

---

# TAREA 4

---

CAPACITACIÓN C#

📧 Ezequiel Remus  
ezequielremus@gmail.com

27 de marzo de 2023

## ENUNCIADO

**Keywords** First keyword · Second keyword · More

### 1. Tarea Debug

#### 1.1. Ejecutar la aplicacion alumnos

##### 1.1.1. Enunciado

Ejecutar la aplicación alumnos con todas las líneas sin comentar del archivo **Program**, la misma debería ejecutarse sin problemas y mostrar por consola la ejecución de todos los métodos finalizando con una lista de alumnos en la DB. Si modificaron el connection string en la clase “AccesoDatos”, puede que no muestre datos si sus tablas locales están vacías. Enviar captura de pantalla de la consola.

##### 1.1.2. Solución

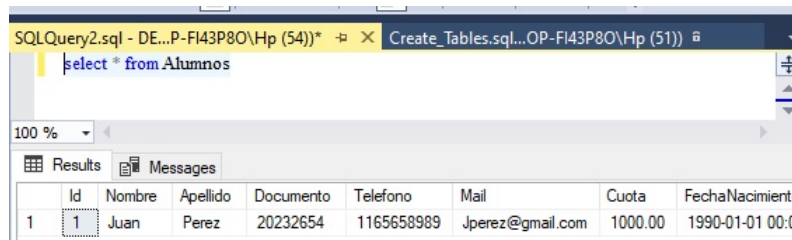
Entonces, la tarea consiste en compilar la aplicación de **AppAlumnos**. En este caso, yo hice la prueba conectando la App a mi *localhost* en **sql server**. Para esto modifique las líneas de código en la clase *AccesoDatos*

```
/// <summary>
/// Datos de el servidor de Datos.
/// </summary>
2 referencias
public AccesoDatos()
{
    /// Para la base de datos en la nube
    //serverBD = "SQL5110.site4now.net";
    //usuarioBD = "db_a96920_alumnos_admin";
    //PasswordBD = "Bootcamp2023";
    //basedatos = "db_a96920_alumnos";
    ///armo conexion a la BD en la nube
    //strcondatos = @"Data Source=" + serverBD + ";Initial Catalog=" + basedatos + ";User ID=" + usuarioBD + ";Password=" + PasswordBD;

    // Para la base de datos local
    //Nombre del servidor
    serverBD = "DESKTOP-FI43P80\\SQLEXPRESS01";
    // Base de datos a la que accede
    basedatos = "Alumnos_Clase";
    // Conexion local usuario windows
    strcondatos = @"Data Source=" + this.serverBD + ";Initial Catalog=" + this.basedatos + ";Integrated Security=true";
}
```

Figura 1: Modifico Convección A Base de Datos Local

Luego, Agregue un alumno a la base de datos desde **sql Server** con el código de *insert* pasado en clase

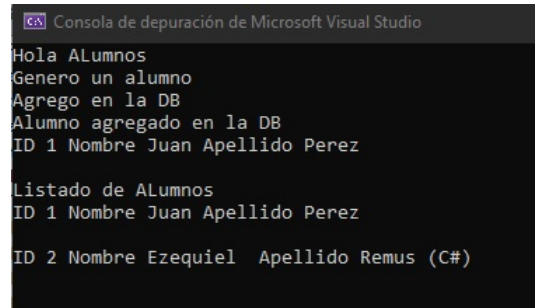


Id	Nombre	Apellido	Documento	Telefono	Mail	Cuota	FechaNacimiento
1	Juan	Perez	20232654	1165658989	jperez@gmail.com	1000.00	1990-01-01 00:00:00

Figura 2: Alumno Agregado en la Base de Datos

Corremos la AppAlumnos realizando un Debug. Cabe aclarar que modifique los datos a agregar del alumno en el `Programs.cs` pasandole los míos.

La salida por consola en este caso resulta Donde se ve que el programa lee los datos desde la base de datos a la que



```

C:\> Consola de depuración de Microsoft Visual Studio

Hola ALumnos
Genero un alumno
Agrego en la DB
Alumno agregado en la DB
ID 1 Nombre Juan Apellido Perez

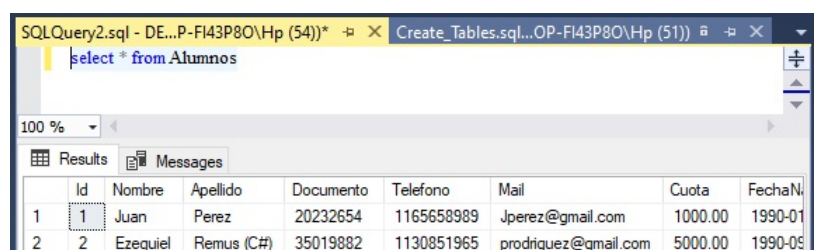
Listado de ALumnos
ID 1 Nombre Juan Apellido Perez

ID 2 Nombre Ezequiel Apellido Remus (C#)
  
```

Figura 3: Salida Final por consola

accede. Además, esto lo hace despues de haber agregado a un nuevo alumno (el que pase con mis datos).

Por ultimo, reviso que se haya realizado la escritura sobre la base de datos



Id	Nombre	Apellido	Documento	Telefono	Mail	Cuota	FechaNacimiento
1	Juan	Perez	20232654	1165658989	jperez@gmail.com	1000.00	1990-01-01 00:00:00
2	Ezequiel	Remus (C#)	35019882	1130851965	prodriguez@gmail.com	5000.00	1990-05-01 00:00:00

Figura 4: Base de datos luego de correr el programa

## 1.2. Ejecutar la aplicación AppAlumnos, agregando un breakpoint...

### 1.2.1. Enunciado

Ejecutar la aplicación AppAlumnos, agregando un breakpoint en la primer linea de código. Debugear linea por linea para ir verificando el estado de la misma.

### 1.2.2. Solución

Esto no era más que agregar un *brakePoint* (Punto de interrupción) al `Program.cs` y empezar a ejecutar Y como

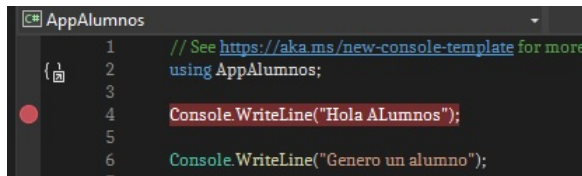


Figura 5: Agrego el brakePoint

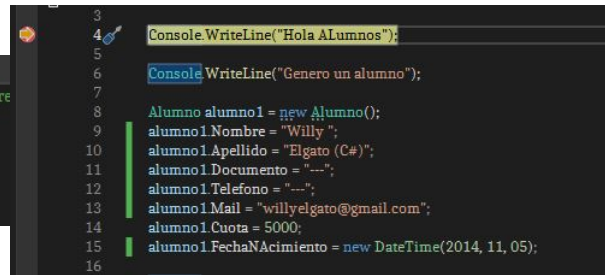


Figura 6: Ejecuto desde el brakePoint

cuando empezamos a hacer el *debug* la ejecución se frena en el brakePoint, en la consola no nos aparecera nada Luego,



Figura 7: Salida primer brake

al seguir realizando el debug nos va a aparecer una flecha indicando donde esta frenada en ese momento la compilación y cual sera la siguiente linea de código que se ejecutara Además, en consola podemos ver que líneas corrieron

