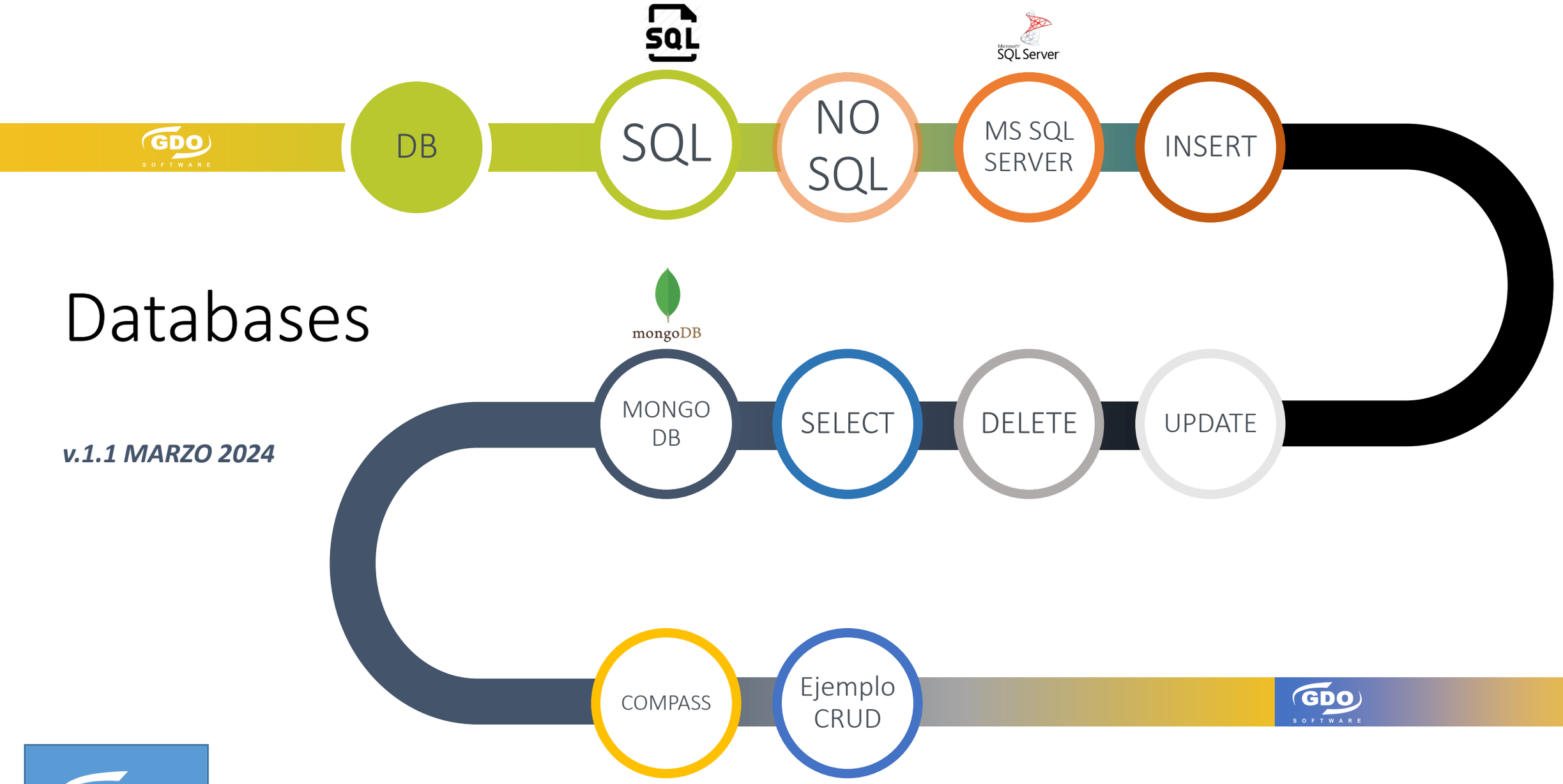
CAPÍTULO 4. Databases

v.1.1 MARZO 2024

Ricardo Moraleda Gareta

[Director departamento de software de GDO Software]



Databases

v.1.1 MARZO 2024

DATABASES

SQL / NoSQL

Structured Query Language (SQL) es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos **relacionales**.

