realiza con el componente *bindingSource*.

Se enlaza al origen de datos mediante sus propiedades **DataSource** (hace referencia a la **base de datos**) y **DataMember** (hace referencia a la **tabla**).

Un **control** se **enlaza al componente *bindingSource*** mediante su propiedad **DataBindings**



Miembros del objeto BindingSource

Una vez establecido el origen de datos al que se enlaza, el objeto *bindingSource* proporciona métodos y propiedades para acceder a los datos subyacentes. Para ello utiliza un **puntero de lectura: los controles enlazados a un *BindingSource*** (con la excepción del control DataGridView, que muestra todos los datos de una tabla) **sólo muestran los datos correspondientes a un registro determinado** (en concreto, se muestra el número de registro indicado en la propiedad **Position**). Este registro es accesible a través de la propiedad **Current**, mientras que la lista completa de datos es accesible a través de la propiedad **List**.

Para **cambiar el registro mostrado por el puntero de lectura** se puede utilizar la propiedad **position** o ejecutar alguno de los **métodos** que se verán a continuación.

Propiedades

- **DataSource** – Establece el **origen de datos** (normalmente un *DataSet*) al que se enlaza el componente.
- **DataMember** – Establece la **tabla** dentro del origen de datos a la que se enlaza el componente.

- **Current** – Obtiene una **referencia al registro actual**. Cuando el origen de datos es un *DataSet*, un *DataTable* o un *DataGridView* la propiedad *Current* devuelve un objeto de la clase *DataRowView*. Es posible acceder a los campos del registro mediante la propiedad *Item*:

bs.Current.Item("Cantidad")

- **Position** – Integer. Devuelve o establece la **posición del registro actual**.
- **Count** – Integer. Devuelve el número de registros accesibles.
- **Sort** – String. Obtiene o establece los nombres de columna utilizados para **ordenar** las filas del origen de datos.

Métodos

- **Remove(objeto)** – Elimina el registro indicado.
- **RemoveCurrent** – Elimina el registro actual.
- **RemoveAt(posicion)** – Elimina el registro en la posición especificada.
- **EndEdit** – Guarda las modificaciones realizadas en los registros en el origen de datos al que apunta el objeto *BindingSource*.
- **CancelEdit** – Cancela las modificaciones realizadas en los registros (siempre que éstas no se hayan guardado con el método *EndEdit*).
- **AddNew** – Crea un nuevo registro vacío.
- **Find(objeto)** – Busca un registro determinado en el origen de datos. Devuelve el índice en el que se encuentra.
- **MoveLast** – Desplaza el puntero de lectura al último registro: la propiedad *Current* apuntará al último registro.
- **MoveNext** – Desplaza el puntero de lectura al siguiente registro.
- **MovePrevious** – Desplaza el puntero de lectura al registro anterior.
- **MoveFirst** – Desplaza el puntero de lectura al primer registro.