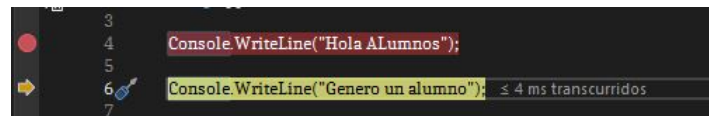


Figura 8: El debug fue avanzando

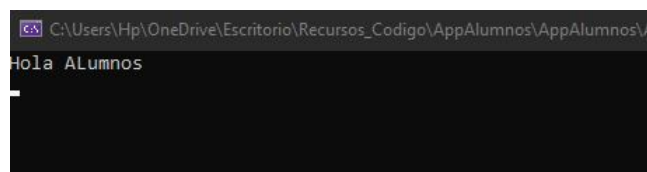


Figura 9: Salida de la primer lectura

### 1.3. En la linea 54 poner un brakpoint

#### 1.3.1. Enunciado

En la linea 54 poner un brakpoint, ejecutar el programa hasta esa linea. Enviar un print con los datos de la variable `.alumnoObtenidoDesdeLaDB` que se muestra en la ventana "Variables Locales." "Locals" del visual estudio

### 1.3.2. Solución

Otra vez, agregamos un breakpoint en la línea indicada. En mi caso es la 53 (Ya que en la 54 el visual no me lo permitía por no existir una sentencia en la cual frenar) Al frenar en la línea se tiene para el valor de la variable indicada lo

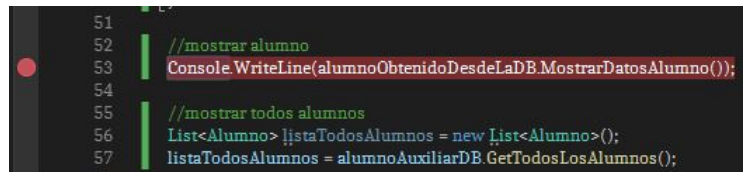


Figura 10: Pongo el brake

siguiente

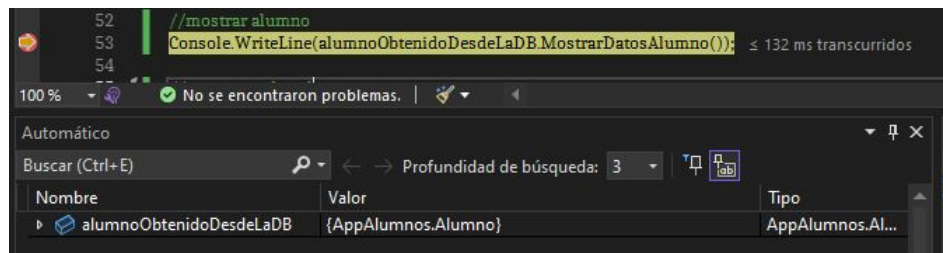


Figura 11: Ejecuto hasta el brake + valor de la variable

## 2. Tarea Bases de Datos

### 2.1. Agregar en la clase Profesor el metodo AgregarEnDb()

#### 2.1.1. Enunciado

Agregar en la clase Profesor el método “AgregarEnDb()”, que agregue haga un insert en la tabla “Profesores” Instanciar un profesor con sus datos y agregarlo en la base de datos Enviar Captura de pantalla del código, la consola y la tabla donde se vea el profesor agregado

#### 2.1.2. Solución

Lo primero que debemos hacer para poder agregar un profesor es tener acceso a la base de datos. Para esto, debemos importar en la clase Profesor la clase AccesoDatos y instanciar una variable de AccesoDatos

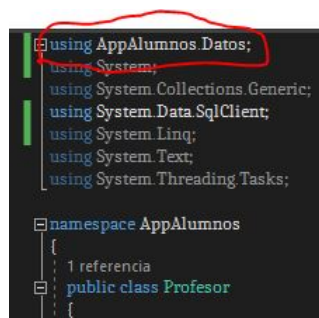


Figura 12: Agrego el using para tener acceso desde profesor

```
private AccesosDatos ac = new AccesosDatos();
```

Figura 13: Otorgo Colección al profesor

Ahora, utilizando el metodo AgregarEnDb() de la clase Alumno modifiko el insert segun los datos del profesor. También agrego avisos en porcentajes de subida (solo por jugar). Instanciamos ahora un profesor y utilizamos el metodo para

```
1 referencia
public int AgregarEnDb()
{
    Console.WriteLine("Agregando en la DB\n\t\t0%");
    string query = $"insert into profesores (Nombre, Apellido, Documento, Telefono, Mail, Sueldo, FechaNacimiento)";
    Console.WriteLine(".....\t\t10%");
    query += $"values(' {this.Nombre}', '{this.Apellido}', '{this.Documento}', '{this.Telefono}', " +
        $" '{this.Mail}', '{this.Sueldo}', '{this.FechaNacimiento.ToString("yyyyMMdd")})' ";
    Console.WriteLine(".....\t\t30%");
    try
    {
        Console.WriteLine(".....\t\t40%");
        SqlCommand command = new SqlCommand(query);
        Console.WriteLine(".....\t\t60%");
        int r = ac.ejecQueryDevuelveInt(command);
        Console.WriteLine(".....\t\t80%");
        return r;
    }
    catch (Exception ex)
    {
        //Log.Escribir(0, "ERROR", "Ocurrio un error en ChangeAddPartidaToProduct actualizando que guarde partida a producto " + ex.Message);
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}
```

Figura 14: Otorgo Colección al profesor

poder agregar a la base de datos

```
Profesor profe1 = new Profesor {
    Nombre = "Rodrigo",
    Apellido = "Moreiras",
    Documento = "20232654",
    Telefono = "1165658989",
    Mail = "rodrigo.moreiras22@gmail.com",
    Sueldo = 1000,
    FechaNacimiento = new DateTime(1990, 01, 01),
};

int resultado = profe1.AgregarEnDb();
if (resultado != 1)
{
    Console.WriteLine("No se pudo agregar Profesor en DB");
    return;
}
Console.WriteLine("Profesor agregado en la DB");
```

Figura 15: Instancio

Veamos si Compila!

```

Consola de depuración de Microsoft Visual Studio
Hola Profesores
Genero un alumno
Agregando en la DB
0%
..... 10%
..... 30%
..... 40%
..... 60%
..... 80%
Profesor agregado en la DB
  
```

Figura 16: Salida por consola

Ahora, la pregunta del millón... ¿Se agrego a la base de datos?

SQLQuery3.sql - DE...P-FI43P8O\Hp (54)) \* X Create\_Tables.sql...OP-FI43P8O\Hp (51)) a

select \* from Alumnos  
select \* from Profesores

100 %

Results Messages

	Id	Nombre	Apellido	Documento	Telefono	Mail	Sueldo	FechaNacimiento
1	1	Rodrigo	Moreiras	20232654	1165658989	rodrigo.moreiras22@gmail.com	1000.00	1990-01-01 00:00:00.000

Figura 17: Salida por consola

Claro que si!

## 2.2. Agregar en la clase Profesor el metodo ActualizarEnDb()

### 2.2.1. Enunciado

Agregar en la clase Profesor el método **ActualizarEnDb()**, que reciba un profesor como parámetro y modifique el mismo en la tabla *Profesores*

Al profesor agregado en el punto anterior, modificarle el teléfono utilizando este método (por defecto el profesor no va a tener ID asignado, asignárselo en el código para poder ejecutar el código)

Enviar Captura de pantalla del código, la consola y la tabla donde se vea el profesor antes de la modificación y luego con el teléfono modificado.

