

Programación Web .NET Core

Módulo 3

Acceso a datos con ADO.NET

Acceso a datos con ADO.NET

En esta etapa, implementaremos una aplicación que se conecta a un servidor de base de datos SQL Server para poder listar, agregar, borrar y modificar datos de una tabla. Se utilizará la tecnología **ADO.NET** para el acceso a Sql Server.

1. Crearemos en **SQL Server Management Studio** una base de datos llamada **Comercio** y dentro de ella una tabla llamada **Articulos** con la estructura que vemos en la imagen:

```
SQLQuery1.sql - D:\PHIA54K\amerc (62J) - 8 X
1  create database Comercio
2  go
3  use Comercio
4  go
5  create table Articulos(
6  Codigo int primary key,
7  Descripcion varchar(50) not null,
8  Precio float not null
9  )
10 go
```



2. Luego de crear la tabla, procederemos a insertar algunos registros y a verificar su existencia en la tabla:

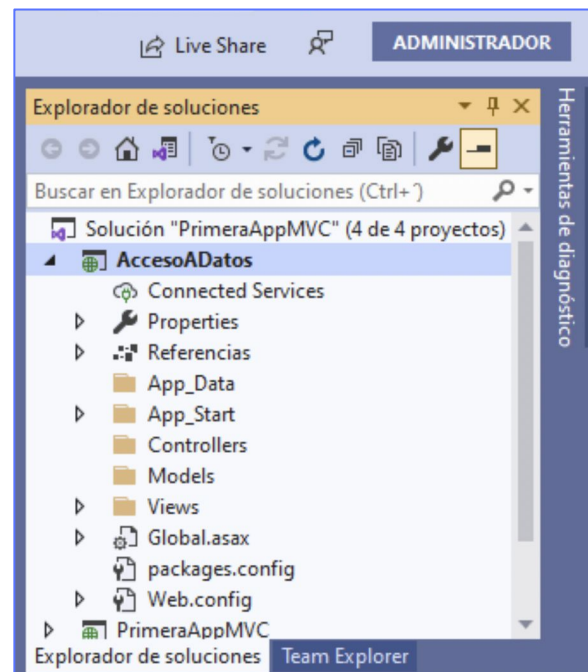
```
SQLQuery1.sql - D:\PHAS4K\amerc (62)
1 insert into Articulos values
2 (1, 'Placa de video GeForce 1650', 75000),
3 (2, 'Monitor Samsung Curve Led 19', 50000),
4 (3, 'Procesador Intel I5 10400', 25000)
5 go
6
7 select * from Articulos
```

