Integracion de Mongodb (DB no relacional) con c#

Mongodb es un sistema de base de datos NoSql ortientado a documentos, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (muy similar a JSON) con un esquema dinamico. El mismo se encuentra desarrollado en C++. Como principal ventaja o caracteristica que lo destaca es sin duda la velocidad, aunque tiene muchas mas.

Instalacion motor Mongodb y Cliente :

Link MongoDB : **Instalar desde la pagina oficial Mongodb Community Server.**

Instalacion de Visual Studio:

Link Visual Studio : **Instalar desde la pagina oficial Visual Studio.**

Nuevo proyecto e Instalacion de Driver :

- Crear proyecto en Visual Studio, el mismo puede ser proyecto de terminal, lo que sea mas comodo para probarlo.
- Dentro del proyecto abrir consola de administrador de paquetes NuGet (herramientas/Administrador de paquetes NuGet/Consola de administrador)
- Ejecutar en la consola : 'Install-Package MongoDB.Driver'
- En caso de tener errores configuracion el repositorio de paquetes (Ir administrador de paquetes NuGet en herramientas y agregar : 'https://api.nuget.org/v3/index.json')

Modelo de datos :

Como bien se puede imaginar tenemos la opcion al igual que Entity Framework para poder crear ORM (Object Relational Mapping) asi de esta manera mappear nuestras clases model con la coleccion correspondiente a consultar.
Ejemplo :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

namespace ReadMass.Class.Massive
{
    public class Massive
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; }

        [BsonElement("athlete")]
        public string Athlete { get; set; }

        [BsonElement("age")]
        public string Age { get; set; }

        [BsonElement("year")]
        public string Year { get; set; }

        [BsonElement("closingceremonydate")]
        public string ClosingCeremonyDate { get; set; }

        [BsonElement("goldmedals")]
        public string GoldMedals { get; set; }

        [BsonElement("silvermedals")]
        public string SilverMedals { get; set; }

        [BsonElement("bronzemedals")]
        public string BronzeMedals { get; set; }

        [BsonElement("totalmedals")]
        public string TotalMedals { get; set; }
    }
}
```

Conexion con MongoDB:

Ahora vamos a ver como se realiza la conexion al cluster de MongoDB, una ves teniendo nuestra instancia nos conectamos a la base de datos que se requiera. En este caso nos conectamos a "CentralData".

```
using System;
using MongoDB.Driver;

namespace ReadMass.Conection
{
    public class ConectionMongoDB
    {

        private static MongoClient con = null;
        private static IMongoDatabase _dataBase;

        public void GetConection( )
        {
            this.conexion();
        }

        public IMongoDatabase GetIntance( )
        {
            if( con == null)
            {
                this.GetConection();
            }

            return _dataBase;
        }

        private void conexion( )
        {
            try
            {
                con = new MongoClient("mongodb://localhost:27017");
                _dataBase = con.GetDatabase("CentralData");
            }
            catch ( Exception e )
            {
                throw new Exception("Error al intentar conectar
con la base de datos !!!" + e.Message.ToString());
            }
        }

    }
}
```

Coleccion, consulta, insert :

Por ultimo vamos a ver como utilizar la coleccion, realizar una consulta y un insert, en el caso del insert si la coleccion no existe la crea.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ReadMass.Conection;
using MongoDB.Driver;

namespace ReadMass.Services.Massive
{
    public class MassiveService
    {
        private IMongoDatabase _coneccion;
        private IMongoCollection massiveDB;

        public MassiveService( ConectionMongoDB conDB )
        {
            this._coneccion = conDB.GetIntance();
            this.massiveDB = this._coneccion.GetCollection("Massive");
        }

        public async Task addMassive (ReadMass.Class.Massive.Massive data)
=> await Task.FromResult(massiveDB.InsertOneAsync(data));
```

```
                                public async Task>
    getMassiveBuilderFilter( string filter )
                                {
                                    //Obtenemo todos los datos
                                    filter = Builders.Filter.Eq(x => x.Year, filter );
                                    return (List) await massiveDB.FindAsync(
    (FilterDefinition)filter );
                                }

                                public async Task> getMassive( string filter )
                                {
                                    //Obtenemos las primeras 30 tuplas de la consulta.
                                    var result = await massiveDB.Find(x => x.Year == filter)
    .Skip(0).Limit(30).ToListAsync();
                                    return result;
                                }

                            }
                        }
```