### 2.2.2. Solución

```

1 referencia
public int ActualizarEnDb(Profesor profesorAAActualizar)
{
    string query = $"update profesores set Nombre = '{profesorAAActualizar.Nombre}', " +
        $"Apellido='{profesorAAActualizar.Apellido}', " +
        $"Documento='{profesorAAActualizar.Documento}', " +
        $"Telefono = '{profesorAAActualizar.Telefono}', " +
        $"Mail = '{profesorAAActualizar.Mail}', " +
        $"Sueldo = {profesorAAActualizar.Sueldo}, " +
        $"FechaNacimiento = '{profesorAAActualizar.FechaNacimiento.ToString("yyyyMMdd")}' " +
        $"where id = {profesorAAActualizar.Id}";

    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 18: Método ActualizarEnDb

Results		Messages						
	Id	Nombre	Apellido	Documento	Telefono	Mail	Sueldo	FechaNacimiento
1	1	Carmen	Núñez	20232654	1122334455	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000
2	10	(pepe)	Núñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1990-01-01 00:00:00.000
3	12	Rodrigo	Moreiras	20232654	1165658989	rodrigo.moreiras22@gmail.com	1000.00	1990-01-01 00:00:00.000
4	13	Carmen	Núñez	20232654	1122334455	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000

Figura 19: Base de datos Antes de Actualizar

```

87 // Para Actualizar los datos de un Profesor
88 Profesor prof1 = new Profesor
89 {
90     Id = 13,
91     Nombre = "Carmen",
92     Apellido = "Núñez",
93     Documento = "20232654",
94     Telefono = "1165658989",
95     Mail = "carmen@iafe.uba.ar",
96     Sueldo = 1000,
97     FechaNacimiento = new DateTime(1960, 01, 01),
98 };
99
100 Profesor profesorAuxiliarDB = new Profesor();
101 int resultado = profesorAuxiliarDB.ActualizarEnDb(profe1);
102
103 if (resultado != 1)
104 {
105     Console.WriteLine("No se pudo actualizar alumno en DB");
106     return;
107 }

```

Figura 20: Archivo Programs a ejecutar



Results		Messages						
	Id	Nombre	Apellido	Documento	Telefono	Mail	Sueldo	FechaNacimiento
1	1	Carmen	Nuñez	20232654	1122334455	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000
2	10	(pepe)	Nuñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1990-01-01 00:00:00.000
3	12	Rodrigo	Moreiras	20232654	1165658989	rodrigo.moreiras22@gmail.com	1000.00	1990-01-01 00:00:00.000
4	13	Carmen	Nuñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000

Figura 21: Base de datos después de modificado el Teléfono

## 2.3. Agregar en la clase profesor el método EliminarEnDb()

### 2.3.1. Enunciado

Agregar en la clase profesor el método **EliminarEnDb()** que reciba como parámetro el Id del profesor a eliminar y lo elimine de la base de datos Instanciar un profesor con sus datos y agregarlo en la base de datos Agregarle Manualmente el Id con el que se agrego en la DB. Eliminarlo usando el metodo creado

### 2.3.2. Solución

```

/// <summary>
/// Elimina los datos de un alumno en la base de datos
/// </summary>
/// <param name="idProfesorEliminar"></param>
/// <returns>
/// 1 => si salio todo bien
/// -1 => si salio todo mal
/// </returns>
1 referencia
public int EliminarEnDb(int idProfesorEliminar)
{
    Console.WriteLine("Buscando al profe a borrar...");
    string query = $"delete profesores where id={idProfesorEliminar}";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        Console.WriteLine("Profesor Eliminado");
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 22: Metodo EliminarEnDb()

Results		Messages						
	Id	Nombre	Apellido	Documento	Telefono	Mail	Sueldo	FechaNacimiento
1	1	Carmen	Nuñez	20232654	1122334455	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000
2	10	(pepe)	Nuñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1990-01-01 00:00:00.000
3	12	Rodrigo	Moreiras	20232654	1165658989	rodrigo.moreiras22@gmail.com	1000.00	1990-01-01 00:00:00.000
4	13	Carmen	Nuñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000

Figura 23: Base de datos antes de Eliminar



```

/// Elimino profesor segun Id
Console.WriteLine("Elimino al Profesor");
Console.WriteLine("Agregue el id del profe a Eliminar");
int id = Convert.ToInt32(Console.ReadLine());

Profesor profesorAuxiliarDB = new Profesor();
int resultado2 = profesorAuxiliarDB.EliminarEnDb(id);
if (resultado2 != 1)
{
    Console.WriteLine("No se pudo Eliminar profesor en DB");
    return;
}

```

Figura 24: Código en el *Program.cs*

Consola de depuración de Microsoft Visual Studio

```

Hola Profesores
Elimino al Profesor
Agregue el id del profe a Eliminar
13
Buscando al profe a borrar...
Profesor Eliminado

```

Figura 25: Lo que vemos por consola

100 %

Results Messages

	Id	Nombre	Apellido	Documento	Telefono	Mail	Sueldo	FechaNacimiento
1	1	Carmen	Nuñez	20232654	1122334455	carmen@iafe.uba.ar	1000.00	1960-01-01 00:00:00.000
2	10	(pepe)	Nuñez	20232654	1165658989	carmen@iafe.uba.ar	1000.00	1990-01-01 00:00:00.000
3	12	Rodrigo	Moreiras	20232654	1165658989	rodrigo.moreiras22@gmail.com	1000.00	1990-01-01 00:00:00.000

Figura 26: Base de datos después de Eliminar

### 3. Tarea Parte Dos

#### 3.1. Armado de la Base de Datos para App Stock

Armos la base de datos StockApp con la siguiente estructura

##### 3.1.1. Trabajamos con la Tabla de Artículos

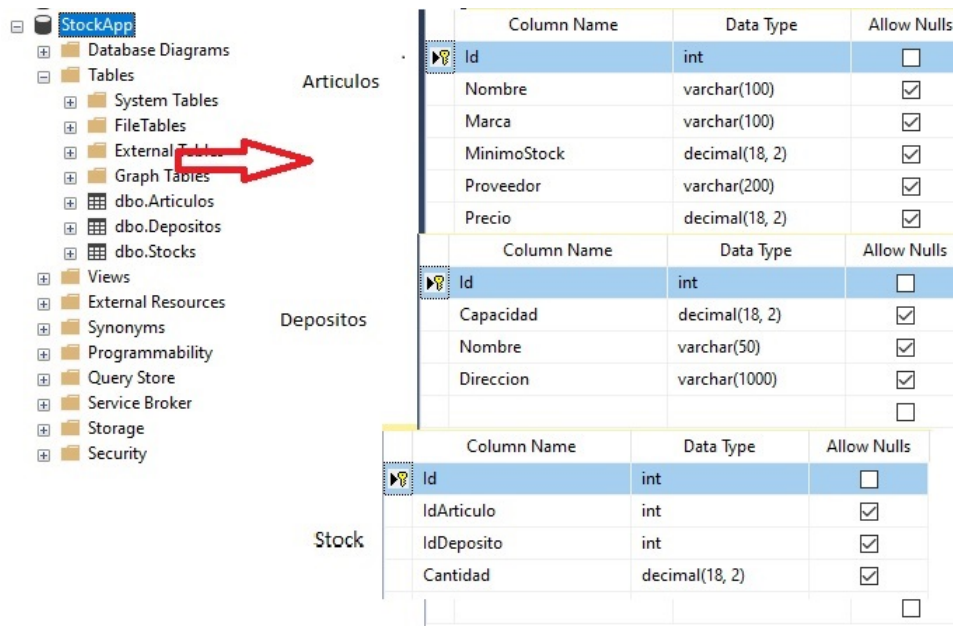


Figura 27: Árbol de la base de Datos

Ahora, agrego los tres Artículos

```
SQLQuery3.sql - DE...P-FI43P8O\Hp (63)) * - X
insert into articulos (Nombre, Marca, MinimoStock, Proveedor, Precio)
Values ('Hanna 0000', 'Blanca Flor', 30, 'Molinos', 130)

insert into articulos (Nombre, Marca, MinimoStock, Proveedor, Precio)
Values ('Fideos', 'Knor', 10, 'Knor Argentina', 160)

insert into articulos (Nombre, Marca, MinimoStock, Proveedor, Precio)
Values ('Arroza Oro', 'Gallo', 10, 'Molinos', 140)

100 %
Messages
(1 row affected)
(1 row affected)
(1 row affected)
Completion time: 2023-03-26T16:11:03.0927483-03:00
```