He trabajado con varias bases de datos:

SQL
- MySQL
- ORACLE
- Microsoft SQL Server
- IBM DB2
- PostgreSQL

NoSQL
- mongoDB

Ejemplos de bases de datos

1. Mostraré un ejemplo de base de datos relacional (MS SQL Server 2017 Express) y su aplicación con C# (.NET).

Microsoft SQL Server

C# Microsoft .NET

2. Mostraré un ejemplo de base de datos no relacional (Mongo DB Community) y su aplicación con C# (.NET).

mongoDB

C# Microsoft .NET

MS SQL SERVER

MS SQL SERVER

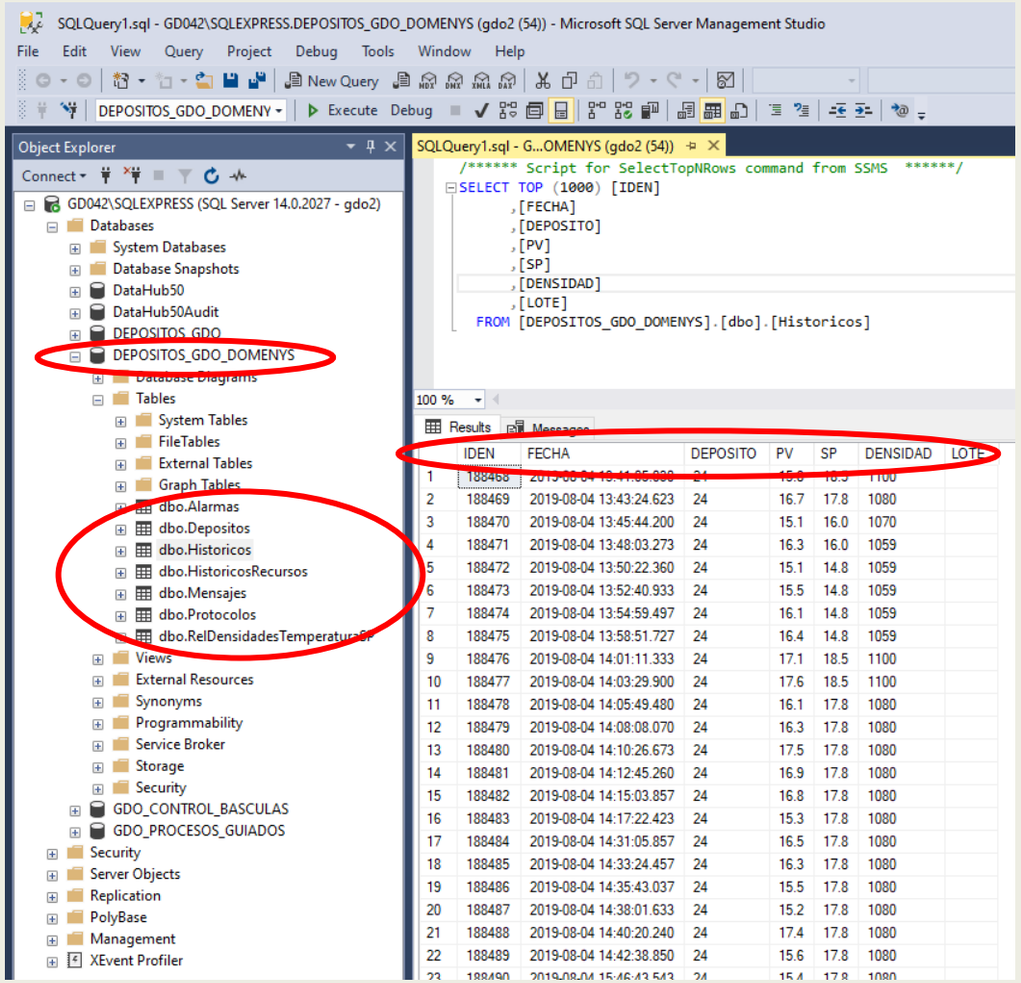
SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial.

https://www.microsoft.com/es-es/sql-server/sql-server-2017

Me he instalado en mi PC Windows MS SQL Server 2017 Express y su GUI MSSQLS Management Studio.