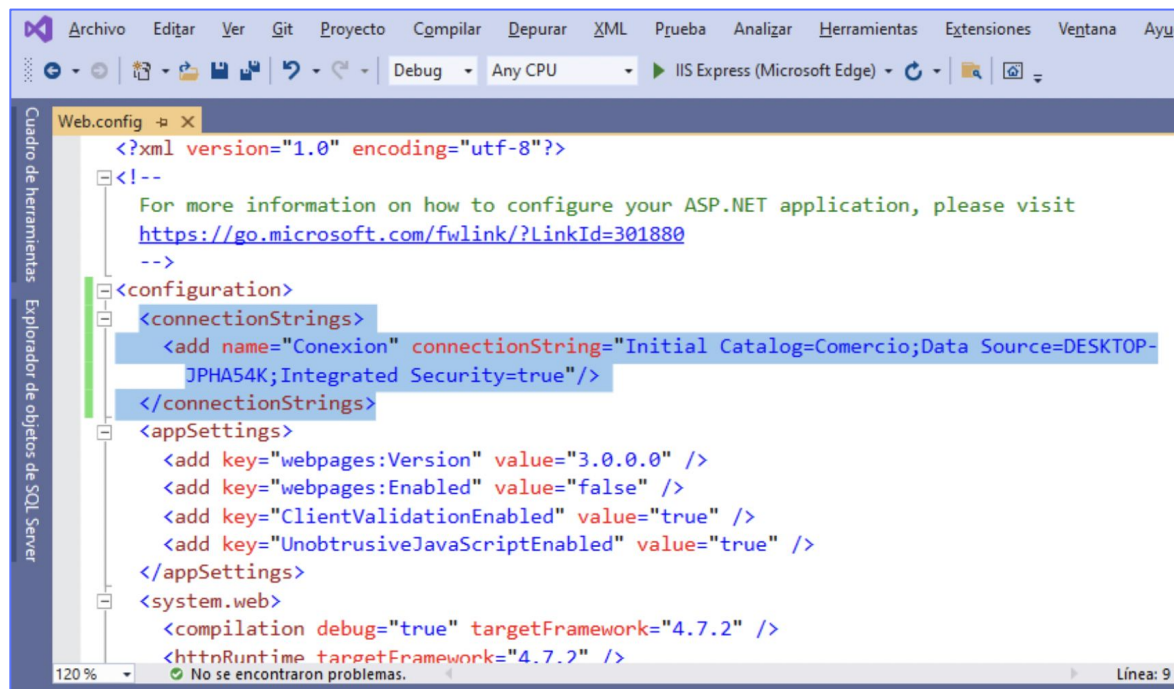
130 %

Results Messages

	Codigo	Descripcion	Precio
1	1	Placa de video GeForce 1650	75000
2	2	Monitor Samsung Curve Led 19	50000
3	3	Procesador Intel I5 10400	25000

3. A continuación, crearemos nuestro proyecto.
4. Una vez creado, modificaremos el archivo de configuración del sitio **web.config** agregando una entrada para la cadena de conexión a la base de datos. Una vez agregada la conexión, guardaremos este archivo. Como vemos en la imagen de la pantalla siguiente.





Visual Studio interface showing the Web.config file. The file content is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=301880
-->
<configuration>
  <connectionStrings>
    <add name="Conexion" connectionString="Initial Catalog=Comercio;Data Source=DESKTOP-
    JPHA54K;Integrated Security=true"/>
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
</configuration>
```

The status bar at the bottom indicates "No se encontraron problemas." (No problems found.) and "Línea: 9" (Line: 9).

5. A continuación, en la carpeta **Models** agregaremos dos clases de modelo, **Articulo** y **AdmArticulo**:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AccesoADatos.Models
{
    public class Articulo
    {
        public int Codigo { get; set; }
        public string Descripcion { get; set; }
        public float Precio { get; set; }
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace AccesoADatos.Models
{
    public class AdmArticulo
    {
        private SqlConnection conexion;

        private void Conectar()
        {
            string stringConexion = ConfigurationManager.ConnectionStrings["Conexion"].ToString();
            conexion = new SqlConnection(stringConexion);
        }
    }
}
```





```
public int Alta(Articulo pArticulo)
{
    Conectar();
    SqlCommand sentencia = new SqlCommand("insertintoArticulos(Codigo, Descripcion, Precio)
    values (@codigo, @descripcion, @precio)", conexion);

    sentencia.Parameters.Add("@codigo", SqlDbType.Int);
    sentencia.Parameters.Add("@descripcion", SqlDbType.VarChar);
    sentencia.Parameters.Add("@precio", SqlDbType.Float);

    sentencia.Parameters["@codigo"].Value = pArticulo.Codigo;
    sentencia.Parameters["@descripcion"].Value = pArticulo.Descripcion;
    sentencia.Parameters["@precio"].Value = pArticulo.Precio;

    conexion.Open();

    int i = sentencia.ExecuteNonQuery();

    conexion.Close();

    return i;
}
```



```
public List<Articulo> TraerTodos()
{
    Conectar();
    List<Articulo> articulos = new List<Articulo>();

    SqlCommand sentencia = new SqlCommand("select codigo, descripcion, precio from articulos",
        conexion);

    conexion.Open();

    SqlDataReader registros = sentencia.ExecuteReader();

    while (registros.Read())
    {
        Articulo articulo = new Articulo
        {
            Codigo = int.Parse(registros["codigo"].ToString()),
            Descripcion = registros["descripcion"].ToString(),
            Precio = float.Parse(registros["precio"].ToString())
        };

        articulos.Add(articulo);
    }
}
```

```
...  
  
conexion.Close();  
return articulos;  
}  
  
public Artículo TraerArticulo(int pCodigo)  
{  
    Conectar();  
  
    SqlCommand sentencia = new SqlCommand("select codigo, descripcion, precio  
    from articulos where codigo=@codigo", conexion);  
  
    sentencia.Parameters.Add("@codigo", SqlDbType.Int);  
    sentencia.Parameters["@codigo"].Value = pCodigo;  
  
    conexion.Open();  
  
    SqlDataReader registros = sentencia.ExecuteReader();  
  
    Artículo articulo = new Artículo();  
  
    if (registros.Read())  
    {  
        articulo.Codigo = int.Parse(registros["codigo"].ToString());  
        articulo.Descripcion = registros["descripcion"].ToString();  
        articulo.Precio = float.Parse(registros["precio"].ToString());  
    }  
}
```

```
...
conexion.Close();
return articulo;
}

public int Modificar(Articulo pArticulo)
{
    Conectar();

    SqlCommand sentencia = new SqlCommand("update articulos set
    descripcion=@descripcion, precio=@precio where codigo=@codigo", conexion);

    sentencia.Parameters.Add("@descripcion", SqlDbType.VarChar);
    sentencia.Parameters["@descripcion"].Value = pArticulo.Descripcion;

    sentencia.Parameters.Add("@precio", SqlDbType.Float);
    sentencia.Parameters["@precio"].Value = pArticulo.Precio;

    sentencia.Parameters.Add("@codigo", SqlDbType.Int);
    sentencia.Parameters["@codigo"].Value = pArticulo.Codigo;

    conexion.Open();

    int i = sentencia.ExecuteNonQuery();

    conexion.Close();

    return i;
}
```