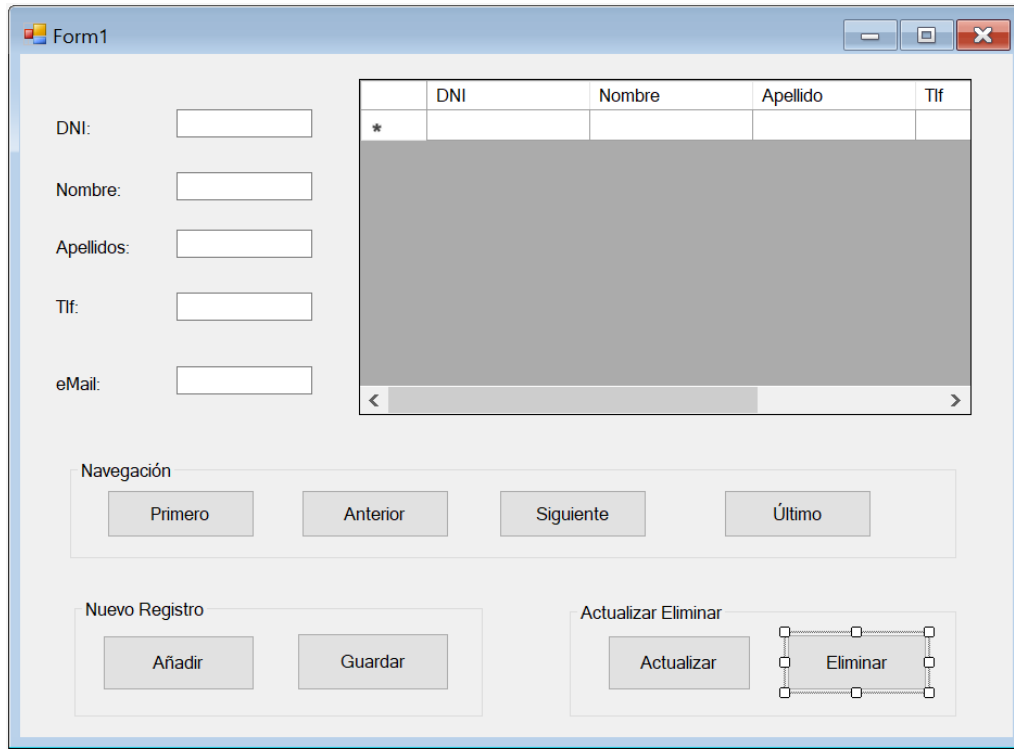
Eventos

- **CurrentChanged** – Se produce cuando la propiedad *Current* cambia.
- **PositionChanged** – Se produce después de haber cambiado el valor de la propiedad *Position*.

7. Ejemplo Completo

Vamos a continuación a ver un ejemplo completo del estilo del tema 9 y ver cómo podemos utilizar el componente `bindingSource` y los componentes enlazados al mismo para hacer las operaciones CRUD con la BD.

Tendríamos el formulario:



	DNI	Nombre	Apellido	Tlf
*				

Hemos puesto además en el formulario un label para mostrar el número de registro actual y total.

Repasamos el código de el evento Load y los botones de navegación:

```
// Este subprograma muestra el registro actual y total
private void mostrarDatos()
{
    lblPosicion.Text = (profesoresBindingSource.Position + 1) +
        " de " + profesoresBindingSource.Count;
}
private void Form1_Load(object sender, EventArgs e)
{
    // TODO: esta línea de código carga datos en la tabla
    'institutoDataSet.Profesores' Puede moverla o quitarla según sea necesario.
    this.profesoresTableAdapter.Fill(this.institutoDataSet.Profesores);

    mostrarDatos();
}
```

```
private void btnPrimero_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveFirst();
    mostrarDatos();
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MovePrevious();
    mostrarDatos();
}

private void btnSiguiente_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveNext();
    mostrarDatos();
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    profesoresBindingSource.MoveLast();
    mostrarDatos();
}
```

Vamos a ver ahora el código para añadir un nuevo registro. Al igual que en el tema anterior tenemos un botón que crea un nuevo registro (borrando los textBox) y otro que nos permite guardar lo que haya introducido el usuario (este botón podría controlar si lo introducido es correcto...):

```
private void btnAnadir_Click(object sender, EventArgs e)
{
    // Añade un nuevo registro
    profesoresBindingSource.AddNew();
}

private void btnGuardar_Click(object sender, EventArgs e)
{
    // Termina la edición del registro
    profesoresBindingSource.EndEdit();

    // Actualiza la BD
    profesoresTableAdapter.Update(institutoDataSet.Profesores);

    mostrarDatos();
}
```

Nos encontramos ahora con un pequeño problema, y es que, si probamos nuestro programa, aparentemente funciona bien.

Pulsamos el botón de Añadir:

Form1

1 de 7

DNI:

Nombre:

Apellidos:

Tlf:

eMail:

DNI	Nombre
4444	Michel
5555	José Ramón
6666	Pedro
7777	Alberto

Navegación

Primero Anterior Siguiente Último

Nuevo Registro

Añadir Guardar

Actualizar Eliminar

Actualizar Eliminar

Rellenamos los datos del nuevo registro y pulsamos Guardar:

Form1

8 de 8

DNI:

Nombre:

Apellidos:

Tlf:

eMail:

DNI	Nombre
4444	Michel
5555	José Ramón
6666	Pedro
7777	Alberto
8888	Pepito

Navegación

Primero Anterior Siguiente Último

Nuevo Registro

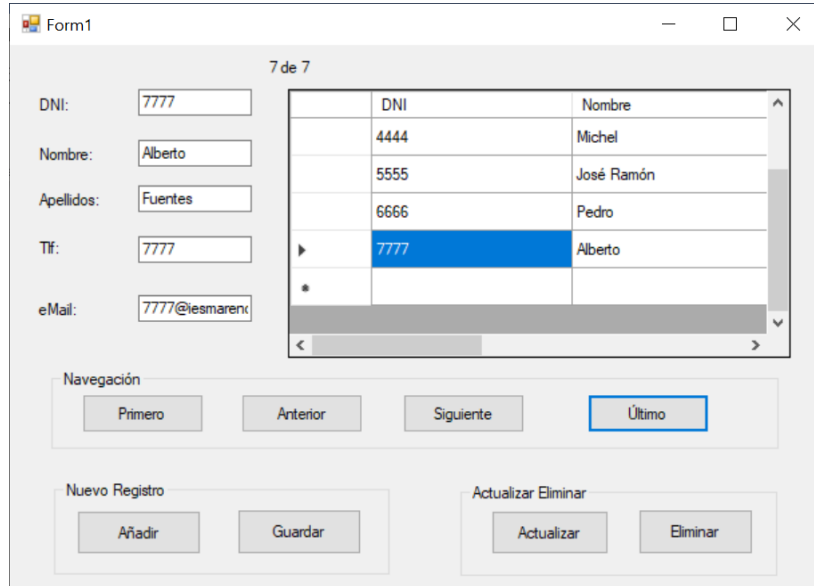
Añadir Guardar

Actualizar Eliminar

Actualizar Eliminar

Podemos navegar y tenemos el nuevo registro.

El problema ocurre cuando cerramos el programa y volvemos a ejecutarlo. Resulta que ese nuevo registro que habíamos añadido no está:



DNI	Nombre
4444	Michel
5555	José Ramón
6666	Pedro
7777	Alberto

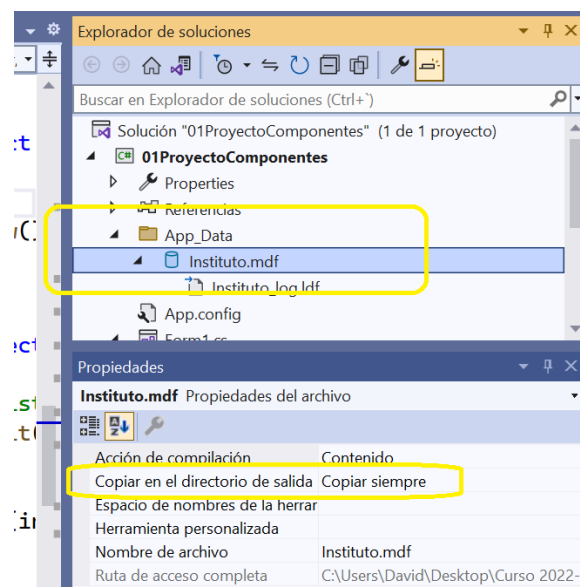
¿Por qué está sucediendo esto?

Cuando trabajamos en Visual Studio con un fichero local de BD (el que añadimos al principio del tema: App_Data/Instituto.mdf), que está situado dentro de la carpeta de nuestra aplicación, VS hace una copia de esa carpeta a la carpeta donde está el ejecutable de Debug, que es el que ejecutamos nosotros al depurar dentro del entorno (Proyecto/bin/Debug).