Se compone de la instancia de base de datos (PC\SQLEXPRESS), diferentes bases de datos. Cada base de datos se compone de tablas y cada tabla en columnas.

Resumiendo, existen las PK (Claves primarias) y las FK (Claves foráneas) para establecer las relaciones entre campos de las tablas.



MS SQL SERVER

MS SQL SERVER

Una base de datos se compone de varias tablas, denominadas relaciones.

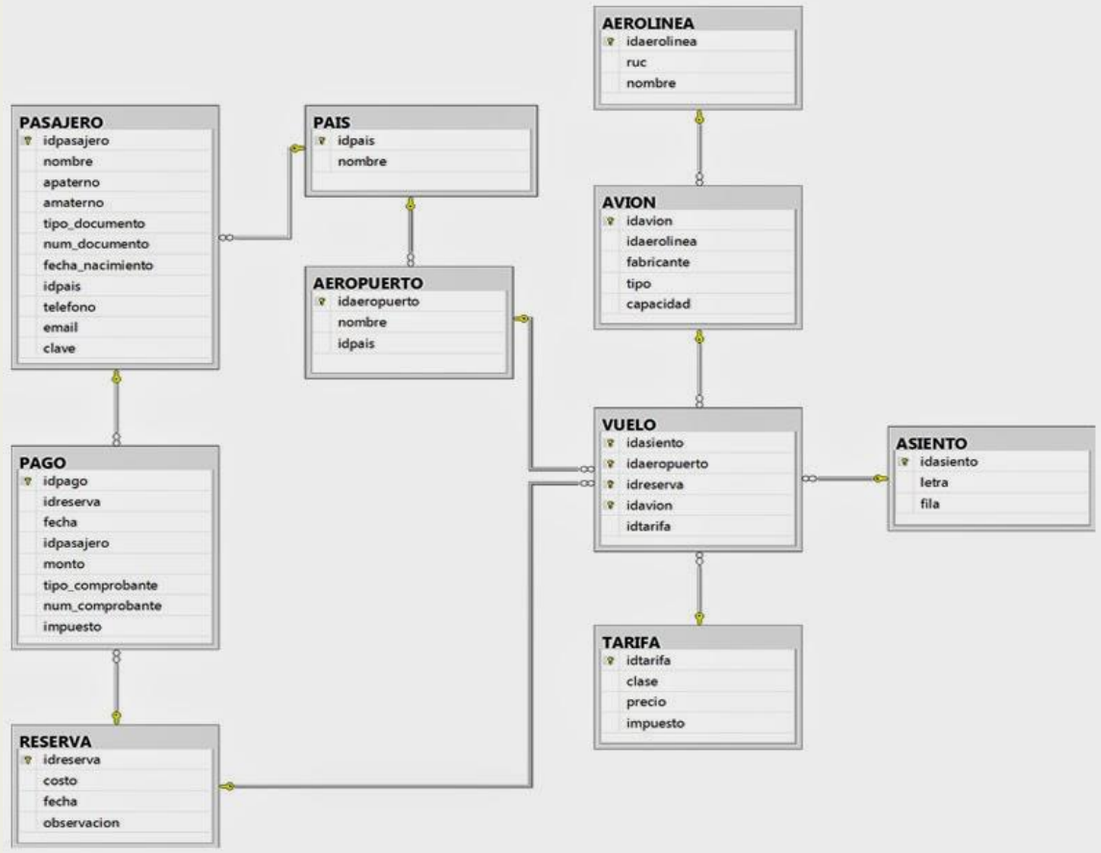
No pueden existir dos tablas con el mismo nombre ni registro.

