**1.-** El método Buscar() de AccesoDAO tiene implementado un código reducido, que se utilizará para reducir el código de los métodos Insertar, Borrar, BorradoVirtual, Modificar, Obtener

**2.-** El código se puede mejorar tal y como se indica en los comentarios del código.

public bool BorradoVirtual(object objeto)

{

bool borrado = false;

StartTransaction();

string sql;

// Ver si se puede mejorar sin necesidad de recorrer todas las propiedades del objeto

// Tener en cuenta, que el campo con el nombre Borrado lo tendrán todas las tablas.

foreach (var item in objeto.GetType().GetProperties())

{

if (item.Name.Contains("Borra"))

{

item.SetValue(objeto, "1");

}

**3.-** Ver Tlibro y comprobar cómo genera automáticamente el código de la clave.

**4.-** Comprobar si este código debe ser elimando. **Es utilizado en UtilSql**.

private static string obtenerClave(Type clase)

{

foreach (var item in ObtenerNombrePropiedades(clase))

{

if (item.StartsWith("Cod"))

{

return item;

}

}

return null;

**5.-** Es muy probable que el Dictionary SENTENCIAS no funcione bien. Es posible que no grabe en el fichero las sentencias en dicho MAP.

**6.-** Comprobar si la instrucción if (item.Borrado.Equals("0")) es necesaria. En mi opinión no es necesaria porque cuando hacemos la búsqueda se hace solo para los registros donde el Campo Borrado = 0, por lo que nunca pueden llevar registros con un estado diferente. COMPROBAD el razonamiento.

private void ObtenerTodosLibros()

{

try

{

List<object> libros = new List<object>();

foreach (TLibro item in control.Obtener(new TLibro().GetType()))

{

if (item.Borrado.Equals("0"))

{

lstLibros.Items.Add(item);

}

}

7.- El formulario frmLibro internamente tiene como nombre clase Libro. **OJO**, esto hay que modificarlo a frmLibro puesto que llevaría a error dejándolo como clase Libro. LO MEJOR es crear el formulario desde el principio con los nombres correctos o REFACTORIZAR sabiendo qué estamos haciendo.

8.- En la clase UtilSql tenemos el método ObtenerClave() que nos devuelve el campo clave de la clase que se ha enviado por parámetro. La aplicación utiliza Reflexión para sacarla de la clase, utilizando el métedo ObtenerNombrePropiedades que también se encuentra en UtilSql.

Si consideramos que todas las claves de las tablas tienen que comenzar con el prefijo **Cod**, no necesitaríamos realizar la reflexión para cargar una lista con las propiedades de la clase y luego por comparación escoger la que contenga el literal **Cod**. La forma de montar la clave sería, añadir al literal **Cod**, el nombre de la tabla sin el prefijo **T** del pojo . Ejemplo **Cod+Libro. Este método de generación de clave, hace depender la arquitectura de la nomenclatura de los campos . Para que esta arquitectura tenga sentido y sea genérica, los campos de las tablas deben llamarse igual que los del Pojo y las tablas de la base de datos tendrán su correspondiente DTO con el mismo nombre, de esta manera podemos hacer el mapeo de los datos.**

**Podíamos haber tomado otro camino, tal y como lo hicimos en las versiones JAVA, utilizando Anotaciones . Consistiría en poner una anotación dentro de la clase, justo encima del campo que nos interesa . En nuestro caso en CodLibro. De esta manera mediante reflexión podemos interrogar a la clase y cuando encuentre esa anotación se coge el atributo asociado, en nuestro caso CodLibro, de esta manera hacemos la arquitectura más genérica y dependiendo menos de la estructura de la tabla y del nombre de los campos del DTO.**

**Comprobar si en posteriores versiones de la arquitectura en C#, en algún momento se cambió la forma de buscar el código de la tabla.**

**El método obtenerClave() de UtilSql se gestiona de una forma en 1-LibreriaV5.1-Inicio y de otra en 1-LibreriaV5.1-Base.**

**Conviene con WinMerge cruzar las dos aplicaciones anteriores**

**9.-** En AltaClick() comprobad si podríamos eliminar la búsqueda del libro, puesto que se ha obtenido una nueva instancia de libro junto a su clave generada automáticamente. Estaríamos intentando buscar algo que con toda seguridad no existe.

**10.-** En los métodos ModificarClick() y BajasClick() como los objetos se encuentran en la lista junto a su código, no es necesario comprobar si existe ese dato en la tabla. En un desarrollo web o Cliente/ Servidor sería conveniente hacer la búsqueda, por si en algún momento alguien ha borrado ese registro de la tabla. Nosotros no entraremos en ese detalle. Nuestra lista la cargamos con la parte del objeto que nos interesa.