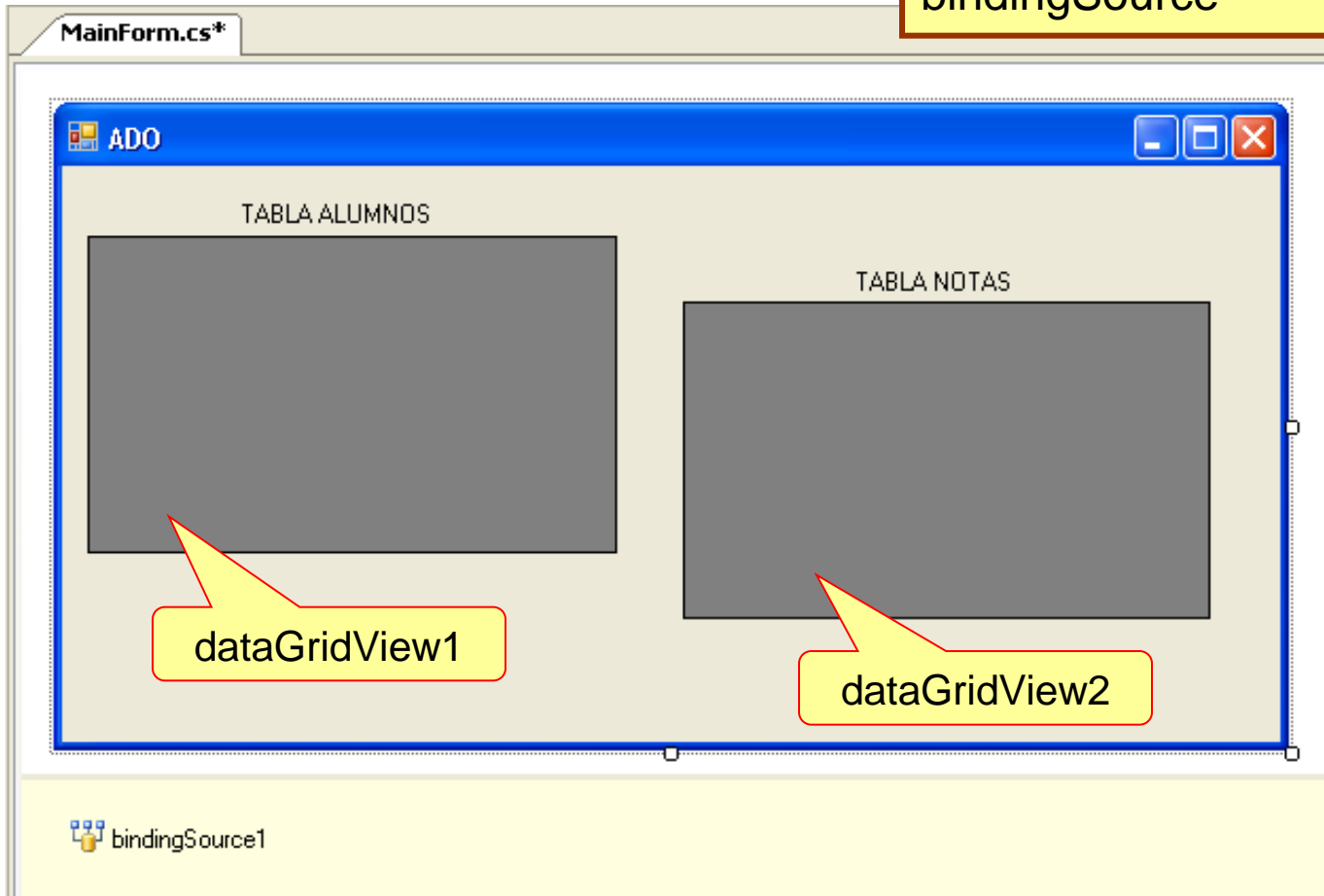


ADO – Persistencia - XML

ADO.NET

Agregue en un formulario dos
DataGridView y un
bindingSource



ADO.NET

```
public partial class MainForm : Form
{
    DataSet ds=new DataSet("Mi dataset");
    public MainForm()
    {
        // The InitializeComponent() call is required for Windows Forms designer support.
        //
        InitializeComponent();
        //
        // TODO: Add constructor code after the InitializeComponent() call.
        //

        //Tabla Maestro
        DataTable dt=new DataTable("Alumnos");
        dt.Columns.Add("Nombre",typeof(string));
        dt.Columns.Add("Nro",typeof(int));
        ds.Tables.Add(dt);

        //Tabla Detalle
        dt=new DataTable("Notas");
        dt.Columns.Add("Nro",typeof(int));
        dt.Columns.Add("Nota",typeof(float));
        ds.Tables.Add(dt);
    }
}
```

En el constructor del formulario, luego de la llamada al método `InitializeComponent()`, agregue el siguiente código

Continúa ...