Figura 28: Código SQL para el agregado de los articulos

Los cuales, podemos verificar que se hayan agregado correctamente

100 %

	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina 0000	Blanca Flor	30.00	Molinos	130.00
2	2	Fideos	Knor	10.00	Knor Argentina	160.00
3	3	Arroza Oro	Gallo	10.00	Molinos	140.00

Figura 29: Articulos Agregados

Modificamos el Nombre y el precio del primer articulo

```
update articulos set Nombre = 'Harina Integral', Precio = 180
where id=1
```

Figura 30:Codigo para actualizar el articulo indicado

Vemos que el resultado nos queda

	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	2	Fideos	Knor	10.00	Knor Argentina	160.00
3	3	Arroza Oro	Gallo	10.00	Molinos	140.00

Figura 31: Articulo Actualizado en la DB

Y como ya no trabajamos con Fideos, los eliminamos de la base de datos

```
delete articulos where id=2
```

Figura 32:Codigo para Eliminar el articulo indicado

Vemos que el resultado nos queda

100 %

	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00

Figura 33: Vemos que se elimino el articulo correctamente

Listo, no trabajamos mas con fideos.

### 3.1.2. Trabajamos con la Tabla de Depositos

Sobre la tabla de depositos, agregamos tres Depositos diferentes (con datos ficticios)

```

insert into depositos (Capacidad, Nombre, Direccion)
Values (1000, 'Mini depositos', 'Av Nazca 3500')

insert into depositos (Capacidad, Nombre, Direccion)
Values (3000, 'Deposito Promedio', 'Av Cabildo 2500')

insert into depositos (Capacidad, Nombre, Direccion)
Values (1000, 'Super Deposito', 'Av Corrientes 4000')

```

Figura 34: Código SQL para agregar los depositos

Verificamos si se agregaron correctamente

Results		Messages		
	Id	Capacidad	Nombre	Direccion
1	1	1000.00	Mini depositos	Av Nazca 3500
2	2	3000.00	Deposito Promedio	Av Cabildo 2500
3	3	1000.00	Super Deposito	Av Corrientes 4000

Figura 35: Depositos en la DB

Ahora, Mini Depositos esta creciendo y debido a esto aumentaron su capacidad y Cambiaron de nombre, por lo que hay que cambiar sus datos en la Base de Datos

```

update depositos set Capacidad = 4000, Nombre = 'Depositos Mayores'
where id=1

```

Figura 36: Actualizamos el deposito de Id=1

Veamos si se actualizo Correctamente

Results		Messages		
	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	2	3000.00	Deposito Promedio	Av Cabildo 2500
3	3	1000.00	Super Deposito	Av Corrientes 4000

Figura 37: En Base de Datos

Debido a problemas diversos, el deposito Promedio debio cerrar, por lo que ya no es necesario en la base. Entonces lo eliminamos

```
delete depositos where id=2
```

Figura 38: En Base de Datos

Vemos en la base si se eliminó correctamente

results		Messages		
	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Corrientes 4000

Figura 39: En Base de Datos

### 3.1.3. Trabajamos con la Tabla de Stock

Agrego los stocks

```
insert into stocks (IdArticulo, IdDeposito, Cantidad)
Values (1,1,500)
insert into stocks (IdArticulo, IdDeposito, Cantidad)
Values (2,2,800)
insert into stocks (IdArticulo, IdDeposito, Cantidad)
Values (3,3,400)
```

Figura 40: Comando para Agregar Stocks

Verificamos que hayan ingresado

Results		Messages		
	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	500.00
2	2	2	2	800.00
3	3	3	3	400.00

Figura 41: Bd de Stocks

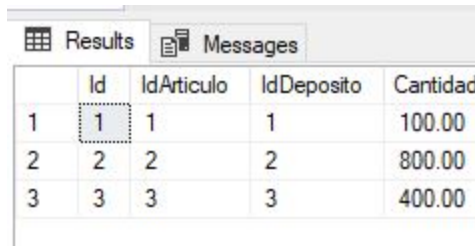
Ahora, modificamos la cantidad del Stock de id=1

```
update stocks set Cantidad = 100
where id=1
```

Figura 42: Comando para actualizar

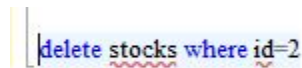
Verificamos que haya actualizado





	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	2	2	2	800.00
3	3	3	3	400.00

Figura 43: Bd de Stocks actualizada

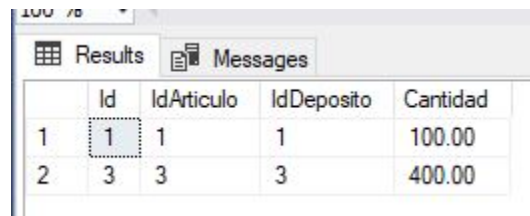


```
delete stocks where id=2
```

Figura 44: Comando de Eliminación

Eliminamos el Stock de Id=2

Verificamos que se haya eliminado



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	3	3	400.00

Figura 45: Bd de Stocks resultante

### 3.2. Parte de Código

Una vez descargada la App de Stock, tenemos que vincularla en la base de datos. Para esto debemos agregar los datos de nuestro localhost en la clase de AccesoDatos



```
0 referencias
public AccesoDatos(string enterpriseRelationship, string prueba)
{
    //Para SQL en la nube
    //serverBD = "SQL5110.site4now.net";
    //usuarioBD = "db_a96920_alumnos_admin";
    //PasswordBD = "Bootcamp2023";
    //basedatos = "db_a96920_alumnos";

    //armo conexion a la BD
    //strcondatos = @"Data Source=" + serverBD + ";Initial Catalog=" + basedatos + ";User ID=" + usuarioBD + ";Password=" + PasswordBD;

    //Para SQL en mi computadora
    serverBD = "DESKTOP-FI43P8O\\SQLEXPRESS01";
    basedatos = "StockApp";
    strcondatos = @"Data Source=" + this.serverBD + ";Initial Catalog=" + this.basedatos + ";Integrated Security=true";
}
}
```

Figura 46: Conexión Con Local Host

Luego, en las clases **Articulo**, **Deposito** y **Stock** debemos agregar las siguientes lineas para poder acceder a la librería de AccesoDatos y así poder trabajar con esta.



```
using AppObjetos.Datos;

private AccesoDatos ac = new AccesoDatos();
```

Figura 47: Agregar en todas las clases

### 3.2.1. Trabajamos con los Articulos

#### AgregarEnDb()

Creamos el método **AgregarEnDb()**

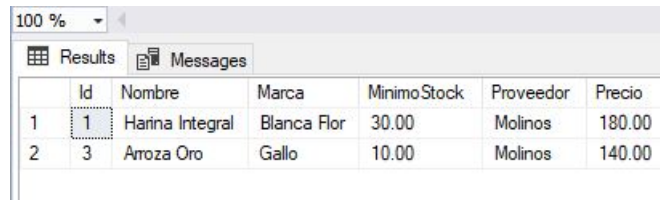
```
/// <summary>
/// Metodo utilizado para agregar al articulo en la base de datos
/// </summary>
/// <returns>
/// 1 => si salio todo bien
/// -1 => si salio todo mal
/// </returns>
1 referencia
public int AgregarEnDb()
{
    string query = $"insert into articulos (Nombre, Marca, MinimoStock, Proveedor, Precio)";
    query += $"values('{this.Nombre}', '{this.Marca}', {this.MinimoStock}, '{this.Proveedor}', {this.Precio}) ";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        //Log.Escribir(0, "ERROR", "Ocurrió un error en ChangeAddPartidaToProduct actualizando que guarde partida a producto " + ex.Message);
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}
```