Cada tabla es a su vez un conjunto de campos (columnas) y registros (filas).

La relación entre una tabla padre y un hijo se lleva a cabo por medio de las claves primarias y claves foráneas.

Las claves primarias (PK) son la clave principal de un registro dentro de una tabla y estas deben cumplir con la integridad de datos.

Las claves foráneas (FK) se colocan en la tabla hija, contienen el mismo valor que la clave primaria del registro padre; por medio de estas se hacen las formas relacionales.



MS SQL SERVER

C# INSERT

Vamos a hacer un ejemplo. En el fichero App.config de la solución del proyecto (Windows Forms) añadimos la conexión a base de datos llamada "ConexionBD"

```
<connectionStrings>
  <add name="ConexionBD" connectionString="Data Source=PC\SQLEXPRESS;Initial Catalog=DEPOSITOS_GDO_DOMENYS;User Id=usuario;Password=password"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Creamos una clase que contendrá todos los métodos referente a base de datos (insert, delete, update, select) programados de la manera tradicional.

El siguiente ejemplo es de inserción de datos. La query sql es *INSERT INTO <table> (campos) VALUES (values)*.

Crea una nueva conexión cogiendo los datos con la referencia "ConexionBD" creada antes y un Command (con la conexión y la query sql).

```
public bool writeHistory(Deposito dep)
{

    string query = "INSERT INTO dbo.Historicos (DEPOSITO, PV, SP, DENSIDAD, LOTE) VALUES (@deposito, @valor1, @valor2, @valor3, @valor4)";

    using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ConexionBD"].ConnectionString))
    {
        using (SqlCommand command = new SqlCommand(query, conn))
        {

            try
            {
                conn.Open();
                command.Parameters.AddWithValue("@deposito", dep.id);
                command.Parameters.AddWithValue("@valor1", dep.Temperatura);
                command.Parameters.AddWithValue("@valor2", dep.SetPointTemp);
                command.Parameters.AddWithValue("@valor3", dep.DensidadInicial);
                command.Parameters.AddWithValue("@valor4", dep.lote);

                int result = command.ExecuteNonQuery();

                // Check Error
                if (result <= 0)
                {
                    Scada.Log.Error("Error inserting data into Database!");
                }
                return true;
            }
            catch (Exception e)
            {
                if (conn.State == System.Data.ConnectionState.Open) conn.Close();
                Scada.Log.Error("Error al guardar histórico. " + e.ToString());
            }
        }
    }
    return false;
}
```

IDEN	FECHA	DEPOSITO	PV	SP	DENSIDAD	LOTE
188991	2019-08-06 20:42:37.180	24	16.2	33.0	1070	2019M2
188992	2019-08-06 20:44:55.737	24	17.2	33.0	1070	2019M2
188993	2019-08-06 20:47:14.320	24	18.0	33.0	1070	2019M2
188994	2019-08-06 20:49:32.903	24	15.2	33.0	1070	2019M2
188995	2019-08-06 20:51:51.483	24	15.2	33.0	1070	2019M2
188996	2019-08-06 20:54:10.070	24	15.6	33.0	1070	2019M2

Abre la conexión, rellena los parámetros con los atributos del objeto a insertar y llama a ExecuteNonQuery().

Se utiliza la sentencia using() en conexiones, comandos, etc. para despreocuparnos en cerrar conexiones y liberar recursos. Al finalizar lo hace por ti.

En este caso IDEN es un autonumérico (lo rellena la tabla) y FECHA se pone sola al insertar el registro ya que hemos configurado que el campo FECHA se rellene con getdate().

MS SQL SERVER

C# UPDATE

En la misma clase creada enteriormente crearemos otro método para actualizar datos ya existentes con la sentencia Update.

La query sql es *UPDATE <table> SET campo1 = valor1, campo2 = valor2 WHERE camporeferencia = valor*. Campo de referencia suele ser el id del dato existente que vamos a modificar.

Se quiere actualizar el campo FECHA_FIN con la fecha actual y ESTADO con uno específico "UNACK_RTN".