ADO.NET

```
//Relación Maestro/Detalle
DataColumn colPadre=ds.Tables["Alumnos"].Columns["nro"];
DataColumn colHija=ds.Tables["Notas"].Columns["nro"];
DataRelation dr=new DataRelation("relacion",colPadre,colHija);
ds.Relations.Add(dr);

//Enlace de controles del formulario
bindingSource1.DataSource = ds.Tables["Alumnos"];
dataGridView1.DataSource=bindingSource1;
dataGridView2.DataSource=bindingSource1;
dataGridView2.DataMember = "relacion";
```

Ejecute y compruebe su funcionamiento.

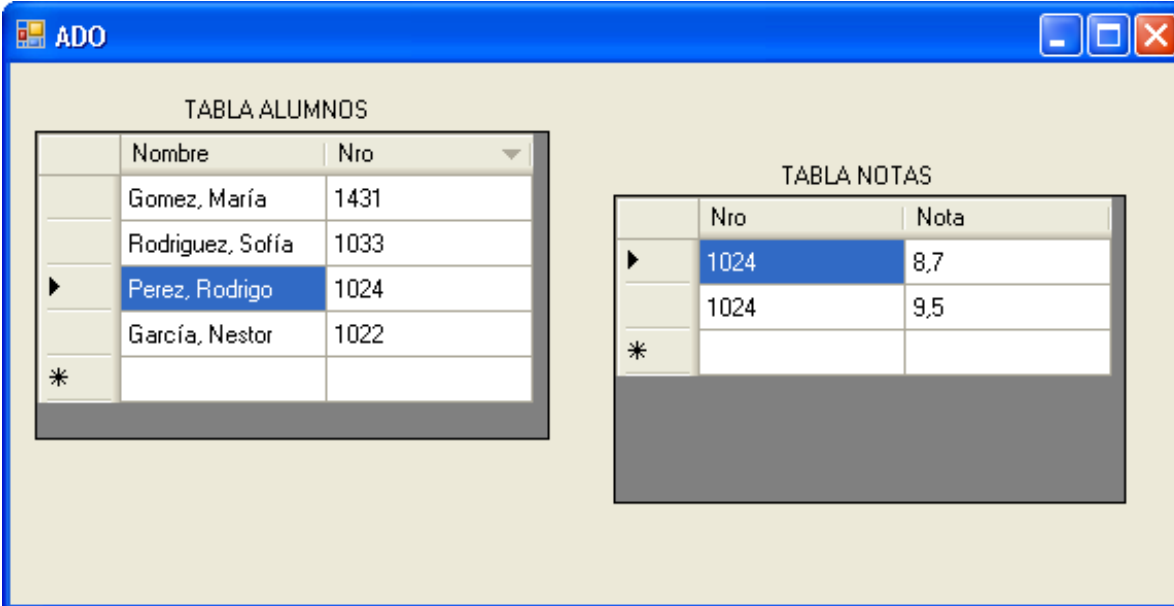
También puede agregar datos por código ...

ADO.NET

```
//Carga de datos Alumnos  
dt = ds.Tables["Alumnos"];  
dt.Rows.Add("Perez, Rodrigo",1024);  
dt.Rows.Add("García, Nestor",1022);  
dt.Rows.Add("Rodriguez, Sofía",1033);  
dt.Rows.Add("Gomez, María",1431);
```

```
//Carga de datos Notas  
dt=ds.Tables["Notas"];  
dt.Rows.Add(1024,8.7);  
dt.Rows.Add(1024,9.5);  
dt.Rows.Add(1022,7);  
dt.Rows.Add(1022,5);
```

