



APP CRUD C#

Primero DAW/DAM

Descripción breve

Primera toma de contacto para realizar operaciones básicas sobre una base de datos SQL Server utilizando C# y sentencias SQL

Antº Javier Miras Llamas

Tabla de contenido

Base de datos	2
Clases para la gestión de la base de datos	5
Clase CConexionBD	5
Clase CProductosBD	6
Clase CCategoriasBD	13
Clase CMarcasBD	15
APP CRUD	17
Ventana principal	17
Ventana inserción/modificación	26
Ejercicios.....	32

Base de datos

La base de datos la vamos a crear en SQL Server. Se llamará ERP y tendrá las siguientes tablas:

Tabla	Campos	Tipo	Descripción
categorias	categoria_id	Entero (PK)	Clave primaria
	categoria	Cadena (32)	Nombre de la categoría
marcas	marca_id	Entero (PK)	Clave primaria
	marca	Cadena (32)	Nombre de la marca
productos	producto_id	Entero (PK)	Clave primaria
	categoria_id	Entero (FK)	Clave ajena a la tabla categorías
	marca_id	Entero (FK)	Clave ajena a la tabla marcas
	producto	Cadena (32)	Nombre del producto
	precio	Decimal (10,2)	Precio del producto

El esquema entidad/relación es de la siguiente forma:



Ejecutamos el siguiente script con las instrucciones SQL necesarias para la creación de la base de datos.

```
-- Creación de la base de datos

CREATE DATABASE ERP;
GO

USE ERP;
GO

-- Creación de las tablas

CREATE TABLE categorias(
    categoria_id INT IDENTITY (1,1) PRIMARY KEY,
    categoria NVARCHAR(32)
);
GO

CREATE TABLE marcas(
    marca_id INT IDENTITY (1,1) PRIMARY KEY,
    marca NVARCHAR(32)
);
GO

CREATE TABLE productos(
    producto_id INT IDENTITY (1,1) PRIMARY KEY,
    categoria_id INT,
    marca_id INT,
    producto NVARCHAR(32),
    precio DECIMAL(10,2)

    CONSTRAINT productos_categorias_fk FOREIGN KEY (categoria_id) REFERENCES categorias(categoria_id),
    CONSTRAINT productos_marcas_fk FOREIGN KEY (marca_id) REFERENCES marcas(marca_id)
);
GO

-- Inserción de datos
INSERT INTO categorias VALUES
    (''),
    ('Portátiles'),
    ('PCs escritorio'),
    ('Impresoras'),
```

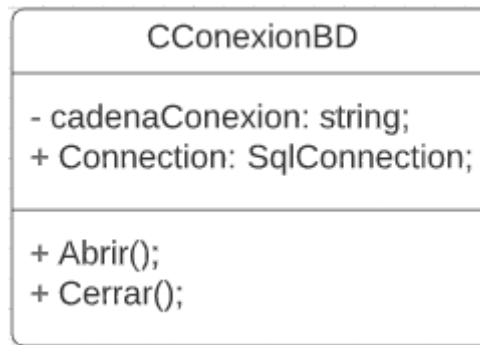
```
('Monitores'),  
( 'Teclados'),  
( 'Tarjetas video'),  
( 'Altavoces'),  
( 'Micrófonos');  
GO  
  
INSERT INTO marcas VALUES  
( ''),  
( 'Brother'),  
( 'HP'),  
( 'LG'),  
( 'Logitech'),  
( 'Lenovo'),  
( 'Asus'),  
( 'Dell'),  
( 'Sansumg'),  
( 'Gygabyte'),  
( 'Epson'),  
( 'Nvidia');  
GO
```

Clases para la gestión de la base de datos

Vamos a crear las clases para la gestión de la base de datos. La primera que crearemos se la clase para conectarnos a la misma. Después, crearemos tres clases más para la gestión de las tres tablas de nuestro proyecto.

Clase CConexionBD

La primera clase que crearemos será *CConexionBD* que es la que nos realizará la conexión a la base de datos.



El código de la clase se muestra a continuación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
```

```
namespace crud
{
    public class CConexionBD
    {
        // Cadena de conexión con la base de datos.
```

```
// En mi caso el servidor es PC-I5, debéis cambiarlo por vuestro servidor.
static private string cadenaConexion = @"Server=PC-I5\SQLEXPRESS;DataBase=ERP;Integrated Security=true;";