...

```
public int Borrar(int pCodigo)
{
    Conectar();

    SqlCommand sentencia = new SqlCommand("delete from articulos where codigo=@codigo", conexion);
    sentencia.Parameters.Add("@codigo", SqlDbType.Int);
    sentencia.Parameters["@codigo"].Value = pCodigo;

    conexion.Open();

    int i = sentencia.ExecuteNonQuery();

    conexion.Close();

    return i;
}
}
```

- a. La clase **AdmArticulo** tiene un método que nos permite insertar una fila en la tabla llamada **Articulos** llamando al método **Alta**, al que le enviamos como parámetro un objeto de tipo **Articulo** llamado **pArticulo**.
 - b. Ejecutaremos al método **Conectar** cuyo objetivo es inicializar la variable **conexion** con la conexión a la base de datos **Comercio**.
 - c. Seguido a ello, creamos un objeto de tipo **SqlCommand** y pasamos como parámetro la sentencia SQL que nos permite insertar una fila en la tabla **Articulos**.
 - d. Definimos los tipos de datos y valores de los parámetros, y los agregamos a la colección de parámetros de la sentencia SQL.
 - e. Abrimos la conexión y pedimos a SQL Server que ejecute la sentencia correspondiente.
 - f. Por último, cerramos la conexión.
- Los métodos **Modificar**, **Borrar**, **TraerArticulo** y **TraerTodos** son similares, pero implementando la lógica correspondiente a esas operaciones.
- Ahora pasaremos a crear nuestro controlador, al que llamaremos **Home**.



Listado de artículos

Dentro del código de este controlador, implementaremos en la acción **Index** la lógica para poder mostrar todos los artículos en una página HTML.