Ejecute



ADO

TABLA ALUMNOS

	Nombre	Nro
	Gomez, María	1431
	Rodriguez, Sofía	1033
▶	Perez, Rodrigo	1024
	García, Nestor	1022
*		

TABLA NOTAS

	Nro	Nota
▶	1024	8,7
	1024	9,5
*		

ADO.NET

■ Componente **BindingSource**

- ❑ Simplifica el enlace de datos de los controles de un formulario
- ❑ Proporcionan notificación de cambios y otros servicios como ordenación y filtrado de datos.
- ❑ El elemento actual se puede recuperar mediante la propiedad **Current**

ADO.NET

■ Ejercitación

- Agregue el botón Filtrar y compruebe el comportamiento de la instrucción:

□ `bindingSource1.Filter="nro=1024";`

ADO

TABLA ALUMNOS

	Nombre	Nro
	Gomez, María	1431
	Rodriguez, Sofía	1033
▶	Perez, Rodrigo	1024
	García, Nestor	1022
*		

Filtrar

TABLA NOTAS

	Nro	Nota
▶	1024	8,7
	1024	9,5
*		

ADO.NET

■ Ejercitación

Compruebe por separado el funcionamiento de las siguientes expresiones:

```
bindingSource1.Filter="nro=1024";
```

```
bindingSource1.Filter="nombre like 'G*'";
```

```
bindingSource1.Filter="nombre like 'G*' and  
Nro > 1400";
```

```
bindingSource1.Filter="";
```


ADO.NET

■ Ejercitación

Compruebe por separado el funcionamiento de las siguientes expresiones:

```
bindingSource1.Sort = "Nro";
```

```
bindingSource1.Sort = "Nro desc";
```

```
bindingSource1.Sort = "Nombre desc, Nro asc";
```

ADO.NET

■ Ejercitación

Agregue un
TextBox para
que el usuario

filtre la tabla Alumnos escribiendo un patrón que debe estar presente en cualquier parte del Nombre del alumno. Utilice el evento `TextChanged` del `TextBox`.

Utilice el evento `CurrentChanged` del `bindingSource1` para escribir el nombre del alumno corriente en el título de la ventana (las filas del `bindingSource` deben considerarse objetos `DataRowView`)

García, Nestor

TABLA ALUMNOS

Filtro: ar

	Nombre	Nro
	Gomez, María	1431
▶	García, Nestor	1022
*		

TABLA NOTAS

	Nro	Nota
▶	1022	7
	1022	5
*		

ADO.NET

■ Ejercitación

Windows application window titled "García, Nestor" showing two data tables.

TABLA ALUMNOS

Filtro:

	Nombre	Nro
	Gomez, María	1431
▶	García, Nestor	1022
*		

TABLA NOTAS

	Nro	Nota
▶	1022	7
	1022	5
*		

```
void TextBox1TextChanged(object sender, EventArgs e)
{
    bindingSource1.Filter="Nombre like '"+'*'+textBox1.Text+'*''";
}

void BindingSource1CurrentChanged(object sender, EventArgs e)
{
    this.Text=(bindingSource1.Current as DataRowView) ["Nombre"].ToString();
}
```

Leer y escribir datos XML

eXtensible Markup Language

Qué es XML?

```
<?xml version="1.0"?>  
<autores>  
  <autor ID="1">  
    <nombre>Perez</nombre>  
  </autor>  
  <!-- Este es un comentario. -->  
</autores>
```

Qué es XML?

- XML es un subconjunto de el Standard Generalized Markup Language (SGML), y está definido por el World Wide Web Consortium (W3C).
 - Proporciona un método uniforme para describir e intercambiar estructuras de datos
 - XML define sólo la estructura de los datos, no la forma de presentación
-

Qué es XML?

- HTML es un Lenguaje de Presentación

```
<span style="font-family:Arial">Date: 05/04/2005</span>  
<span style="font-family:Verdana">Subject: .NET</span>  
<span style="font-family:Arial">Speaker: Bill Gates</span>
```

- XML es un lenguaje descriptivo

```
<Event>  
  <Date>05/04/2005</Date>  
  <Subject>XML</Subject>  
  <Speaker>Bill Gates</Speaker>  
</Event>
```

Qué es XML?