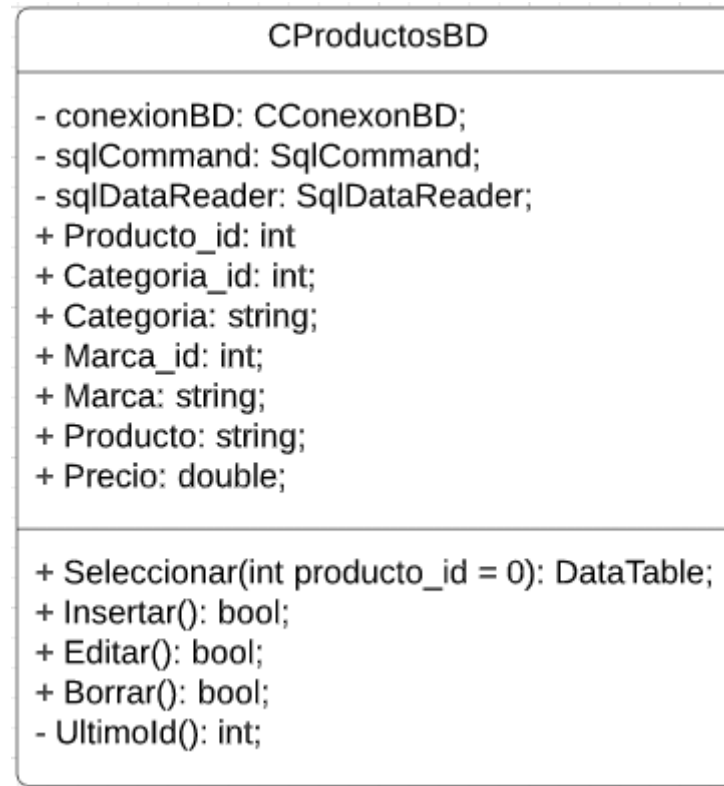
// Conexión a la base de datos.
public SqlConnection Connection { get; } = new SqlConnection(cadenaConexion);

public void Abrir()
{
    // Si la conexión está cerrada, la abrimos.
    if (Connection.State == ConnectionState.Closed)
        Connection.Open();
}

public void Cerrar()
{
    // Si la conexión está abierta, la cerramos.
    if (Connection.State == ConnectionState.Open)
        Connection.Close();
}
}
```

Clase CProductosBD

La clase *CProductosBD* se encargará de gestionar la tabla *productos* de nuestra base de datos. En ella realizaremos las cuatro operaciones: SELECT, INSERT, UPDATE y DELETE.



El código es el siguiente:

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace crud
{
    public class CProductosBD : CConexionBD
    {
        // Para realizar la conexión a la base de datos.
        private CConexionBD conexionBD = new CConexionBD();

        // Para ejecutar un procedimiento almacenado o realizar las sentencias SQL.
    }
}
```



```
private SqlCommand sqlCommand = new SqlCommand();

// Para almacenar los datos de una sentencia SELECT.
private SqlDataReader sqlDataReader;

// Propiedades para almacenar los datos de un registro de la tabla.
public int Producto_id { get; set; }
public int Categoria_id { get; set; }
public String Categoria { get; set; }
public int Marca_id { get; set; }
public String Marca { get; set; }
public String Producto { get; set; }
public double Precio { get; set; }

public DataTable Seleccionar(int producto_id = 0)
{
    // Para almacenar la tabla leída en memoria.
    DataTable dataTable = new DataTable();

    try
    {
        // Realizamos la conexión.
        conexionBD.Abrir();

        // Y se la asignamos al comando SQL.
        sqlCommand.Connection = conexionBD.Connection;

        // Indicamos el tipo de comando. En este caso una sentencia SQL.
        sqlCommand.CommandType = CommandType.Text;

        // Si me han pedido todos los productos.
        if (producto_id == 0)
        {
            sqlCommand.CommandText =
                "SELECT producto_id AS Id, producto AS Producto, categorias.categoria AS Categoría," +
                " marcas.marca AS Marca, precio AS Precio" +
                " FROM productos" +
                " INNER JOIN categorias ON productos.categoria_id = categorias.categoria_id" +
                " INNER JOIN marcas ON productos.marca_id = marcas.marca_id" +
                " ORDER BY producto, categoria, marca";
        }
        else
        {