Figura 48: Código del método

```
Articulo Articulo1 = new Articulo {
    Nombre = "Papas fritas (C#)",
    Marca = "Lays",
    MinimoStock = 15,
    Proveedor = "MiamiLand",
    Precio = 200,
};
// Agrego el Articulo con el metodo AgregarEnDb()
int resultado = Articulo1.AgregarEnDb();
// Valido
if (resultado != 1)
{
    Console.WriteLine("No se pudo agregar Profesor en DB");
    return;
}
else {
    Console.WriteLine("Profesor agregado en la DB\n");
}
```

Figura 49: Instancio en *Programs.cs*

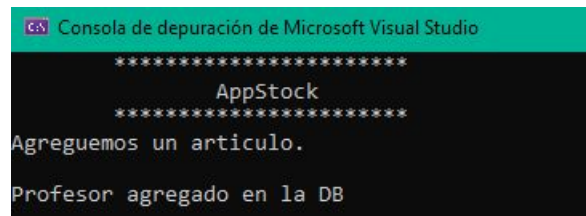
Base de datos antes de correr el programa



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00

Figura 50: Base de Datos Antes

Corro el programa, vemos por consola que aparentemente salio todo bien



```

*****
AppStock
*****
Agreguemos un articulo.
Profesor agregado en la DB
  
```

Figura 51: Salida Por consola Al agregar un Articulo

Verificamos que se haya agregado el Articulo a la base de datos



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00
3	4	Papas fritas (C#)	Lays	15.00	Miamland	2000.00

Figura 52: Base de datos luego de agregar el articulo

## ActualizarEnDb()

Ahora, Creamos el metodo **ActualizarEnDb()**. Este debe recibir un **Articulo** que va a corresponderse con el **Articulo** a modificar.

```

/// <summary>
/// Actualiza los datos de un articulo segun los datos pasador por parametro
/// </summary>
/// <param name="articuloAAActualizar"></param>
/// <returns></returns>
0 referencias
public int ActualizarEnDb(Articulo articuloAAActualizar)
{
    string query = $"update articulos set Nombre = '{articuloAAActualizar.Nombre}', " +
        $"Marca='{articuloAAActualizar.Marca}', " +
        $"MinimoStock={articuloAAActualizar.MinimoStock}, " +
        $"Proveedos = '{articuloAAActualizar.Proveedor}', " +
        $"Precio = {articuloAAActualizar.Precio}, " +
        $"where id = {articuloAAActualizar.Id}";

    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 53:Codigo del Metodo ActualizarEnDb()

Inicializamos un articulo con el proveedor modificado

```

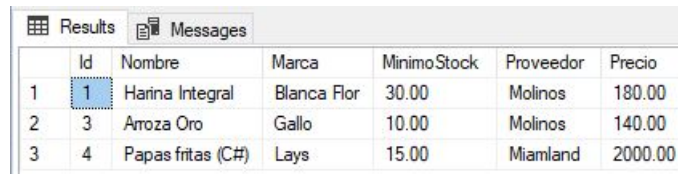
//Instancio El Articulo modificandole el proveedor
Console.WriteLine("Agreguemos un articulo.\n");
Articulo Articulo1 = new Articulo();
Articulo1.Id = 4;
Articulo1.Nombre = "Papas fritas (C#)";
Articulo1.Marca = "Lays";
Articulo1.MinimoStock = 15;
Articulo1.Proveedor = "ElPreoveedorModificado";
Articulo1.Precio = 2000;

//Instancio un articulo auxiiar para llamar al metodo y agregarlo
Articulo ArticuloAuxiliar =new Articulo();
// Agrego el Articulo con el metodo AgregarEnDb()
int resultado = ArticuloAuxiliar.ActualizarEnDb(Articulo1);
// Valido
if (resultado != 1)
{
    Console.WriteLine("No se pudo actualizar Articulo en DB");
    return;
}
else {
    Console.WriteLine("Profesor Actualizar en la DB\n");
}

```

Figura 54:Codigo del *Program.cs*

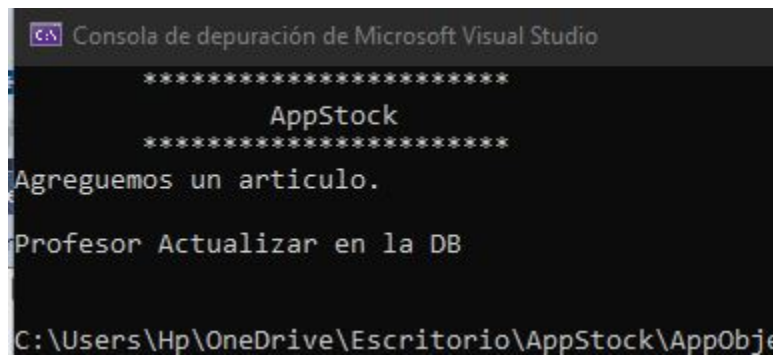
Revisamos como estaba antes de actualizar la base de datos



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00
3	4	Papas fritas (C#)	Lays	15.00	Miamland	2000.00

Figura 55: Base de Datos antes de actualizar

Actualizamos, vemos que aparece en la consola



```

*****
AppStock
*****
Agreguemos un articulo.
Profesor Actualizar en la DB
C:\Users\Hp\OneDrive\Escritorio\AppStock\AppObj
  
```

Figura 56: Salida Por consola al actualizar

Verificamos la base de datos



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00
3	4	Papas fritas (C#)	Lays	15.00	ElProveedorModificado	2000.00

Figura 57: Base de datos luego de actualizar el dato del proveedor

## EliminarEnDb()

Ahora, escribimos el método para eliminar un artículo de la base de datos. Este nos queda de la siguiente manera

```
/// <summary>
///  Elimina los datos de un artículo en la base de datos
/// </summary>
/// <param name="idArticuloEliminar"></param>
/// <returns></returns>
1 referencia
public int EliminarEnDb(int idArticuloEliminar)
{
    string query = $"delete articulos where id={idArticuloEliminar}";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}
```

Figura 58: Código del Método EliminarEnDb()


Agregamos un artículo, para luego eliminarlo

```
//Instancio El Artículo modificandole el proveedor
Console.WriteLine("Agreguemos un artículo.\n");
Articulo articulo1 = new Articulo();
Articulo1.Id = 5;
Articulo1.Nombre = "Nachos (C#)";
Articulo1.Marca = "Doritos";
Articulo1.MinimoStock = 25;
Articulo1.Proveedor = "Nachito";
Articulo1.Precio = 200;

if (Articulo1.AgregarEnDb() != 1) {
    Console.WriteLine("No se pudo Agregar Artículo en DB");
    return;
}
else
{
    Console.WriteLine("Artículo Agregado en la DB\n");
}
```

Figura 59: Agrego un artículo para eliminarlo

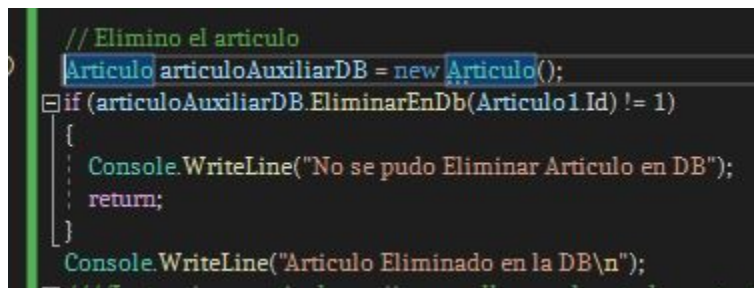
Vemos si se agrego a la base



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00
3	4	Papas fritas (C#)	Lays	15.00	ElPreoveedorModificado	2000.00
4	5	Nachos (C#)	Doritos	25.00	Nachito	200.00

Figura 60: Reviso Base de Datos

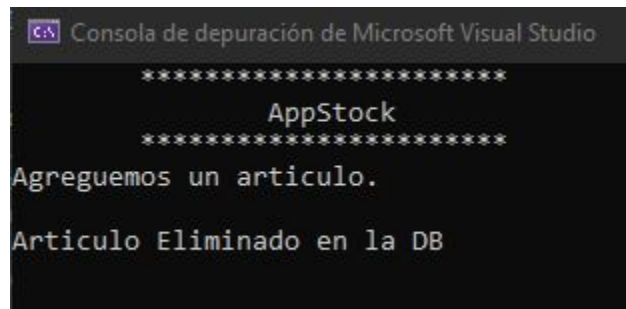
Ahora, utilizo el metodo para eliminar el articulo