- Proporciona un método uniforme para describir e intercambiar estructuras de datos
- Puede definir sus propios elementos y atributos
- Los elementos pueden estar anidados
- En un documento XML se encuentran:
 - ❑ Instrucciones de procesamiento
 - ❑ Elementos
 - ❑ Atributos
 - ❑ Comentarios

Qué es XML?

Instrucciones de procesamiento

```
<?xml version="1.0"?>
```

Atributos

```
<autores>
```

Elementos

```
<autor ID="1">
```

```
<nombre>Perez</nombre>
```

```
</autor>
```

Comentarios

```
<!-- Este es un comentario. -->
```

```
</autores>
```

Ventajas de XML

- XML es un estándar industrialmente aceptado.
 - Independiente de la aplicación
 - XML no es nada más que texto.
 - Es legible
-

XML

■ Elementos

- Un elemento usualmente consiste en un **tag de inicio** y un **tag de cierre**.
- El primer elemento que contiene a todos los otros elementos es llamado **elemento raíz**.
- Todos los elementos dentro del raíz, son llamados **elementos hijos**.
- Cualquier elemento hijo puede tener **elementos anidados**.

XML

■ Atributos

- ❑ Cualquier elemento puede contener atributos.
- ❑ Se utilizan para definir contenido dentro de los elementos.
- ❑ Se declara el nombre del atributo seguido del valor asignado en el tag de inicio.
- ❑ Se usa comillas simples o dobles para colocar el valor del atributo.

■ Comentarios

- ❑ Los comentarios son opcionales.

XML Bien formado

- Cumple con las especificaciones de las recomendaciones de W3C para XML 1.0.
- Contiene exactamente un elemento raíz (el elemento documento).
- Todos los elementos hijos son anidados apropiadamente unos con otros.
- El tag inicio y el tag fin de un elemento dado se encuentra dentro del cuerpo del mismo elemento padre.
- Un elemento vacío, por ejemplo `<P></P>` puede abreviarse como `<P/>`.

Ejemplo de un XML bien-formado:

```
<empleados>
  <empleado>
    <nombre>Stuart Munson</nombre>
    <cargo>Programmer</cargo>
  </empleado>
  <empleado>
    <nombre>Robert Brown</nombre>
    <cargo>Tester</cargo>
  </empleado>
</empleados>
```

Ejemplo de un XML mal-formado:

```
<empleados>
```

```
  <empleado>
```

```
    <nombre>Stuart Munson</nombre>
```

```
    <cargo>Programmer</cargo>
```

```
  <empleado>
```

```
    <nombre>Robert Brown</nombre>
```

```
    <cargo>Tester</cargo>
```

```
  </empleado>
```

```
</empleados>
```

XML Válidos (esquemas)

- Un XML es válido si cumple con el conjunto de requerimientos
 - Los DTDs son documentos que definen las etiquetas válidas dentro de un documento XML.
 - Los DTDs no son documentos XML en sí mismos, no son demasiado extensibles, no nos permite establecer validaciones complejas.
 - Los DTDs evolucionaron. Esta evolución son los schemas XML (XSD).
-

XML Válidos (esquemas)

- El principal aporte de XSD es el gran número de tipos de datos que incorpora.
- Soporta tipos de datos típicos de los lenguajes de programación, como también tipos personalizados simples y complejos.
- XSD se basa en Namespaces. Cada Namespace contiene elementos y atributos.

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Ejemplo de XSD

Se indica la versión de XML y la codificación que se usa (XSD es un XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema">
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título"/>
        <xsd:element name="Autor"/>
        <xsd:element name="Editorial"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="float"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Se indica al usuario la página que ofrece las pautas de creación de XSD en las que se basará la descripción del esquema.

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema" >
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

El elemento raíz se llama "Libro" y tiene tres hijos y un atributo.

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema" >
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Los hijos son “Título”, “Editorial” que deben aparecer una vez y “Autores” que puede aparecer de una a diez veces.

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema" >
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

La secuencia indica que los elementos deben aparecer en orden, es decir, primero el “Título”, luego los “Autores” y por último la “Editorial”

Ejemplo de XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema" >
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Los tres elementos son de tipo string. El atributo de libro se llama "precio" y es de tipo double.