1. Para poder trabajar con objetos de tipo **Articulo** deberemos agregar la referencia **using** a **Models**.
2. Creamos un objeto de la clase **AdmArticulo** y llamamos al método **TraerTodos()** y se lo pasamos como argumento a la vista:

```
public ActionResult Index()
{
    AdmArticulo objAdmArt = new AdmArticulo();
    return View(objAdmArt.TraerTodos());
}
```

3. Ya que la vista todavía no existe, la agregaremos. Se debe hacer clic con el botón derecho del mouse sobre **Index()** para seleccionar **Agregar Vista**. En la ventana, a continuación, en plantilla, seleccionamos **Empty**.
4. Para que se muestren todos los artículos, deberemos codificar en el archivo **Index.cshtml**.
5. En la vista con la siguiente sintaxis, indicamos el modelo, de la siguiente manera:

```
@model IEnumerable<AccesoADatos.Models.Articulo>
```

6. En el **div** principal del body crearemos la estructura de una tabla y mediante un **foreach** recorreremos la lista que llegó desde el controlador y se almacenó en la propiedad **Model1**:

```
<div>
<tableclass="table">
<tr>
<th>Codigo</th>
<th>Descripción</th>
<th>Precio</th>
<th>Borrar</th>
<th>Modificar</th>
</tr>
```


...

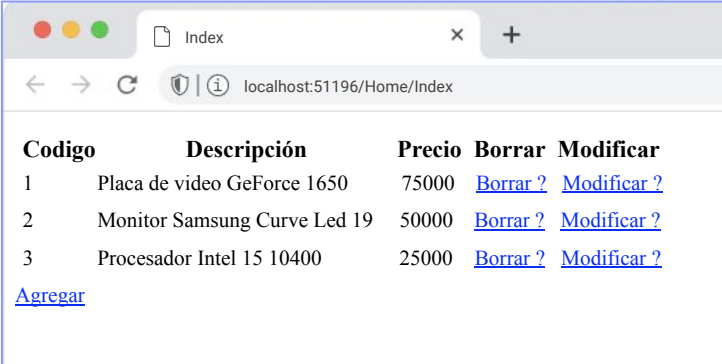
```
@foreach (var articulo in Model)
{
<tr>
<td>@articulo.Codigo</td>
<td>@articulo.Descripcion</td>
<td>@articulo.Precio</td>
<td><a href="/Home/Baja?cod=@articulo.Codigo">Borrar ?</a></td>
<td><a href="/Home/Modificacion?cod=@articulo.Codigo">Modificar ?</a></td>
</tr>
}

</table>
<a href="/Home/Alta">Agregar</a>
</div>
```



7. Si ejecutamos nuestra aplicación ya podemos ver el listado completo de artículos, como se muestra en la imagen de la derecha:

Debajo de la página se agrega un vínculo a la acción **Alta** del controlador **Home** (todavía se encuentra sin codificar).



A screenshot of a web browser window. The address bar shows 'localhost:51196/Home/Index'. The page displays a table with 5 columns: 'Codigo', 'Descripción', 'Precio', 'Borrar', and 'Modificar'. There are 3 rows of data representing computer components. Below the table is a link labeled 'Agregar'.

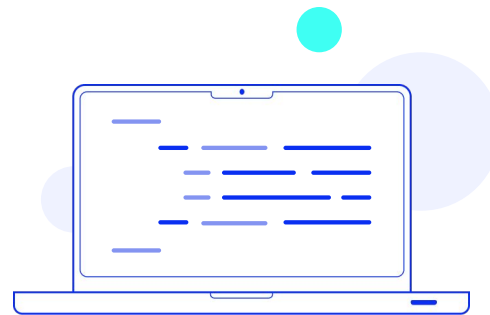
Codigo	Descripción	Precio	Borrar	Modificar
1	Placa de video GeForce 1650	75000	Borrar ?	Modificar ?
2	Monitor Samsung Curve Led 19	50000	Borrar ?	Modificar ?
3	Procesador Intel 15 10400	25000	Borrar ?	Modificar ?