```

```
// En caso contrario un producto en concreto.
sqlCommand.CommandText =
    "SELECT producto_id AS Id, producto AS Producto, productos.categoria_id, " +
    " categorias.categoria AS Categoría, productos.marca_id, marcas.marca AS Marca, ," +
    " precio AS Precio" +
    " FROM productos" +
    " INNER JOIN categorias ON productos.categoria_id = categorias.categoria_id" +
    " INNER JOIN marcas ON productos.marca_id = marcas.marca_id" +
    " WHERE producto_id=" + producto_id;
}

// Ejecutamos la sentencia.
sqlDataReader = sqlCommand.ExecuteReader();

// Guardamos el resultado en memoria.
dataTable.Load(sqlDataReader);

// Si me indicaron que seleccionase un único registro, y este existe.
if ((producto_id != 0) && (dataTable.Rows.Count != 0))
{
    // Obtenemos las filas de la tabla en memoria (En este sólo hay una única fila).
    DataRow[] rows = dataTable.Select();

    // Asignamos a cada propiedad el valor del registro leído.
    Producto_id = producto_id;
    Producto = rows[0]["producto"].ToString();
    Categoria_id = Convert.ToInt32(rows[0]["categoria_id"].ToString());
    Categoría = rows[0]["categoría"].ToString();
    Marca_id = Convert.ToInt32(rows[0]["marca_id"].ToString());
    Marca = rows[0]["marca"].ToString();
    Precio = Convert.ToDouble(rows[0]["precio"].ToString());
}
}
finally
{
    // Cerramos los datos leídos.
    sqlDataReader.Close();

    // Cerramos la conexión.
    conexionBD.Cerrar();
}

// Devolvemos la tabla almacenada en memoria.
```

```
        return dataTable;
    }

    public bool Insertar()
    {
        // Para devolver si la operación se hizo correctamente, o no.
        bool bInsertada = false;

        try
        {
            // Es similar a la selección, salvo la sentencia SQL.
            conexionBD.Abrir();

            sqlCommand.Connection = conexionBD.Connection;

            sqlCommand.CommandType = CommandType.Text;

            // Observar que hemos utilizado Format para construir la sentencia.
            // El producto se ha puesto entre comillas ('{2}'), porque es una cadena.
            // En el Precio hay que cambiar la coma (,) por un punto (.) para poder guardarlo en la tabla.
            sqlCommand.CommandText =
                string.Format("INSERT INTO productos VALUES ({0}, {1}, '{2}', {3})",
                    Categoria_id, Marca_id, Producto, Convert.ToString(Precio).Replace(",", "."));

            // Ejecutamos la sentencia, indicando que no es una consulta SELECT, y
            // aprovechamos el número de registros que nos devuelve. En este caso debe ser 1.
            bInsertada = sqlCommand.ExecuteNonQuery() == 1;

            // Si la inserción fue correcta, obtenemos el valor de la clave primaria.
            if (bInsertada)
            {
                Producto_id = UltimoId();
            }
        }
        finally
        {
            conexionBD.Cerrar();
        }

        // Devolvemos si la operación fue correcta o no.
        return bInsertada;
    }
}
```

```
public bool Editar()
{
    bool bEditada = false;

    try
    {
        conexionBD.Abrir();

        sqlCommand.Connection = conexionBD.Connection;

        sqlCommand.CommandType = CommandType.Text;

        // Recordad que el producto, al ser una cadena, va entre comillas ('{2}').
        // De nuevo hay que tener cuidado con el Precio, hay que cambiar la coma (,) por un punto (.) para poder
        // guardarlo en la tabla.
        sqlCommand.CommandText =
            string.Format("UPDATE productos SET categoria_id={0}, marca_id={1}, producto='{2}', precio={3}" +
                " WHERE producto_id={4}",
                Categoria_id, Marca_id, Producto, Convert.ToString(Precio).Replace(",", "."),
                Producto_id);

        bEditada = sqlCommand.ExecuteNonQuery() == 1;
    }
    finally
    {
        conexionBD.Cerrar();
    }

    return bEditada;
}

public bool Borrar()
{
    bool bBorrada = false;

    try
    {
        conexionBD.Abrir();

        sqlCommand.Connection = conexionBD.Connection;

        sqlCommand.CommandText = "DELETE productos WHERE producto_id=" + Producto_id;
```

```
        sqlCommand.CommandType = CommandType.Text;

        bBorrada = sqlCommand.ExecuteNonQuery() == 1;
    }
    finally
    {
        conexionBD.Cerrar();
    }

    return bBorrada;
}

private int UltimoId()
{
    int ultimo_id = 0;

    try
    {
        // Esta sentencia obtiene el último producto insertado.
        sqlCommand.CommandText = "SELECT @@IDENTITY as ultimo_id";

        sqlDataReader = sqlCommand.ExecuteReader();

        DataTable dataTable = new DataTable();

        dataTable.Load(sqlDataReader);

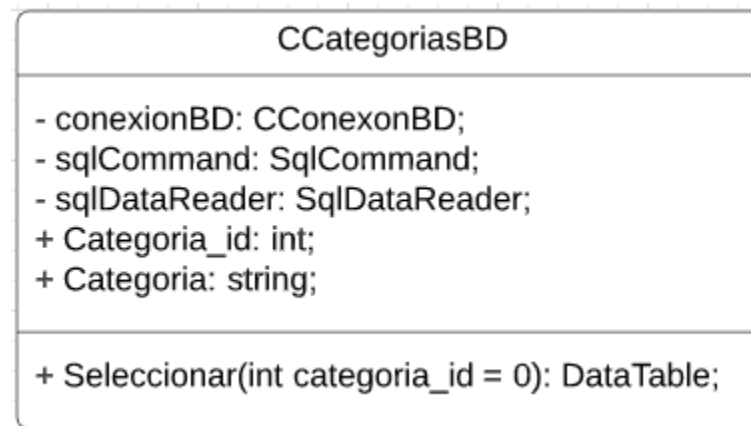
        DataRow[] rows = dataTable.Select();

        // Obtenemos la clave primaria del último producto insertado.
        ultimo_id = Convert.ToInt32(rows[0]["ultimo_id"].ToString());
    }
    finally
    {
        sqlDataReader.Close();
    }

    return ultimo_id;
}
}
```

Clase CategoriasBD

La clase *CCategoriasBD* se encargará de gestionar la tabla *categorias* de nuestra base de datos. En ella realizamos sólo la operación: SELECT.