Otro ejemplo

```
<xsd:element name="LastName" minOccurs="0"  
maxOccurs="*" type="string"> </xsd:element>
```

Otro ejemplo

Dentro del documento XML,
<LastName> puede ocurrir 0 o más
veces. El tipo de elemento es string:

```
<xsd:element name="LastName" minOccurs="0"  
maxOccurs="*" type="string"> </xsd:element>
```

XML y DataSet

- Con **ADO.NET** es muy sencillo convertir datos en formato **XML**. También es muy sencillo generar un esquema **XSD** para validar los documentos.
- Tanto **DataSet** como **DataTable** poseen métodos para trabajar con **XML**, entre ellos **ReadXml**, **WriteXml**, **ReadXmlSchema**, **WriteXmlSchema**

XML y DataSet

- **ReadXml**. Pone el **DataSet** con datos **XML** leídos desde un archivo.
- **WriteXml**. Escribe el contenido completo del **DataSet** en un archivo.
- **ReadXmlSchema**. Lee un **esquema XML** desde un archivo y configura el **DataSet** (por ejemplo creando los objetos **DataColumn**)
- **WriteXmlSchema**. Escribe en un archivo el **esquema XSD** describiendo el contenido del **DataSet**

XML y DataSet

Agregue este botón y codifique el manejador para el evento click:

```
void BotonGuardarClick(object sender, EventArgs e)
{
    ds.WriteXml("dataset.xml");
    MessageBox.Show("XML generado!");
}
```



XML y DataSet



```
<?xml version="1.0" standalone="yes"?>
<Mi_x0020_dataset>
  <Alumnos>
    <Nombre>Perez, Rodrigo</Nombre>
    <Nro>1024</Nro>
  </Alumnos>
  <Alumnos>
    <Nombre>García, Nestor</Nombre>
    <Nro>1022</Nro>
  </Alumnos>
  <Alumnos>
    <Nombre>Rodriguez, Sofía</Nombre>
    <Nro>1033</Nro>
  </Alumnos>
  <Alumnos>
    <Nombre>Gomez, María</Nombre>
    <Nro>1431</Nro>
  </Alumnos>
  <Notas>
    <Nro>1024</Nro>
    <Nota>8.7</Nota>
  </Notas>
  <Notas>
    <Nro>1024</Nro>
    <Nota>9.5</Nota>
  </Notas>
  <Notas>
    <Nro>1022</Nro>
    <Nota>7</Nota>
  </Notas>
  <Notas>
    <Nro>1022</Nro>
    <Nota>5</Nota>
  </Notas>
</Mi_x0020_dataset>
```

- Verifique el archivo generado. Búsquelo en la carpeta donde se generó el ejecutable.

Observe que las tablas se guardan una a continuación de la otra como un colección de elementos <Alumnos> y <Notas>

- Coloque la propiedad Nested de la relación en true y vuelva a generar el xml. Verifique el archivo generado.

```
ds.Relations["relacion"].Nested = true;
```

XML y DataSet

```
ds.Relations["relacion"].Nested = true;
```



```
dataset.xml - Bloc de notas
Archivo Edición Formato Ver Ayuda
<?xml version="1.0" standalone="yes"?>
<Mi_x0020_dataset>
  <Alumnos>
    <Nombre>Perez, Rodrigo</Nombre>
    <Nro>1024</Nro>
    <Notas>
      <Nro>1024</Nro>
      <Nota>8.7</Nota>
    </Notas>
    <Notas>
      <Nro>1024</Nro>
      <Nota>9.5</Nota>
    </Notas>
  </Alumnos>
  <Alumnos>
    <Nombre>García, Nestor</Nombre>
    <Nro>1022</Nro>
    <Notas>
      <Nro>1022</Nro>
      <Nota>7</Nota>
    </Notas>
    <Notas>
      <Nro>1022</Nro>
      <Nota>5</Nota>
    </Notas>
  </Alumnos>
  <Alumnos>
    <Nombre>Rodriguez, Sofía</Nombre>
    <Nro>1033</Nro>
  </Alumnos>
  <Alumnos>
    <Nombre>Gomez, María</Nombre>
    <Nro>1431</Nro>
  </Alumnos>
</Mi_x0020_dataset>
```

Ahora el xml refleja la relación maestro/detalle definida

XML y DataSet

The screenshot shows a Windows form titled "ADO". Inside the form, there are two main sections. The top section is titled "TABLA ALUMNOS" and contains a "Filtro" text box and a large gray rectangular area representing a data grid. Below this grid is a button labeled "Guardar como XML". The bottom section is titled "TABLA NOTAS" and also contains a large gray rectangular area representing a data grid. Below this grid is a button labeled "Cargar desde XML". At the bottom of the form, there are two binding sources: "bindingSource1" and "bindingSource2".