```
public bool updateAlarmCom(Deposito dep)
{
    string query = "UPDATE dbo.Alarmas SET FECHA_FIN = getdate(), ESTADO = @estado Where IDEN = @idAlarma AND DEPOSITO = @id";

    using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ConexionBD"].ConnectionString))
    {
        using (SqlCommand command = new SqlCommand(query, conn))
        {
            try
            {
                conn.Open();
                command.Parameters.AddWithValue("@id", dep.id);
                command.Parameters.AddWithValue("@idAlarma", dep.idLastAlarmCom);
                command.Parameters.AddWithValue("@estado", "UNACK_RTN");

                int result = command.ExecuteNonQuery();

                // Check Error
                if (result <= 0)
                {
                    Scada.Log.Error("Ninguna alarma que actualizar!");
                    return false;
                }
                return true;
            }
            catch (Exception e)
            {
                if (conn.State == System.Data.ConnectionState.Open) conn.Close();
                Scada.Log.Error("Error al actualizar la alarma. " + e.ToString());
            }

        }
    }
    return false;
}
```

	IDEN	FECHA_INI	FECHA_FIN	VALOR	DEPOSITO	ESTADO	TEXTO
1	83672	2019-07-23 17:41:51.777	2019-07-23 17:42:21.787	22.5	1	UNACK_RTN	TEMPERATURA FUERA DE MARGEN LO/HI
2	83673	2019-07-23 17:43:02.320	2019-07-23 17:43:22.337	21.1	1	UNACK_RTN	TEMPERATURA FUERA DE MARGEN LO/HI
3	83674	2019-07-23 17:43:42.353	2019-07-23 17:44:02.370	22.1	1	UNACK_RTN	TEMPERATURA FUERA DE MARGEN LO/HI
4	83675	2019-07-23 17:44:12.380	2019-07-23 17:44:22.390	20.5	1	UNACK_RTN	TEMPERATURA FUERA DE MARGEN LO/HI
5	83676	2019-07-23 17:44:42.403	2019-07-23 17:44:52.930	22.6	1	UNACK_RTN	TEMPERATURA FUERA DE MARGEN LO/HI

Abre la conexión, rellena los parámetros con los atributos del objeto a actualizar y llama a ExecuteNonQuery().

Se utiliza la sentencia using() en conexiones, comandos, etc. para despreocuparnos en cerrar conexiones y liberar recursos. Al finalizar lo hace por ti.

MS SQL SERVER

C# DELETE

En la misma clase creada enteriormente crearemos otro método para borrar datos ya existentes con la sentencia Delete.

La query sql es *DELETE FROM <table> WHERE camporeferencia = valor*. Campo de referencia suele ser el id del dato existente que vamos a borrar.

```
public bool delete(Deposito dep)
{

    string query = "DELETE FROM dbo.Depositos Where ID = @id";

    using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ConexionBD"].ConnectionString))
    {
        using (SqlCommand command = new SqlCommand(query, conn))
        {

            try
            {
                conn.Open();
                command.Parameters.AddWithValue("@id", dep.id);

                int result = command.ExecuteNonQuery();

                // Check Error
                if (result <= 0)
                {
                    Scada.Log.Error("Ningún depósito que borrar!");
                    return false;
                }
                return true;
            }
            catch (Exception e)
            {
                if (conn.State == System.Data.ConnectionState.Open) conn.Close();
                Scada.Log.Error("Error al borrar el depósito. " + e.ToString());
            }

        }
    }
    return false;
}
```

	IDEN	ID	DESCRIPCION	ZONA	HABILITADO	IP	PUERTOTCP	WORDTEMPPV
1	1	24	242	FILA1	1	127.0.0.1	502	100
2	2	23	243	FILA1	0	127.0.0.1	502	100
3	3	22	244	FILA1	0	127.0.0.1	502	100
4	4	21	245	FILA1	0	127.0.0.1	502	100

Abre la conexión, rellena el parámetro con el atributo del objeto a borrar y llama a ExecuteNonQuery().Se utiliza la sentencia using() en conexiones, comandos, etc. para despreocuparnos en cerrar conexiones y liberar recursos. Al finalizar lo hace por ti.

MS SQL SERVER

C# SELECT

En la misma clase creada enteriormente crearemos otro método para obtener datos existentes con la sentencia Select.

La query sql es *SELECT FROM <table> (WHERE camporeferencia = valor) ORDER BY campo_ordenacion ASC/DESC* (ascendente o descendente)