El código es el siguiente:

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace crud
{
    class CategoriasBD
    {
        private CConexionBD conexionBD = new CConexionBD();
        private SqlCommand sqlCommand = new SqlCommand();
        private SqlDataReader sqlDataReader;

        public int Categoria_id { get; set; }
        public String Categoria { get; set; }

        public DataTable Seleccionar(int categoria_id = 0)
```

```
{
    DataTable dataTable = new DataTable();
    try
    {
        conexionBD.Abrir();

        sqlCommand.Connection = conexionBD.Connection;

        sqlCommand.CommandType = CommandType.Text;

        // Si me indicaron que tenía que obtener todas las categorías, ...
        if (categoria_id == 0)
            sqlCommand.CommandText = "SELECT * FROM categorias ORDER BY categoria ASC";
        else
            // En caso contrario solo una categoría.
            sqlCommand.CommandText = "SELECT * FROM categorias WHERE categoria_id=" + categoria_id;

        sqlDataReader = sqlCommand.ExecuteReader();

        dataTable.Load(sqlDataReader);

        if ((categoria_id != 0) && (dataTable.Rows.Count != 0))
        {
            DataRow[] rows = dataTable.Select();

            Categoria_id = categoria_id;
            Categoria = rows[0]["categoria"].ToString();
        }
    }
    finally
    {
        sqlCommand.Parameters.Clear();

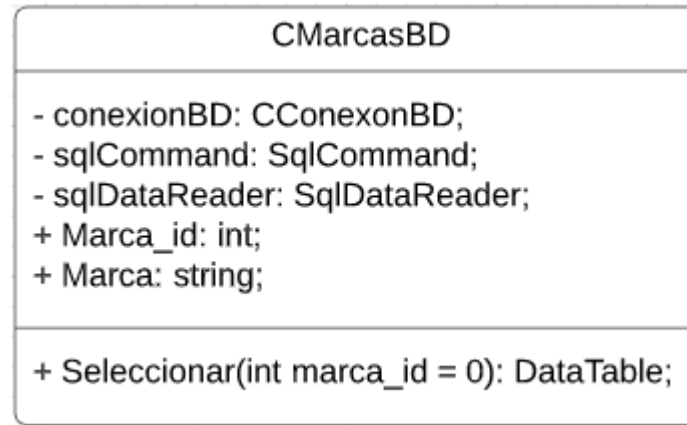
        sqlDataReader.Close();

        conexionBD.Cerrar();
    }

    return dataTable;
}
}
```

Clase CMarcasBD

La clase *CMarcasBD* se encargará de gestionar la tabla *marcas* de nuestra base de datos. En ella realizamos sólo la operación: SELECT.



El código es el siguiente:

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace crud
{
    class CMarcasBD
    {
        private CConexionBD conexionBD = new CConexionBD();
        private SqlCommand sqlCommand = new SqlCommand();
        private SqlDataReader sqlDataReader;

        public int Marca_id { get; set; }
        public String Marca { get; set; }

        public DataTable Seleccionar(int marca_id = 0)
        {
```



```
DataTable dataTable = new DataTable();
try
{
    conexionBD.Abrir();

    sqlCommand.Connection = conexionBD.Connection;

    sqlCommand.CommandType = CommandType.Text;

    // Si me indicaron que tenía que obtener todas las marcas, ...
    if (categoria_id == 0)
        sqlCommand.CommandText = "SELECT * FROM marcas ORDER BY marca ASC";
    else
        // En caso contrario solo una categoría.
        sqlCommand.CommandText = "SELECT * FROM marcas WHERE marca_id=" + marca_id;

    sqlDataReader = sqlCommand.ExecuteReader();

    dataTable.Load(sqlDataReader);

    if ((categoria_id != 0) && (dataTable.Rows.Count != 0))
    {
        DataRow[] rows = dataTable.Select();

        Marca_id = categoria_id;
        Marca = rows[0]["marca"].ToString();
    }
}
finally
{
    sqlCommand.Parameters.Clear();

    sqlDataReader.Close();

    conexionBD.Cerrar();
}

return dataTable;
}
```

APP CRUD

Ventana principal

Para gestionar nuestro CRUD crearemos una aplicación de Windows Form, a la que llamaremos *crud*.