[Agregar](#)

Alta de artículo

Con la página principal que muestra el contenido de la tabla **Articulos**, ahora implementaremos el alta de artículos.

1. Se deben definir dos acciones más en la clase **HomeController**. Pasemos a la pantalla siguiente para poder ver el detalle.



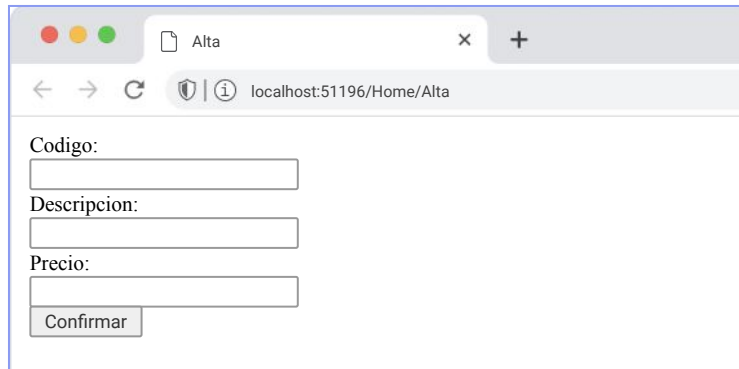
```
publicActionResultAlta()
{
    returnView();
}

[HttpPost]
publicActionResultAlta(FormCollectioncoleccion)
{
    AdmArticuloobjAdmArt = newAdmArticulo();
    Articulo articulo = new Articulo
    {
        Codigo = int.Parse(coleccion["codigo"]),
        Descripcion = coleccion["descripcion"],
        Precio = float.Parse(coleccion["precio"].ToString())
    };
    objAdmArt.Alta(articulo);
    returnRedirectToAction("Index");
}
```

- Si bien las dos acciones se llaman igual: **Alta**, a la segunda le agregamos la anotación cuyo nombre es [**HttpPost**].
 - La primera de las acciones **Alta()** tiene por objetivo mostrar la vista con un formulario HTML que permita ingresar y guardar los datos de un artículo. Cuando se presione el botón de tipo **submit** se ejecutará la otra acción **Alta** que recibe como parámetro los datos que se hayan cargados en el formulario.
2. Solamente a la primera acción **Alta** se le debe asociar una vista. Para esto presionaremos el botón derecho del mouse en **Alta()** y seleccionamos **Agregar Vista**.

```
<div>
<formmethod="post"action="/Home/Alta">
<label>Codigo:<br/>
<inputtype="text"name="codigo"/>
</label>
<br/>
<label>Descripcion:<br/>
<inputtype="text"name="descripcion"/>
</label>
<br/>
<label>Precio:<br/>
<inputtype="text"name="precio"/>
</label>
<br/>
<inputtype="submit"value="Confirmar"/>
</form>
</div>
```

3. Si ejecutamos se mostrará la siguiente página en el navegador, como se ve en la imagen:



The screenshot shows a web browser window with a single tab titled 'Alta'. The address bar displays 'localhost:51196/Home/Alta'. The page content consists of a form with three text input fields. The first field is labeled 'Codigo:', the second 'Descripcion:', and the third 'Precio:'. Below these fields is a button labeled 'Confirmar'.

- La página HTML muestra un formulario donde se deben ingresar el código, descripción y precio del artículo.
- Cuando se presiona el botón **Confirmar** se ejecuta la acción **Alta** del controlador **Home**.
- A este método llega un objeto de la clase **FormCollection** donde se almacenan todos los datos cargados en el formulario.
- No se asocia una vista a esta acción ya que mediante el método **RedirectToAction** se solicita la ejecución de otra acción, en éste caso la acción **Index** que lleva a mostrar el listado completo de artículos.

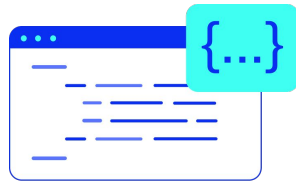
Baja de artículo

Implementaremos ahora el código para borrar un artículo cuando se presione el vínculo **Borrar ?**.

