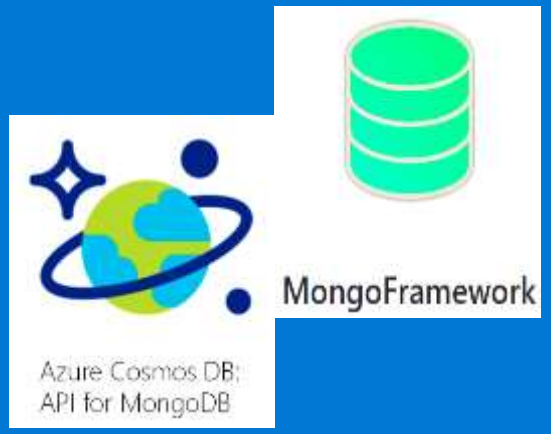
Entity Framework en base de datos NoSQL (Core +)



¿Qué son las bases de datos NoSQL?

Se puede hacer referencia a las bases de datos NoSQL indistintamente como "no relacionales", para destacar el hecho de que pueden administrar altos volúmenes de datos no estructurados que cambian con rapidez de formas diferentes a una base de datos relacional (SQL) con filas y tablas.

Características

1.Esquema flexible: No requieren un esquema fijo, lo que significa que los datos pueden variar en estructura y tipo de manera más libre.
2.Escalabilidad horizontal: Permiten una fácil distribución de datos en varios servidores, lo que facilita el manejo de grandes volúmenes de datos.
3.Alto rendimiento: la base de datos NoSQL está optimizada para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento

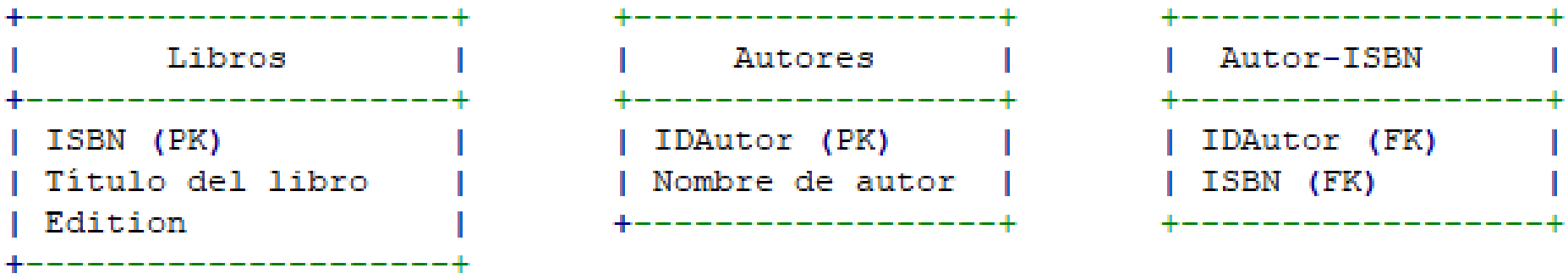
Diferencias entre base de datos NoSQL y SQL

Ejemplo de un documento NoSQL


```
{
  "books": [
    {
      "ISBN": "1234567890",
      "title": "Example 1",
      "edition": 1,
      "author": {
        "id": "A001",
        "name": "Author 1"
      }
    },
    {
      "ISBN": "0987654321",
      "title": "Example 2",
      "edition": 2,
      "author": {
        "id": "A002",
        "name": "Author 2"
      }
    }
  ]
}
```

Diferencias entre base de datos NoSQL y SQL

Ejemplo de tablas SQL



Microsoft

Azure Cosmos DB For MongoDB

¿Qué es Azure Cosmos DB para MongoDB?

Azure Cosmos DB es un servicio de base de datos de nube ofrecido por Microsoft que admite varios modelos de datos, incluido MongoDB. Permite utilizar la API de compatibilidad con MongoDB para interactuar con Azure Cosmos DB como si fuera una instancia de MongoDB.

Para utilizar Azure Cosmos DB con la API de compatibilidad con MongoDB, necesitarás crear una cuenta de Azure Cosmos DB, seleccionar la API de MongoDB al crear la cuenta y luego podrás conectar a tu cuenta de Azure Cosmos DB utilizando una cadena de conexión compatible con MongoDB. Una vez conectado, podrás realizar operaciones de lectura, escritura y consulta utilizando las API y las herramientas habituales de MongoDB.