La ventana principal tendrá las siguientes propiedades:

Componente	Propiedad	Valor
Form	(Name)	FPrincipal
	ControlBox	false
	MaximizeBox	false
	MinimizeBox	false
	MinimunSize	600;480
	StartPosition	CenterScreen
	Text	CRUD C#

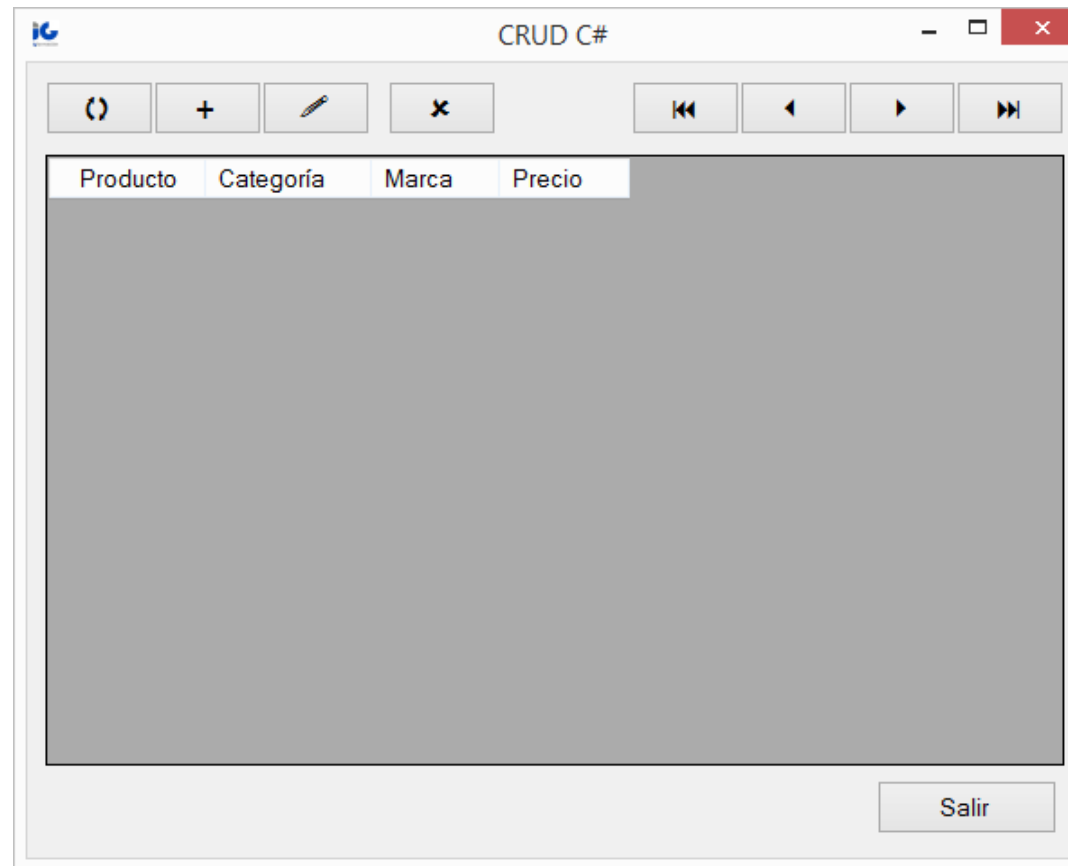
Le añadimos los siguientes componentes:

Componente	Propiedad	Valor
DataGridView	(Name)	dataGridView
	Anchor	Top, Bottom, Left, Right
	MultiSelect	False
	AutoSizeColumnsMode	AllCells
	MultiSelect	false

	ReadOnly	True
	RowHeadersVisible	false
	SelectionMode	FullRowSelect
Button	(Name)	btnRecargar
	Cursor	Hand
	Font	Webdings
	Text	q
Button	(Name)	btnNuevo
	Cursor	Hand
	Text	+
Button	(Name)	btnEditar
	Cursor	Hand
	Font	Webdings
	Text	!
Button	(Name)	btnBorrar
	Cursor	Hand
	Font	Webdings
	Text	û
Button	(Name)	btnPrimero
	Anchor	Top, Right
	Cursor	Hand
	Font	Webdings
	Text	9
Button	(Name)	btnAnterior

	Anchor	Top, Right
	Cursor	Hand
	Font	Webdings
	Text	3
Button	(Name)	btnSiguiente
	Anchor	Top, Right
	Cursor	Hand
	Font	Webdings
	Text	4
Button	(Name)	btnUltimo
	Anchor	Top, Right
	Cursor	Hand
	Font	Webdings
	Text	:
Button	(Name)	btnSalir
	Anchor	Bottom, Right
	Cursor	Hand
	Text	Salir

La app tendrá un aspecto similar al que se muestra a continuación.