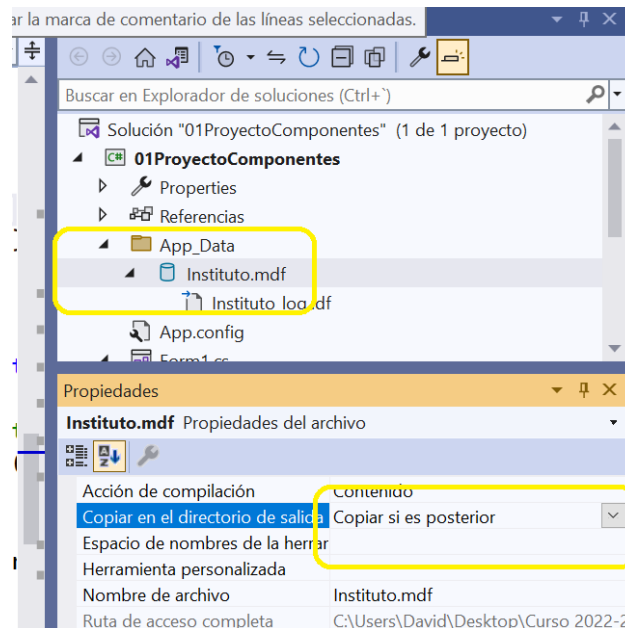
Por tanto, la BD con la que trabajamos al ejecutar está en esa carpeta.

El problema es que, por defecto, VS hace una nueva copia de ese fichero a esa carpeta cada vez que ejecutamos el programa, y por tanto perdemos los cambios que hayamos realizado en la BD.

Podemos verlo mediante las propiedades del fichero:



Podemos, por tanto, evitar este problema cambiando esa propiedad:



De esta manera, se conservarán los cambios en el fichero de BD situado en bin/Debug/App_Data.

Vemos ahora el código de Actualizar y Eliminar (se podría mejorar preguntando si realmente se quiere hacer):

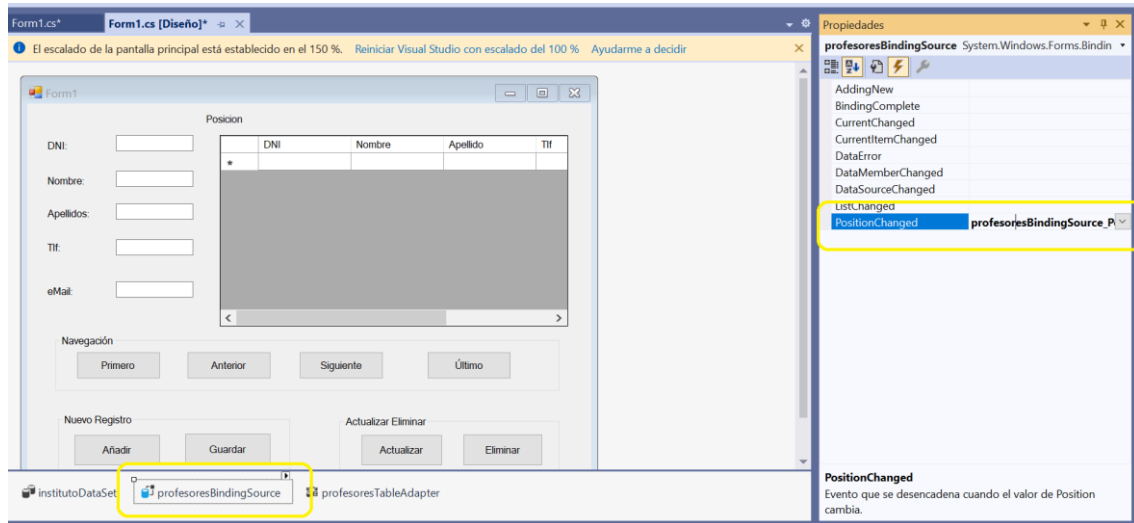
```
private void btnActualizar_Click(object sender, EventArgs e)
{
    // Actualiza la BD
    profesoresTableAdapter.Update(institutoDataSet.Profesores);
}

private void btnEliminar_Click(object sender, EventArgs e)
{
    profesoresBindingSource.RemoveCurrent();

    // Actualiza la BD
    profesoresTableAdapter.Update(institutoDataSet.Profesores);
}
```

Un pequeño detalle del ejercicio realizado hasta ahora es que el label donde mostramos los registros no se actualiza cuando nos desplazamos por los registros no a través de los botones de navegación, sin mediante el dataGridView.

Esto lo podemos solucionar mediante el evento PositionChange del bindingSource:



```
private void profesoresBindingSource_PositionChanged(object sender,
                                                    EventArgs e)
{
    mostrarDatos();
}
```

De esta manera incluso podemos quitar el método `mostrarDatos` de los botones de navegación, ya que cada vez que cambia la posición del registro se ejecuta el evento que acabamos de realizar.