```
// Elimino el articulo
Articulo articuloAuxiliarDB = new Articulo();
if (articuloAuxiliarDB.EliminarEnDb(Articulo1.Id) != 1)
{
    Console.WriteLine("No se pudo Eliminar Articulo en DB");
    return;
}
Console.WriteLine("Articulo Eliminado en la DB\n");
```

Figura 61: Uso el metodo

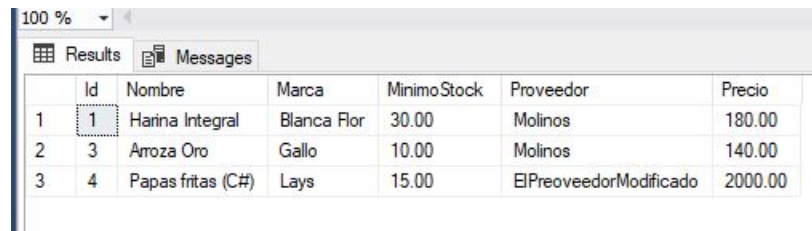
La salida por consola nos indica que lo elimino



```
*****
AppStock
*****
Agreguemos un articulo.
Articulo Eliminado en la DB
```

Figura 62: Salida por consola

Verifico en la base de datos si realmente se elimino



	Id	Nombre	Marca	MinimoStock	Proveedor	Precio
1	1	Harina Integral	Blanca Flor	30.00	Molinos	180.00
2	3	Arroza Oro	Gallo	10.00	Molinos	140.00
3	4	Papas fritas (C#)	Lays	15.00	ElPreoveedorModificado	2000.00

Figura 63: Se elimino el articulo de la Db



### 3.2.2. Trabajamos con los Depositos

#### AgregarEnDb()

Ahora creamos en la clase deposito el método para agregar en la base de datos

```

/// <summary>
/// Metodo utilizado para agregar al articulo en la base de datos
/// </summary>
/// <returns>
/// 1 => si salio todo bien
/// -1 => si salio todo mal
/// </returns>
1 referencia
public int AgregarEnDb()
{
    string query = $"insert into depositos (Capacidad, Nombre, Direccion)";
    query += $"values({this.Capacidad}, '{this.Nombre}', '{this.Direccion}') ";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        //Log.Escribir(0, "ERROR", "Ocurrió un error en ChangeAddPartidaToProduct actualizando que guarde pa
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 64: Código del Método

Instanciamos un Deposito y lo agregamos

```

// Instancio un Deposito
Deposito Deposito1 = new Deposito();
Deposito1.Capacidad = 5000;
Deposito1.Nombre = "DepoAgregadoC#";
Deposito1.Direccion = "Av Monroe 3000";

if (Deposito1.AgregarEnDb() != 1) {
    Console.WriteLine("No se pudo Agregar el Deposito");
}
Console.WriteLine("\t\tDeposiposito Agregado!!!!");

```

Figura 65: Instancio y agrego

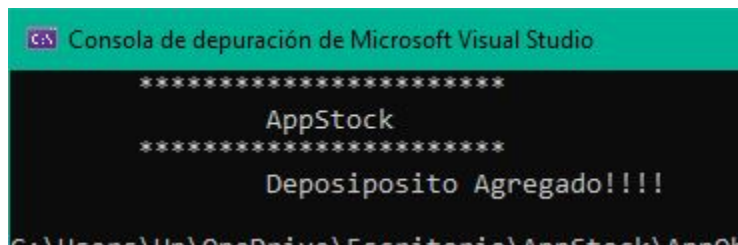
Revisamos como esta la base de datos antes de agregar un Deposito



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Corrientes 4000

Figura 66: Base de Datos antes de agregar

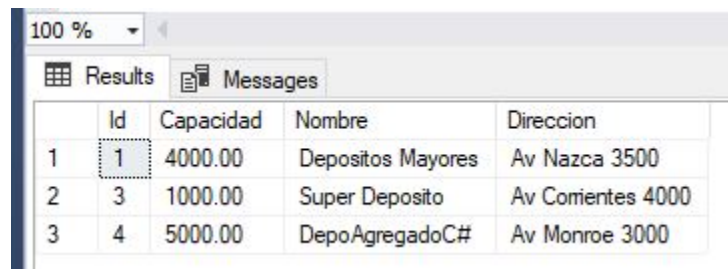
Según la consola salio todo bien



```
*****  
AppStock  
*****  
Deposito Agregado!!!!
```

Figura 67: Base de Datos antes de agregar

Verificamos que se haya agregado el deposito



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Corrientes 4000
3	4	5000.00	DepoAgregadoC#	Av Monroe 3000

Figura 68: Base de Datos despues de agregar

## ActualizarEnDb()

Creo el metodo ActualizaEnDb()

```

/// <summary>
///   Actualiza los datos de un deposito segun los datos pasador por parametro
/// </summary>
/// <param name="depositoAAActualizar"></param>
/// <returns></returns>
1 referencia
public int ActualizarEnDb(Deposito depositoAAActualizar)
{
    string query = $"update depositos set Capacidad = {depositoAAActualizar.Capacidad}, " +
        $"Nombre='{depositoAAActualizar.Nombre}', " +
        $"Direccion = '{depositoAAActualizar.Direccion}' " +
        $"where id = {depositoAAActualizar.Id}";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 69: Código Método ActualizarEnDb()

Instanciamos un deposito con el nombre actualizado

```

/*****
 *   Ejecucion
 *****/
// Titulo
Console.WriteLine("\t*****\n\t\tAppStock\n\t*****");

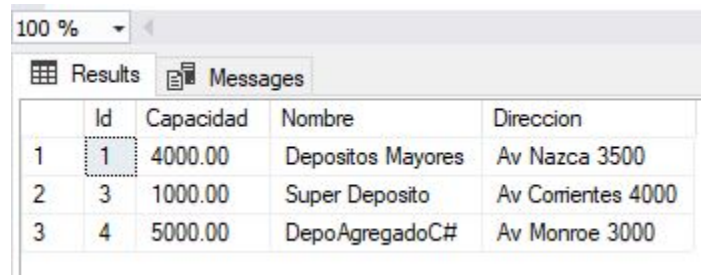
// Instancio un Deposito Modificando el Nombre
Deposito Deposito1 = new Deposito();
Deposito1.Id = 4;
Deposito1.Capacidad = 5000;
Deposito1.Nombre = "DepoActualizadoC#";
Deposito1.Direccion = "Av Monroe 3000";