```
public DataSet getDepositos()
{
    string query = "SELECT * FROM dbo.Depositos order by iden";
    DataSet ds = null;

    using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ConexionBD"].ConnectionString))
    {
        using (SqlCommand command = new SqlCommand(query, conn))
        {

            try
            {
                conn.Open();

                ds = new DataSet();
                SqlDataAdapter da = new SqlDataAdapter(command);
                da.Fill(ds);
            }
            catch (Exception e)
            {
                if (conn.State == System.Data.ConnectionState.Open) conn.Close();
                Scada.Log.Error("Error al get depositos. " + e.ToString());
            }

        }
    }

    return ds;
}
```

	IDEN	ID	DESCRIPCION	ZONA	HABILITADO	IP	PUERTOTCP	WORDTEMPPV
1	1	24	242	FILA1	1	127.0.0.1	502	100
2	2	23	243	FILA1	0	127.0.0.1	502	100
3	3	22	244	FILA1	0	127.0.0.1	502	100
4	4	21	245	FILA1	0	127.0.0.1	502	100

Abre la conexión y relleno un DataSet con todas las filas que cumplen la condición de la query. Se utiliza la sentencia using() en conexiones, comandos, etc. para despreocuparnos en cerrar conexiones y liberar recursos. Al finalizar lo hace por ti.




MONGO DB

MONGO DB

Es una base de datos no relacional, es decir, NO SQL, basada en documentos (objetos) con sus elementos (propiedades) en estilo Binary JSON (BSON), es decir, base de datos orientada a objetos.