ADO

TABLA ALUMNOS

Filtro

Guardar como XML

TABLA NOTAS

dataGridView3

dataGridView4

Cargar desde XML

bindingSource1 bindingSource2

Agregue dos grilla, un
bindingSource y un
botón al formulario


XML y DataSet

- Codifique el manejador para el evento click del nuevo botón de la siguiente manera

```
void BotonCargarClick(object sender, EventArgs e)
{
    DataSet ds2=new DataSet();
    ds2.ReadXml("dataset.xml");
    bindingSource2.DataSource=ds2.Tables["Alumnos"];
    dataGridView3.DataSource = bindingSource2;
    dataGridView4.DataSource = ds2.Tables["Notas"];
    

---


    MessageBox.Show("Dataset \""+ds2.DataSetName+
                    "\" ha sido cargado!");
}
```



No podemos hacer referencia a la relación porque no existe en el DataSet ds2 (observar que no hay información de ella en el archivo xml.)

XML y DataSet

Perez, Rodrigo

TABLA ALUMNOS

Filtro

	Nombre	Nro
	Gomez, María	1431
	Rodriguez, Sofía	1033

¿Qué ocurre si intenta escribir algo que no sea un número?

Guardar como XML

	Nombre	Nro
▶	Perez, Rodrigo	1024
	García, Nestor	1022
	Rodriguez, Sofía	1033
	Gomez, María	1431
*		

Cargar desde XML

TABLA NOTAS

	Nro	Nota
▶	1024	8,7
	1024	9,5
*		

Observar que no hay relación maestro/detalle

	Nro	Nota
▶	1024	8.7
	1024	9.5
	1022	7
	1022	5
*		

XML y DataSet

- En el archivo “dataset.xml” no hay información sobre la relación maestro/detalle ni tampoco sobre el tipo de las columnas de las tablas, por ello no existe ningún impedimento para ingresar datos que no sean numéricos en las columnas “Nro” y “Nota”
- Para guardar la información sobre el tipo de las columnas y las relaciones entre tablas necesitamos guardar el **esquema** del dataset en formato **XSD**