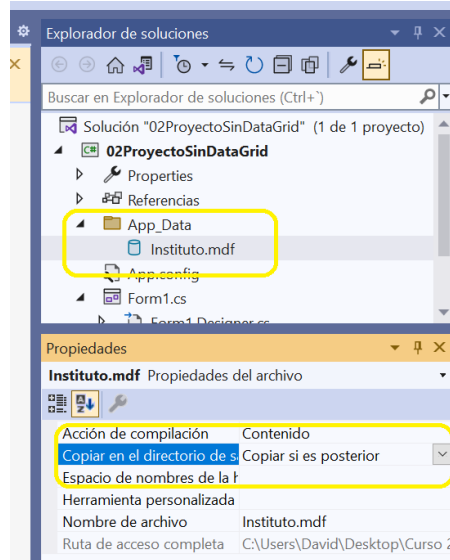
Trabajar sin el DataGridView

Para finalizar el tema vamos a ver cómo podríamos realizar un ejercicio similar al anterior, pero sin el componente `DataGridView`, ya que al elegir los datos del mismo automáticamente nos aparece el `BindingSource` y el `TableAdapter`.

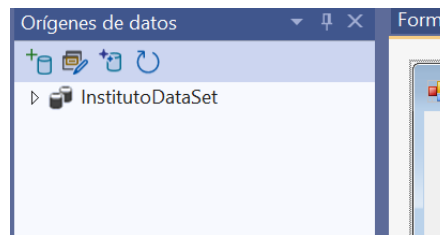
Imaginemos que hemos creado un formulario con el siguiente aspecto:



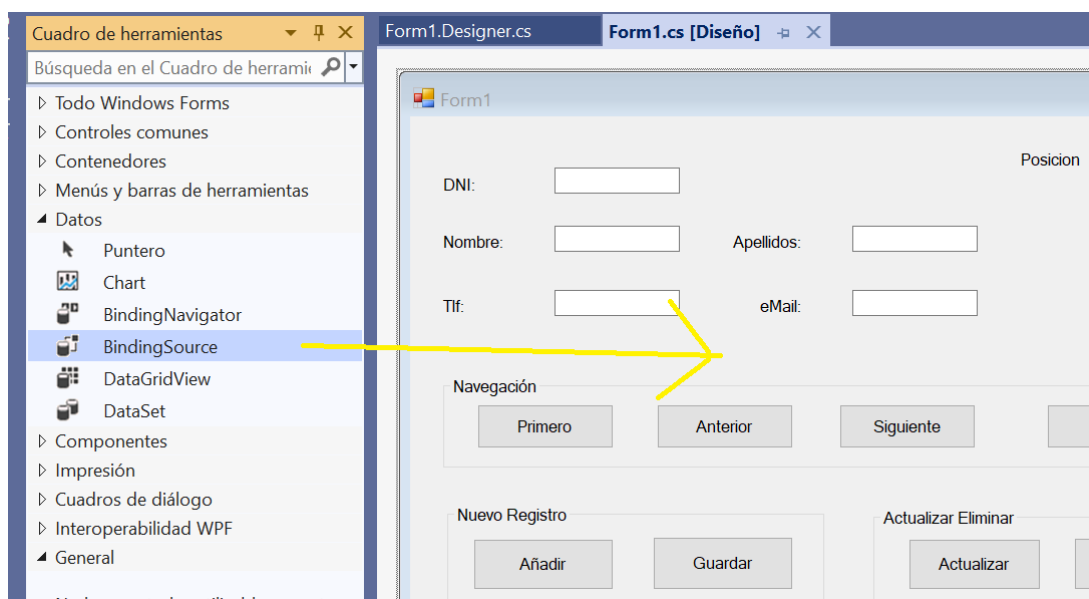
Recordemos, como vimos en apartados anteriores añadir la BD al proyecto y además cambiar la copia en el directorio de salida:



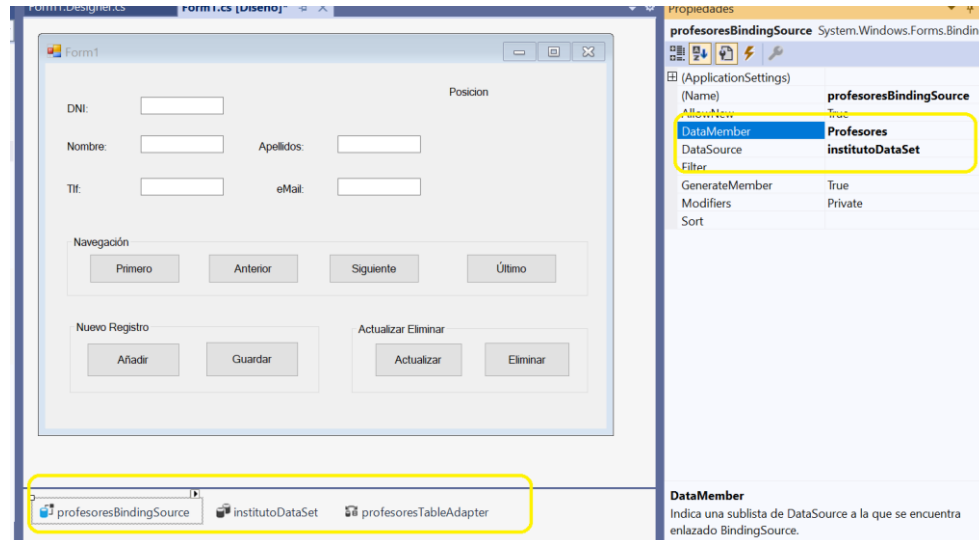
A continuación, creamos el **dataSet** como vimos en el apartado 2:



A continuación, arrastramos un componente **BindingSource** del cuadro de herramientas:

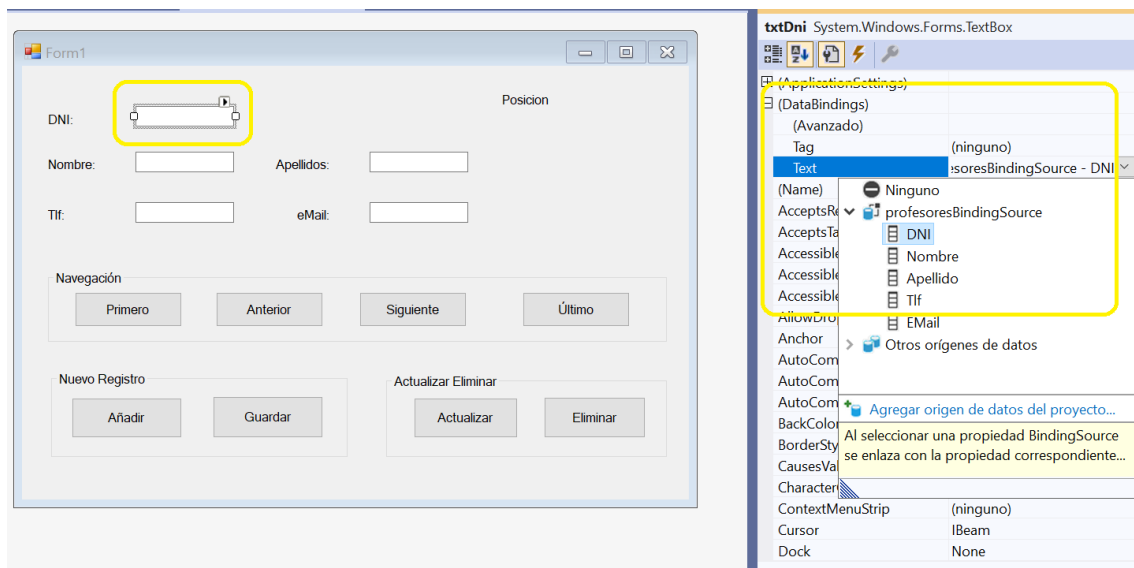


Y, para terminar, elegimos en las propiedades del BindingSource el origen de datos y el DataMember:



Al hacerlo, notamos que ha añadido al formulario un componente de tipo DataSet y otro de tipo TableAdapter.

Por último, elegimos la propiedad DataBindings de los componentes:



A partir de este punto nos serviría el código que hemos visto anteriormente en este apartado.