1. Deberemos codificar la siguiente acción llamada **Baja** en el controlador cuyo nombre es **HomeController**:

```
public ActionResult Baja(int paramCodigo)
{
    AdmArticulo objAdmArt = new AdmArticulo();
    objAdmArt.Borrar(paramCodigo);
    return RedirectToAction("Index");
}
```

2. Cuando desde la vista **Index** se haga clic en el vínculo para borrar un determinado artículo se ejecuta la acción **Baja** y llega como parámetro el código del artículo a borrar
3. Luego de borrarlo de la tabla, procederemos a llamar a la acción **Index**.



Modificación de artículo

Para la modificación de un artículo deberemos implementar dos acciones en el controlador.

1. El código a implementar, es que vemos a continuación:

```
public ActionResult Modificacion(int paramCodigo)
{
    AdmArticulo objAdmArt = new AdmArticulo();
    Articulo articulo = objAdmArt.TraerArticulo(paramCodigo);
    return View(articulo);
}
```



...

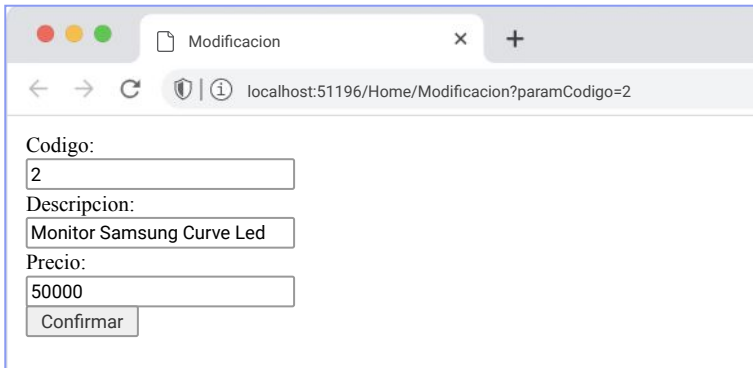
```
[HttpPost]
public ActionResult Modificacion(FormCollection coleccion) {
    AdmArticulo objAdmArt = new AdmArticulo();
    Articulo articulo = new Articulo
    {
        Codigo = int.Parse(coleccion["codigo"].ToString()),
        Descripcion = coleccion["descripcion"].ToString(),
        Precio = float.Parse(coleccion["precio"].ToString())
    };
    objAdmArt.Modificar(articulo);
    return RedirectToAction("Index");
}
```

2. Generaremos la vista para la acción:

```
<div>
<formmethod="post"action="/Home/Modificacion">
<label>Codigo:<br/>
<inputtype="text"name="codigo"value="@Model.Codigo"readonly/>
</label>
<br/>
<label>Descripcion:<br/>
<inputtype="text"name="descripcion"value="@Model.Descripcion"/>
</label>
<br/>
<label>Precio:<br/>
<inputtype="text"name="precio"value="@Model.Precio"/>
</label>
<br/>
<inputtype="submit"value="Confirmar"/>
</form>
</div>
```



Al ejecutar la aplicación y seleccionar el hipervínculo de **Modificar** ? se mostrará la página:



Codigo:
2

Descripcion:
Monitor Samsung Curve Led

Precio:
50000

Confirmar

- Como podemos ver, definimos el control **input** de HTML donde se ingresa el código de artículo de sólo lectura **readonly**.
- Al presionar el botón **Confirmar** se ejecuta la acción modificando los datos de la tabla.





Nota: En esta etapa implementamos una aproximación más cercana a la realidad del código que deberíamos desarrollar en cada una de las carpetas del patrón MVC. En ella, se encuentra separada la lógica de la aplicación en el **Modelo**, la presentación e ingreso de datos en la **Vista** y la comunicación en el **Controlador**. En la vista codificamos HTML y accedemos a los datos mediante Razor.

**¡Sigamos
trabajando!**