XML y DataSet

Se guarda el esquema de las tablas en el mismo archivo xml

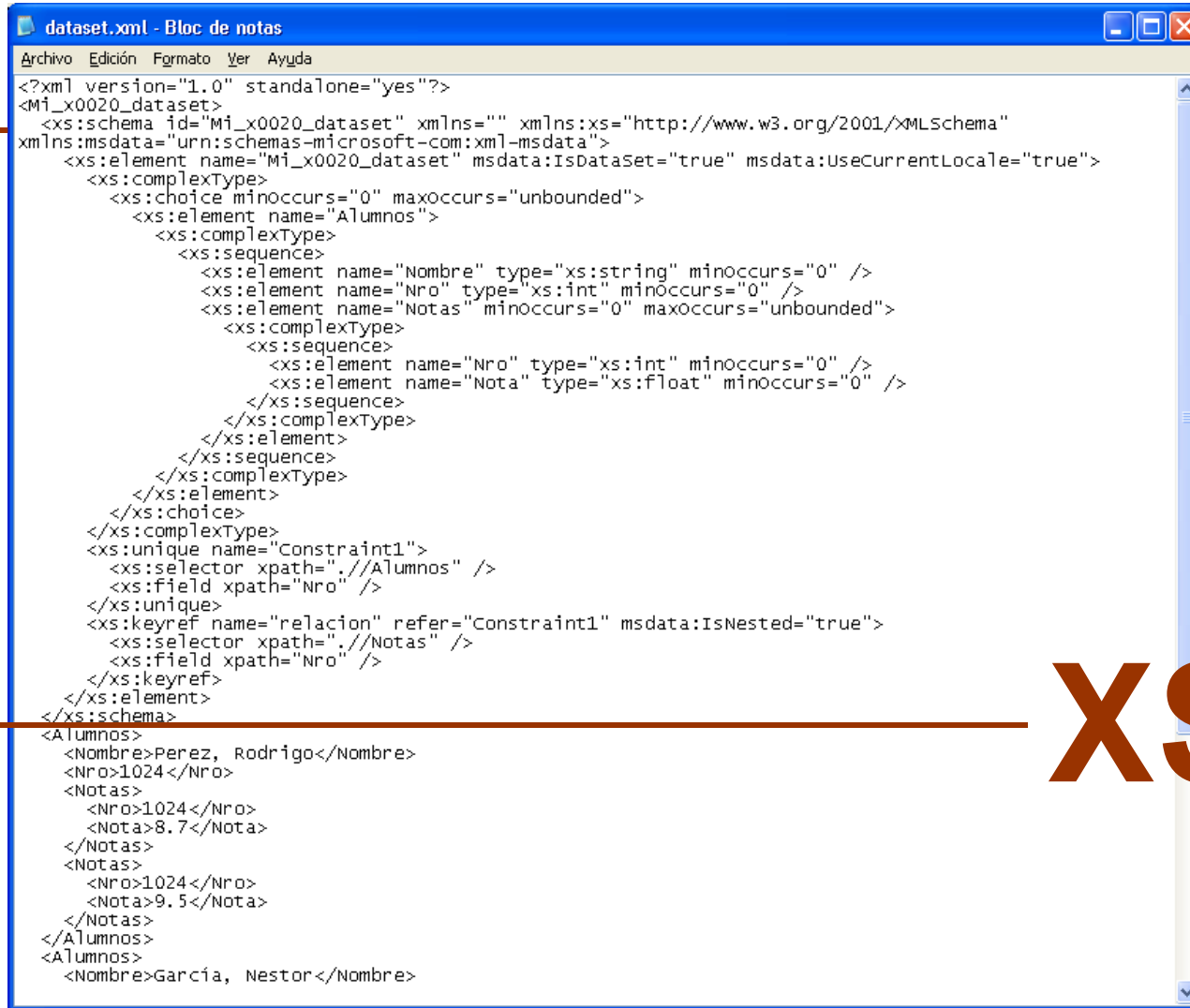
```
void BotonGuardarClick(object sender, EventArgs e)
{
    ds.WriteXml("dataset.xml", XmlWriteMode.WriteSchema);
    MessageBox.Show("XML generado!");
}
```

```
void BotonCargarClick(object sender, EventArgs e)
{
    DataSet ds2=new DataSet();
    ds2.ReadXml("dataset.xml", XmlReadMode.ReadSchema);
    bindingSource2.DataSource=ds2.Tables["Alumnos"];
    dataGridView3.DataSource = bindingSource2;
    dataGridView4.DataSource = bindingSource2;
    dataGridView4.DataMember = "relacion";
    MessageBox.Show("Dataset \""+ds2.DataSetName+
        "\" ha sido cargado! ");
}
```

Se leen los esquemas y se configuran las tablas de acuerdo a ellos

Se puede enlazar dataGridView4 con la relación

XML y DataSet



```
dataset.xml - Bloc de notas
Archivo Edición Formato Ver Ayuda
<?xml version="1.0" standalone="yes"?>
<Mi_x0020_dataset>
  <xs:schema id="Mi_x0020_dataset" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="Mi_x0020_dataset" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
      <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="Alumnos">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Nombre" type="xs:string" minOccurs="0" />
                <xs:element name="Nro" type="xs:int" minOccurs="0" />
                <xs:element name="Notas" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="Nro" type="xs:int" minOccurs="0" />
                      <xs:element name="Nota" type="xs:float" minOccurs="0" />
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
      <xs:unique name="Constraint1">
        <xs:selector xpath="." />
        <xs:field xpath="Nro" />
      </xs:unique>
      <xs:keyref name="relacion" refer="Constraint1" msdata:IsNested="true">
        <xs:selector xpath="." />
        <xs:field xpath="Nro" />
      </xs:keyref>
    </xs:element>
  </xs:schema>
  <Alumnos>
    <Nombre>Perez, Rodrigo</Nombre>
    <Nro>1024</Nro>
    <Notas>
      <Nro>1024</Nro>
      <Nota>8.7</Nota>
    </Notas>
  </Alumnos>
  <Alumnos>
    <Nombre>García, Nestor</Nombre>
```

XSD