// Instancio un deposito Auxiliar
Deposito depoAuxiliar = new Deposito();
//
if (depoAuxiliar.ActualizarEnDb(Deposito1) != 1) {
    Console.WriteLine("No se pudo Actualizar el Deposito");
}
else {
    Console.WriteLine("\t\tDeposito Actualizado!!!!");
}

```

Figura 70: Instanciamos

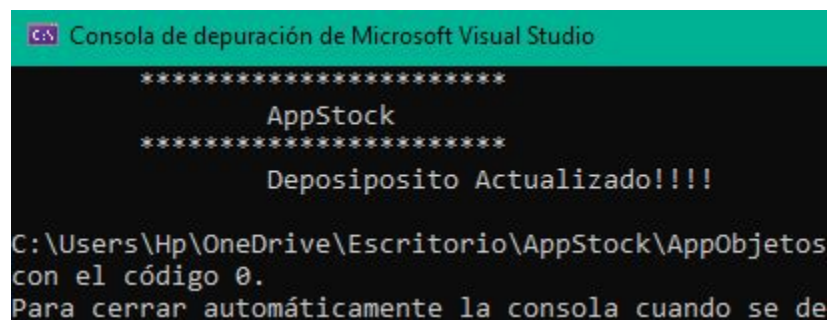
Revisamos la base de datos antes de actualizar



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Comientes 4000
3	4	5000.00	DepoAgregadoC#	Av Monroe 3000

Figura 71: Base de datos Antes

Ejecutamos lo instanciado obteniendo por consola que salio todo bien



```

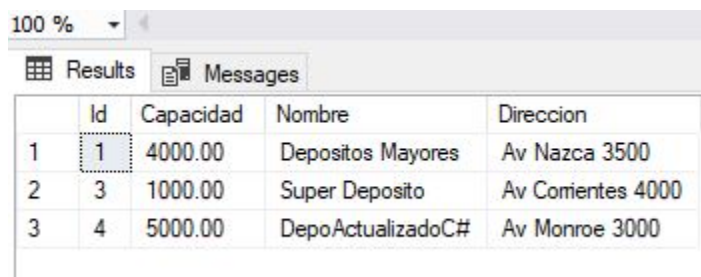
*****
AppStock
*****
Deposito Actualizado!!!!

C:\Users\Hp\OneDrive\Escritorio\AppStock\AppObjetos
con el código 0.
Para cerrar automáticamente la consola cuando se de

```

Figura 72: Salida por consola

Verificamos que se haya actualizado en la base el nombre



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Comientes 4000
3	4	5000.00	DepoActualizadoC#	Av Monroe 3000

Figura 73: Db actualizada

## EliminarEnDb()

Ahora creamos en la clase deposito El método EliminarEnDb()

```

/// <summary>
///  Elimina los datos de un deposito en la base de datos
/// </summary>
/// <param name="idDepositoEliminar"></param>
/// <returns></returns>
1 referencia
public int EliminarEnDb(int idDepositoEliminar)
{
    string query = $"delete depositos where id={idDepositoEliminar}";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 74: Se elimino el articulo de la Db

Escribimos en el program el codigo necesario para hacer una eliminación basica

```

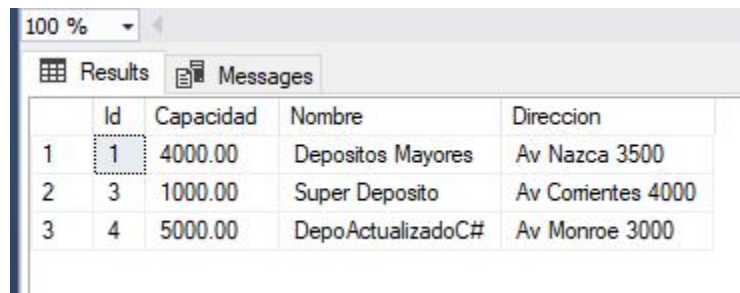
// Instancio un Deposito Modificando el Nombre
Deposito Deposito1 = new Deposito();
Deposito1.Id = 4;
Deposito1.Capacidad = 5000;
Deposito1.Nombre = "DepoActualizadoC#";
Deposito1.Direccion = "Av Monroe 3000";

// Instancio un deposito Auxiliar
Deposito depoAuxiliar = new Deposito();
//
if (depoAuxiliar.EliminarEnDb(Deposito1.Id) != 1) {
    Console.WriteLine("No se pudo Eliminar el Deposito");
}
else {
    Console.WriteLine("\t\tDeposito Eliminado!!!");
}

```

Figura 75: Código *Programs.cs*

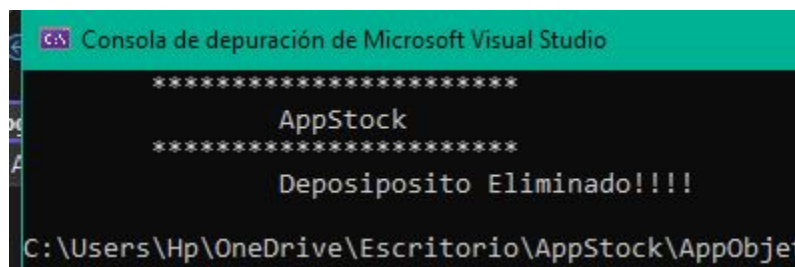
Vemos como estaba antes de eliminar el deposito en la base de datos



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Comientes 4000
3	4	5000.00	DepoActualizadoC#	Av Monroe 3000

Figura 76: Db antes

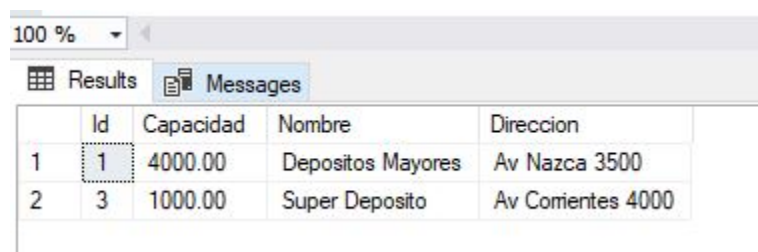
Segun la consola salio todo bien



```
C:\Users\Hp\OneDrive\Escritorio\AppStock>AppObjet
*****
AppStock
*****
Deposito Eliminado!!!!
```

Figura 77: Salida Por consola

Verifico en la base de datos que se haya eliminado el deposito indicado



	Id	Capacidad	Nombre	Direccion
1	1	4000.00	Depositos Mayores	Av Nazca 3500
2	3	1000.00	Super Deposito	Av Comientes 4000

Figura 78: Db despues de eliminar



### 3.2.3. Trabajamos con los Stocks

#### AgregarEnDb()

Creo el metodo AgregarEnDb(). En este caso, debiamos pasarle solo el Id a la query.

```
1 referencia
public int AgregarEnDb()
{
    string query = $"insert into stocks ([IdArticulo, IdDeposito, Cantidad]);
    query += $"values({this.ArticuloGuardado.Id}, {this.DepositoDondeEstaGuardado.Id}, {this.Cantidad})";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        //Log.Escribir(0, "ERROR", "Ocurrio un error en ChangeAddPartidaToProduct actualizando que guarde partida a pr
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}
```

Figura 79: Metodo AgregarEnDb()

Sin embargo, al instanciar el Stock debiamos instanciarlo con un articulo y un deposito a los cuales los instanciamos con los Id para que se pueda agregar el stock.

```
// Instancio un Articulo y lo agrego
Articulo Articulo1 = new Articulo();
Articulo1.Id = 7;
Articulo1.Nombre = "ArticuloParaStock (C#)";
Articulo1.Marca = "Doritos";
Articulo1.MinimoStock = 20;
Articulo1.Proveedor = "Nachito";
Articulo1.Precio = 400;

// Instancio un Deposito y lo agrego
Deposito Deposito1 = new Deposito();
Deposito1.Id = 6;
Deposito1.Capacidad = 4000;
Deposito1.Nombre = "Depo para Stock (C#)";
Deposito1.Direccion = "Av Monroe 3500";
```