El código asociado a la ventana principal se indica a continuación.

```
using System;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace crud
{
    public partial class FPrincipal : Form
    {
```

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

private void btnNuevo_Click(object sender, EventArgs e)
{
    // Instanciamos la clase FProductosModificar para introducir los datos.
    FProductosModificar fProductosModificar = new FProductosModificar();

    // Mostramos el cuadro de diálogo.
    fProductosModificar.ShowDialog();

    // Si se ha pulsado el botón de aceptar.
    if (fProductosModificar.DialogResult == DialogResult.OK)
    {
        // Recargamos la tabla.
        Recargar();

        // Obtenemos la clave primaria del producto insertado.
        int producto_id = fProductosModificar.Producto_id;

        // Buscamos la fila del producto insertado.
        int rowIndex = dataGridView.Rows
            .Cast<DataGridViewRow>()
            .Where(r => r.Cells[0].Value.Equals(producto_id))
            .First()
            .Index;

        // Nos posicionamos en ella.
        dataGridView.CurrentCell = dataGridView[1, rowIndex];
    }
}

private void btnEditar_Click(object sender, EventArgs e)
{
    // Si tenemos registros en la tabla.
    if (dataGridView.RowCount > 0)
    {
        // Obtenemos la clave primaria del producto.
        int producto_id = Convert.ToInt32(dataGridView.CurrentRow.Cells[0].Value);

        // Instanciamos la clase FProductosModificar para modificar los datos.
```

```
// Observar que le pasamos en el constructor la clave primaria.
FProductosModificar fProductosModificar = new FProductosModificar(producto_id);

// Mostramos el cuadro de diálogo.
fProductosModificar.ShowDialog();

// Si se ha pulsado el botón de aceptar.
if (fProductosModificar.DialogResult == DialogResult.OK)
{
    // Recargamos la tabla.
    Recargar();

    // Buscamos la fila del producto editado.
    int rowIndex = dataGridView.Rows
        .Cast<DataGridViewRow>()
        .Where(r => r.Cells[0].Value.Equals(producto_id))
        .First()
        .Index;

    // Nos posicionamos en ella.
    dataGridView.CurrentCell = dataGridView[1, rowIndex];
}
}

private void btnBorrar_Click(object sender, EventArgs e)
{
    // Si tenemos registros en la tabla y...
    // el usuario me confirma que realmente quiere borrar el registro.
    if ((dataGridView.RowCount > 0) &&
        (MessageBox.Show("¿Realmente quiere borrar el producto seleccionado?",
            "Confirmación",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Question) == DialogResult.Yes))
    {
        // Creamos una instancia de la clase CProductosBD.
        CProductosBD productosBD = new CProductosBD();

        // Obtenemos la clave principal del producto a borrar.
        productosBD.Producto_id = Convert.ToInt32(dataGridView.CurrentRow.Cells[0].Value);

        // Si el producto se borra correctamente.
        if (productosBD.Borrar())
```

```
{
    // Obtenemos la fila actual.
    int rowIndex = dataGridView.CurrentRow.Index;

    // Recargamos y vamos a la fila actual, que corresponderá al siguiente producto.
    Recargar(rowIndex);
}
else
    // Sino se ha podido borrar, mensaje de error.
    MessageBox.Show("Al borrar el producto.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private void btnRecargar_Click(object sender, EventArgs e)
{
    // Miramos en qué fila nos encontramos.
    // Si no tenemos filas, nos posicionamos en la primera (0).
    // En caso contrario, en la fila actual del DataGridView.
    // Observar la utilidad, en este caso, del operador ternario. Más limpio que utilizar un if.
    int rowIndex = (dataGridView.RowCount == 0) ? 0 : dataGridView.CurrentRow.Index;

    // Otra forma de mirar en qué fila nos encontramos sería con un if.
    //
    // int rowIndex = 0;
    //
    // if (dataGridView.RowCount > 0)
    //     rowIndex = dataGridView.CurrentRow.Index;

    // Llamamos al procedimiento recargar y nos posicionamos en la fila actual.
    Recargar(rowIndex);
}

private void Recargar(int rowIndex = 0)
{
    // Instanciamos la clase CProductosBD.
    CProductosBD productosBD = new CProductosBD();

    // Recargamos el DataGridView asociando el DataSource con los datos devueltos.
    dataGridView.DataSource = productosBD.Seleccionar();

    // Si tenemos datos...
    if (dataGridView.RowCount > 0)
    {
```



```
// Comprobamos que la fila que nos indican no es superior a la cantidad de filas que tenemos.
// Si es así, nos posicionamos en la última fila.
if (rowIndex >= dataGridView.RowCount)
    rowIndex = dataGridView.RowCount - 1;

// Si nos indican una fila negativa, nos posicionamos en la primera.
if (rowIndex < 0)
    rowIndex = 0;

// Nos posicionamos en la fila indicada.
dataGridView.CurrentCell = dataGridView[1, rowIndex];
}
}

private void FPrincipal_Load(object sender, EventArgs e)
{
    // Cargamos la tabla de productos.
    Recargar();

    // No permitimos que nos inserten filas a través del DataGridView.
    dataGridView.AllowUserToAddRows = false;

    // Las filas de la cabecera las ponemos centradas.
    dataGridView.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