Construida para aplicaciones modernas y basadas en cloud (por ejemplo, para IoT – Internet of Things)

https://www.mongodb.com/

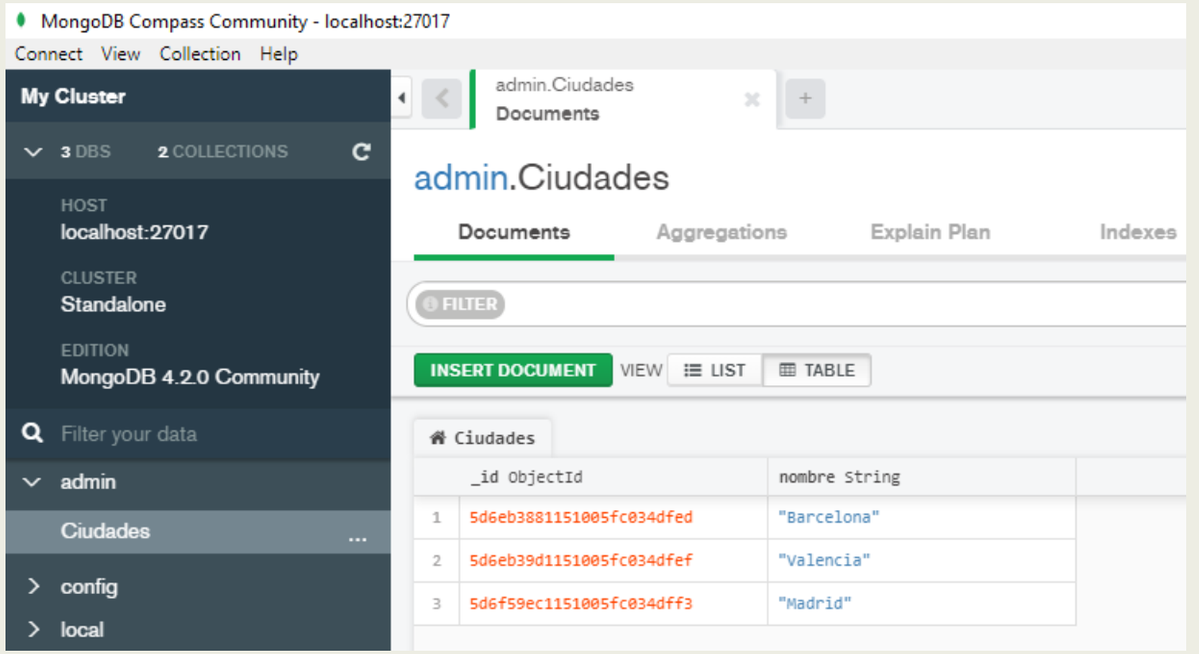
Me he instalado en mi PC Windows MongoDB Server 4.2.0 Community y su GUI MongoDB Compass 1.19.6 Community.

La he instalado como servicio:

MongoDB Server	MongoDB Database Server	En ejecución	Automático

La cadena de conexión o URL es:

mongodb://localhost:27017



He creado en la base de datos "admin" una colección llamada "Ciudades" con 3 documentos con 1 elemento llamado "nombre".

Diccionario NO SQL-SQL: **Colección** es como una Tabla, **Documento** es un objeto que podría ser una row de la tabla y **Elemento** es una propiedad del Documento, es decir, podría ser una column de la row.

MONGO DB

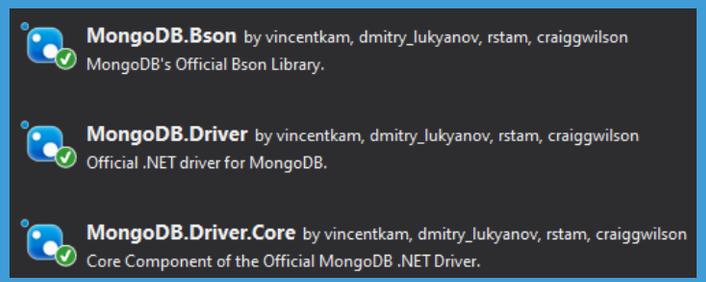
C#

Vamos a hacer un ejemplo CRUD (Create, Read, Update, Delete) con la colección creada "Ciudades". Basado en este enlace.

https://www.c-sharpcorner.com/article/getting-started-with-mongodb-mongodb-with-c-sharp/osoft.com/es-es/dotnet/csharp/

Utilizo el IDE Visual Studio 2015 y el framework .NET 4.5.2.

Creo una nueva aplicación Windows Forms y para esta solución me descargo los drivers de MongoDB 2.9.1. a través de Manage NuGet Packages de VS.



CREATE