Figura 80: Articulo y Deposito a instanciar en Stock

Ahora, instanciamos el Stock y lo agregamos

```
// Instancio un stock Con los articulos y deposito instanciados anteriormente
Stock stock1 = new Stock();
stock1.ArticuloGuardado = Articulo1;
stock1.DepositoDondeEstaGuardado = Deposito1;
stock1.Cantidad = 200;

if (stock1.AgregarEnDb() != 1)
{
    Console.WriteLine("No se pudo Agregar el stock");
}
else {
    Console.WriteLine("Se agrego el Stock");
}
```

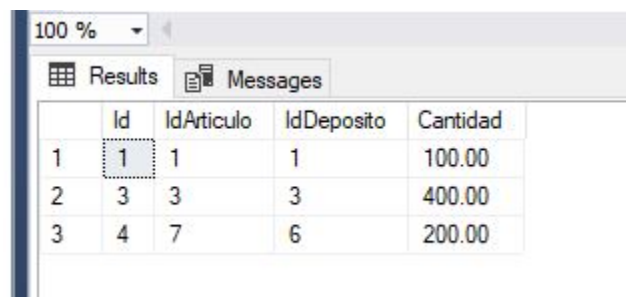
Figura 81: Instancio y agrego Stock

Según la consola salio todo bien

```
*****
AppStock
*****
Agreguemos un articulo.
Se agrego el Stock
```

Figura 82: Salida por Consola

Ahora, verificamos que se haya agregado el stock en la base de datos



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	3	3	400.00
3	4	7	6	200.00

Figura 83: Db después de agregado el stock

## ActualizarEnDb()

En este caso el código del método nos queda de la siguiente forma

```

/// <summary>
/// Actualiza los datos de un articulo segun los datos pasador por parametro
/// </summary>
/// <param name="stockAAActualizar"></param>
/// <returns></returns>
1 referencia
public int ActualizarEnDb(Stock stockAAActualizar)
{
    string query = $"update stocks set IdArticulo = {stockAAActualizar.ArticuloGuardado.Id}, "
        $"IdDeposito={stockAAActualizar.DepositoDondeEstaGuardado.Id}, " +
        $"Cantidad={stockAAActualizar.Cantidad} " +
        $"where id = {stockAAActualizar.Id}";

    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}

```

Figura 84: Código del método ActualizarEnDb

Reutilizando el Artículo y el Deposito instanciados anteriormente, instanciamos un Stock con  $Id = 3$  a actualizar

```

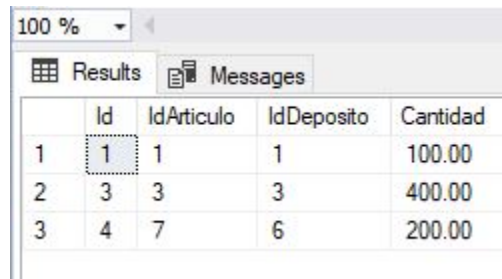
// Instancio un stock Con los articulos y deposito instanciados anteriormente
Stock stock1 = new Stock();
stock1.Id = 3;
stock1.ArticuloGuardado = Articulo1;
stock1.DepositoDondeEstaGuardado = Deposito1;
stock1.Cantidad = 500;

// Modifico la capacidad del Stock
Stock stockAuxiliar = new Stock();
if (stockAuxiliar.ActualizarEnDb(stock1) != 1)
{
    Console.WriteLine("No se pudo Actualizar el stock");
}
else {
    Console.WriteLine("Se Actualizo el Stock");
}

```

Figura 85: Instancia y llamada del método ActualizarEnDb()

Vemos como estaba la Base de Datos antes de actualizar



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	3	3	400.00
3	4	7	6	200.00

Figura 86: Base de Datos Antes

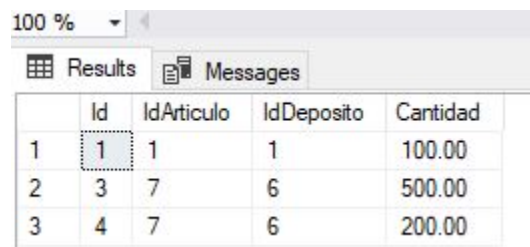
Ejecutamos y vemos en la consola que se actualizo



```
*****  
AppStock  
*****  
Se Actualizo el Stock
```

Figura 87: Salida por Consola

Verificamos la actualización en la base de datos



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	7	6	500.00
3	4	7	6	200.00

Figura 88: Base de datos después de la Actualización

### EliminarEnDb()

Por ultimo, creamos el método EliminarEnDb(). El código se puede ver en la Figura 89

```
/// <summary>
///   Elimina los datos de un articulo en la base de datos
/// </summary>
/// <param name="idStockEliminar"></param>
/// <returns></returns>
1 referencia
public int EliminarEnDb(int idStockEliminar)
{
    string query = $"delete stocks where id={idStockEliminar}";
    try
    {
        SqlCommand command = new SqlCommand(query);
        int r = ac.ejecQueryDevuelveInt(command);
        return r;
    }
    catch (Exception ex)
    {
        return -1;
    }
    finally
    {
        ac.DesConectar();
    }
}
```

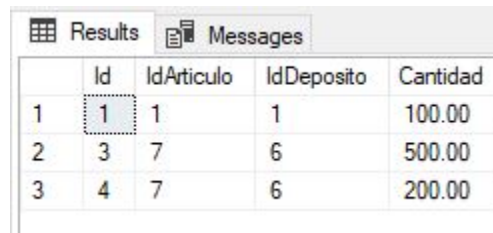
Figura 89: Código del Método EliminarEnDb()

Eliminamos según el  $Id = 4$

```
int idStock = 4;
// Modifico la capacidad del Stock
Stock stockAuxiliar = new Stock();
if (stockAuxiliar.EliminarEnDb(idStock) != 1)
{
    Console.WriteLine("No se pudo Eliminar el stock");
}
else {
    Console.WriteLine("Se Elimino el Stock");
}
```

Figura 90: *Programs.cs* para eliminar el Stock de Id=4

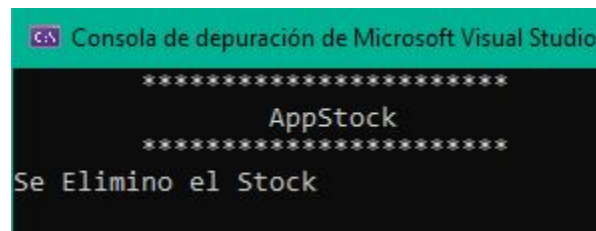
Vemos la base de datos Antes



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	7	6	500.00
3	4	7	6	200.00

Figura 91: Base de Datos Antes

Según la consola salio todo bien

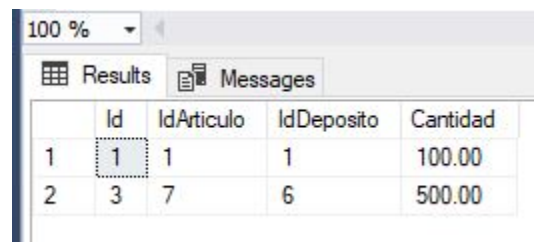


```
C:\> Consola de depuración de Microsoft Visual Studio

*****
AppStock
*****
Se Elimino el Stock
```

Figura 92: Salida por Consola

Verificamos en la base de datos que se haya eliminado el stock.



	Id	IdArticulo	IdDeposito	Cantidad
1	1	1	1	100.00
2	3	7	6	500.00

Figura 93: Base de Datos Después

## 4. Referencias