    // Ocultamos la columna que muestra la clave primaria "id"
    dataGridView.Columns["id"].Visible = false;

    // La columna con los precios la mostramos formateada como moneda (currency)...
    dataGridView.Columns["Precio"].DefaultCellStyle.Format = "c";

    // y la alineamos a la derecha.
    dataGridView.Columns["Precio"].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

    // Si hay algún valor null, lo mostraremos con tres guiones.
    dataGridView.DefaultCellStyle.NullValue = "---";
}

private void btnSalir_Click(object sender, EventArgs e)
{
    // Si el usuarios realmente quiere salir, cerramos la app.
    if (MessageBox.Show("¿Realmente quiere salir de la App?",
        "Confirmación",
```

```
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes)
    {
        Close();
    }
}

private void btnPrimero_Click(object sender, EventArgs e)
{
    // Nos posicionamos en la primera fila del DataGridView.
    dataGridView.CurrentCell = dataGridView[1, 0];
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    // Buscamos la fila anterior.
    int rowIndex = dataGridView.CurrentRow.Index - 1;

    // Si es negativa, es porque estábamos ya en la primera fila.
    if (rowIndex < 0)
        rowIndex = 0;

    // Nos posicionamos en la fila del DataGridView.
    dataGridView.CurrentCell = dataGridView[1, rowIndex];
}

private void btnSiguiente_Click(object sender, EventArgs e)
{
    // Buscamos la fila siguiente.
    int rowIndex = dataGridView.CurrentRow.Index + 1;

    // Si es mayor que la cantidad de filas que hay en el DataGridView, entonces nos vamos a la última fila.
    if (rowIndex >= dataGridView.RowCount)
        rowIndex = dataGridView.RowCount - 1;

    // Nos posicionamos en la fila del DataGridView.
    dataGridView.CurrentCell = dataGridView[1, rowIndex];
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    // Buscamos la última fila.
    int rowIndex = dataGridView.RowCount - 1;
```

```
// Si no había filas en el DataGridView, entonces la fila será la primera.  
if (rowIndex < 0)  
    rowIndex = 0;  
  
// Nos posicionamos en la fila del DataGridView.  
dataGridView.CurrentCell = dataGridView[1, rowIndex];  
}  
}  
}
```

Ventana inserción/modificación

A continuación, añadiremos una nueva ventana con las siguientes propiedades:

Componente	Propiedad	Valor
Form	(Name)	FProductosModificar
	FormBorderStyle	FixedDialog
	StartPosition	CenterScreen
	Text	Productos :: Nuevo

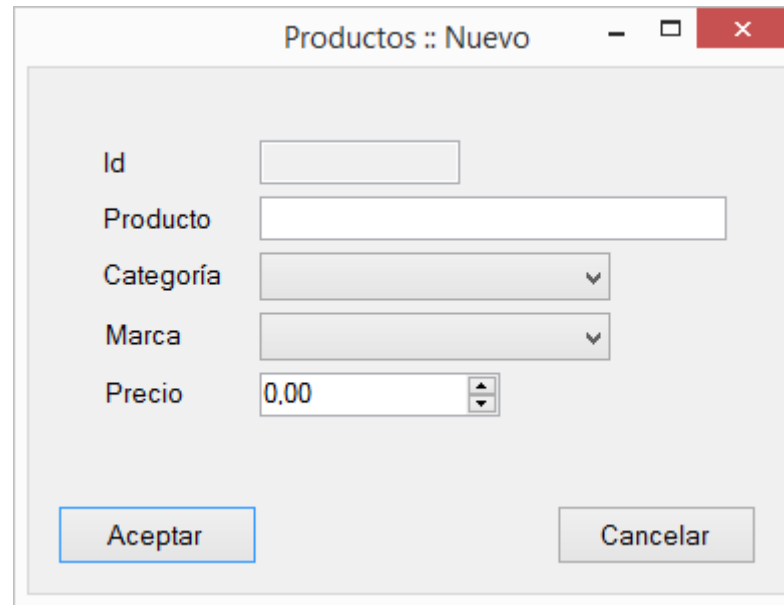
Al formulario le añadiremos los siguientes componentes:

Componente	Propiedad	Valor
Button	(Name)	btnAceptar
	Cursor	Hand
	TabStop	False
	Text	Aceptar
Button	(Name)	btnCancelar

	Cursor	Hand
	DialogResult	Cancel
	TabStop	False
	Text	Cancelar
TextBox	(Name)	txtId
	ReadOnly	true
	TabStop	False
Label	(Name)	lblId
	Text	Id
TextBox	(Name)	txtProducto
	TabIndex	0
Label	(Name)	lblProducto
	Text	Producto
ComboBox	(Name)	cbCategorias
	DropDownStyle	DropDownList
	TabIndex	1
Label	(Name)	lblCategorias
	Text	Categorías
ComboBox	(Name)	cbMarcas
	DropDownStyle	DropDownList
	TabIndex	2
Label	(Name)	lblMarcas
	Text	Marcas
NumericUpDown	(Name)	txtPrecio

	DecimalPlaces	2
	Increment	0,05
	Maximum	1000000
	TabIndex	3
	ThousandsSeparator	true

El aspecto del formulario será el siguiente.



El código asociado a la ventana de inserción/modificación de un producto es el siguiente.