```
MongoClient dbClient = new MongoClient("mongodb://localhost:27017");

IMongoDatabase db = dbClient.GetDatabase("admin");

var ciudades = db.GetCollection<BsonDocument>("Ciudades");

//CREATE
BsonDocument ciudadDoc = new BsonDocument(new BsonElement("nombre", "Málaga"));
ciudadDoc.Add(new BsonElement("ubicacion", "Sur"));
ciudades.InsertOne(ciudadDoc);
```

UPDATE

```
MongoClient dbClient = new MongoClient("mongodb://localhost:27017");

IMongoDatabase db = dbClient.GetDatabase("admin");

var ciudades = db.GetCollection<BsonDocument>("Ciudades");

//UPDATE
BsonDocument updateCiudadDoc = new BsonDocument(new BsonElement("nombre", "Cádiz"));
updateCiudadDoc.Add(new BsonElement("ubicacion", "Sur"));
BsonDocument findCiudadDoc = new BsonDocument(new BsonElement("nombre", "Málaga"));
var updateDoc = ciudades.FindOneAndReplace(findCiudadDoc, updateCiudadDoc);
```

MONGO DB

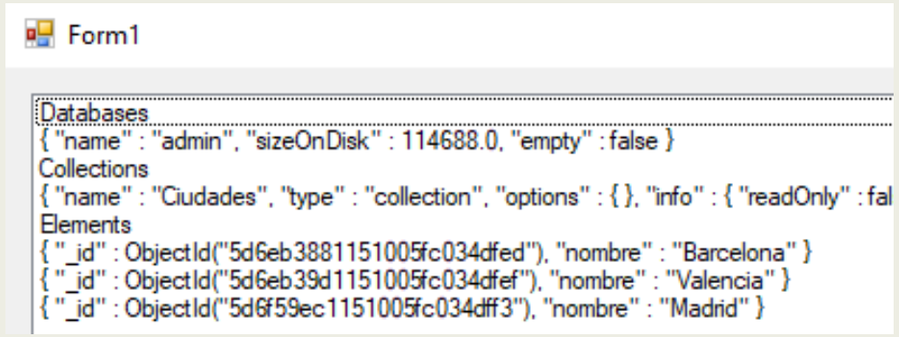
DELETE

```
MongoClient dbClient = new MongoClient("mongodb://localhost:27017");

IMongoDatabase db = dbClient.GetDatabase("admin");

var ciudades = db.GetCollection<BsonDocument>("Ciudades");

//DELETE
BsonDocument findAnotherCiudadDoc = new BsonDocument(new BsonElement("nombre", "Madrid"));
ciudades.FindOneAndDelete(findAnotherCiudadDoc);
```

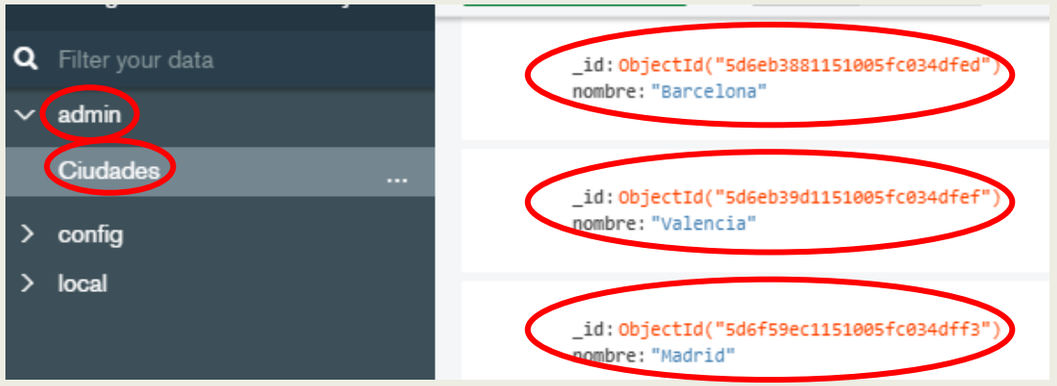
READ

```
MongoClient dbClient = new MongoClient("mongodb://localhost:27017");

IMongoDatabase db = dbClient.GetDatabase("admin");

var ciudades = db.GetCollection<BsonDocument>("Ciudades");

//READ
var dbList = dbClient.ListDatabases().ToList();
listBox1.Items.Add("Databases");

foreach (var item in dbList)
{
    listBox1.Items.Add(item.ToString());
    db = dbClient.GetDatabase(item["name"].ToString());
    var collList = db.ListCollections().ToList();
    listBox1.Items.Add("Collections");

    foreach (var item2 in collList)
    {
        listBox1.Items.Add(item2.ToString());
        var things2 = db.GetCollection<BsonDocument>(item2["name"].ToString());
        var resultDoc = things2.Find(new BsonDocument()).ToList();
        listBox1.Items.Add("Elements");

        foreach (var item3 in resultDoc)
        {
            listBox1.Items.Add(item3.ToString());
        }
        break;
    }
    break;
}
```

Salida en el listBox al recorrer la BBDD “admin” y la colección “Ciudades”.



Así se ve con Compass, el GUI de MongoDB server.



Databases
v.1.1 MARZO 2024

https://www.linkedin.com/in/ricardo-moraleda-gareta-9421099

https://www.linkedin.com/company/gdo-electric1996/

RICARDO MORALEDA GARETA

GDO SOFTWARE