```
using System;
using System.Windows.Forms;

namespace crud
{
```

```
public partial class FProductosModificar : Form
{
    public int Producto_id { get; set; }

    public FProductosModificar(int producto_id = 0)
    {
        InitializeComponent();

        // Clave primaria del producto indicado.
        Producto_id = producto_id;
    }

    private void FProductosModificar_Load(object sender, EventArgs e)
    {
        // Instanciamos las clases CCategoriasBD y CMarcasBD.
        CCategoriasBD categoriasBD = new CCategoriasBD();
        CMarcasBD marcasBD = new CMarcasBD();

        // Obtenemos todos los registros de la tabla.
        cbCategorias.DataSource = categoriasBD.Seleccionar();

        // Mostramos el valor del campo categoría.
        cbCategorias.DisplayMember = "categoria";

        // Indicamos que el valor seleccionado es la clave primaria.
        cbCategorias.ValueMember = "categoria_id";

        // Para las marcas hacemos lo mismo que para las categorías.
        cbMarcas.DataSource = marcasBD.Seleccionar();
        cbMarcas.DisplayMember = "marca";
        cbMarcas.ValueMember = "marca_id";

        // Si me indican un producto en concreto, es que queremos modificarlo.
        if (Producto_id != 0)
        {
            // Instanciamos la clase CProductosBD.
            CProductosBD productosBD = new CProductosBD();

            // Buscamos el producto.
            productosBD.Seleccionar(Producto_id);

            // Mostramos la clave primaria.
            txtId.Text = Convert.ToString(productosBD.Producto_id);
        }
    }
}
```

```
// El nombre del producto.
txtProducto.Text = productosBD.Producto;

// Buscamos en el ComboBox el índice de la categoría seleccionada.
cbCategorias.SelectedIndex = cbCategorias.FindStringExact(productosBD.Categoria);
// cbCategorias.SelectedValue = productosBD.Categoria_id;

// Otra forma de asignar el índice.
// cbMarcas.SelectedIndex = cbMarcas.FindStringExact(productosBD.Marca);
cbMarcas.SelectedValue = productosBD.Marca_id;

// Y finalmente, el precio.
txtPrecio.Value = Convert.ToDecimal(productosBD.Precio);

// Indicamos que estamos modificando.
Text = "Productos :: Modificar";
}
}

private void btnAceptar_Click(object sender, EventArgs e)
{
    // Verificamos que todo es correcto antes de proseguir.
    if (!Correcto())
        return;

    // Por defecto, indicamos que se pulsa el botón OK.
    DialogResult = DialogResult.OK;

    // Instanciamos la clase CProductodBD.
    CProductosBD productosBD = new CProductosBD();

    // Le pasamos a cada una de las propiedades los valores correspondientes.
    productosBD.Producto = txtProducto.Text;
    productosBD.Categoria_id = (int)cbCategorias.SelectedValue;
    productosBD.Marca_id = (int)cbMarcas.SelectedValue;
    productosBD.Precio = Convert.ToDouble(txtPrecio.Value);

    // Si estamos insertando...
    if (Producto_id == 0)
    {
        // Insertamos y verificamos que todo ha ido bien.
        if (productosBD.Insertar())
```

```
{
    Producto_id = productosBD.Producto_id;
}
else
{
    MessageBox.Show("Al insertar el producto.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

    // Si no se ha podido insertar, devolvemos Cancel.
    DialogResult = DialogResult.Cancel;
}
}
else
{
    // y sino, estamos modificando.

    // Indicamos el producto a modificar.
    productosBD.Producto_id = Producto_id;

    // Verificamos que si ha habido un error.
    if (!productosBD.Editar())
    {
        MessageBox.Show("Al modificar el producto.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

        // Si no se ha podido modificar, devolvemos Cancel.
        DialogResult = DialogResult.Cancel;
    }
}
}

private bool Correcto()
{
    if (txtProducto.Text == "")
    {
        MessageBox.Show("Debe indicar el nombre del producto", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtProducto.Focus();

        return false;
    }

    return true;
}
}
```


Ejercicios

1. Añadir a los productos un código. Este código es numérico de tipo entero, y debe ser único.
2. Se tiene que completar la clase *CCategoriasBD* con las operaciones: INSERT, UPDATE y DELETE.
3. Añadir a las categorías el campo código. Este campo es numérico de tipo entero, y debe ser único.
4. Se tiene que completar la clase *CMarcasBD* con las operaciones: INSERT, UPDATE y DELETE.
5. Añadir a las marcas el campo código. Este campo es numérico de tipo entero, y debe ser único.