Pasos de como creamos nuestra base de datos NoSQL desde Azure

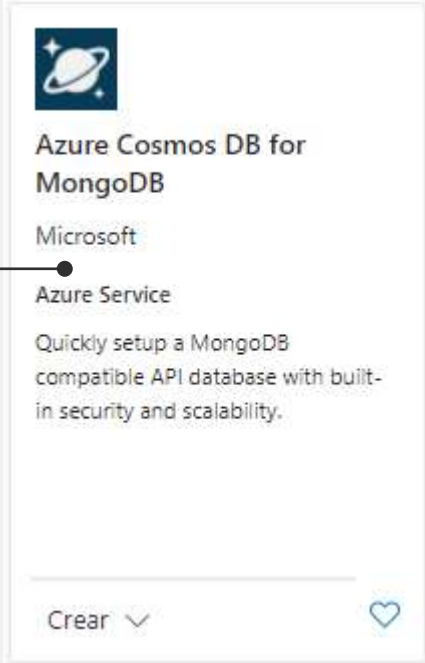
Registrarse en https://azure.microsoft.com/es-es

A

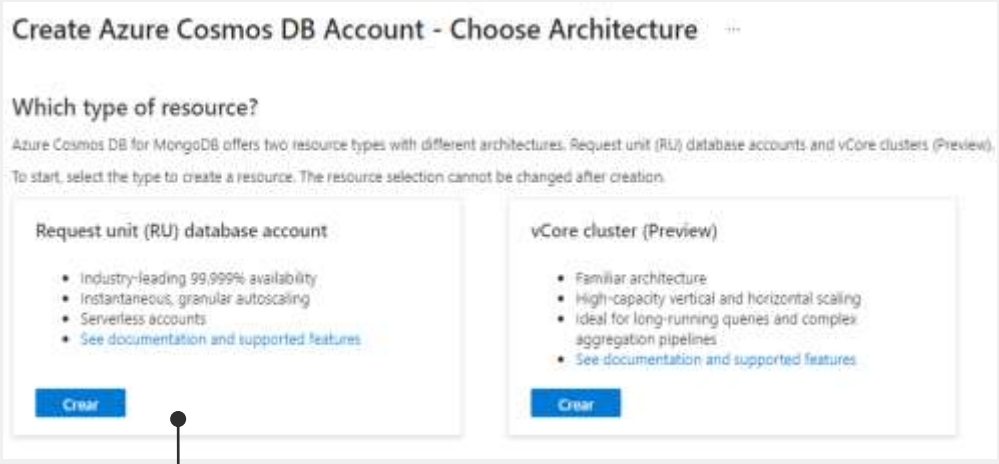
A continuación, seleccione **Crear un recurso**

B

Busque el servicio de Azure Cosmos DB for MongoDB y presione **Crear**

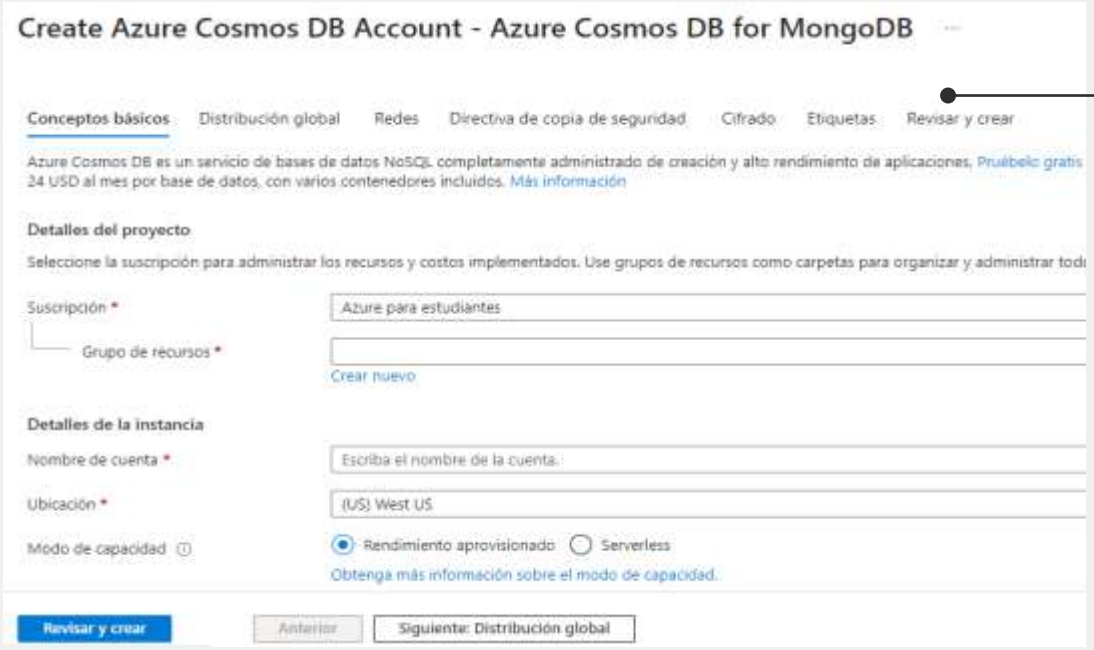


Pasos de como creamos nuestra base de datos NoSQL desde Azure



C

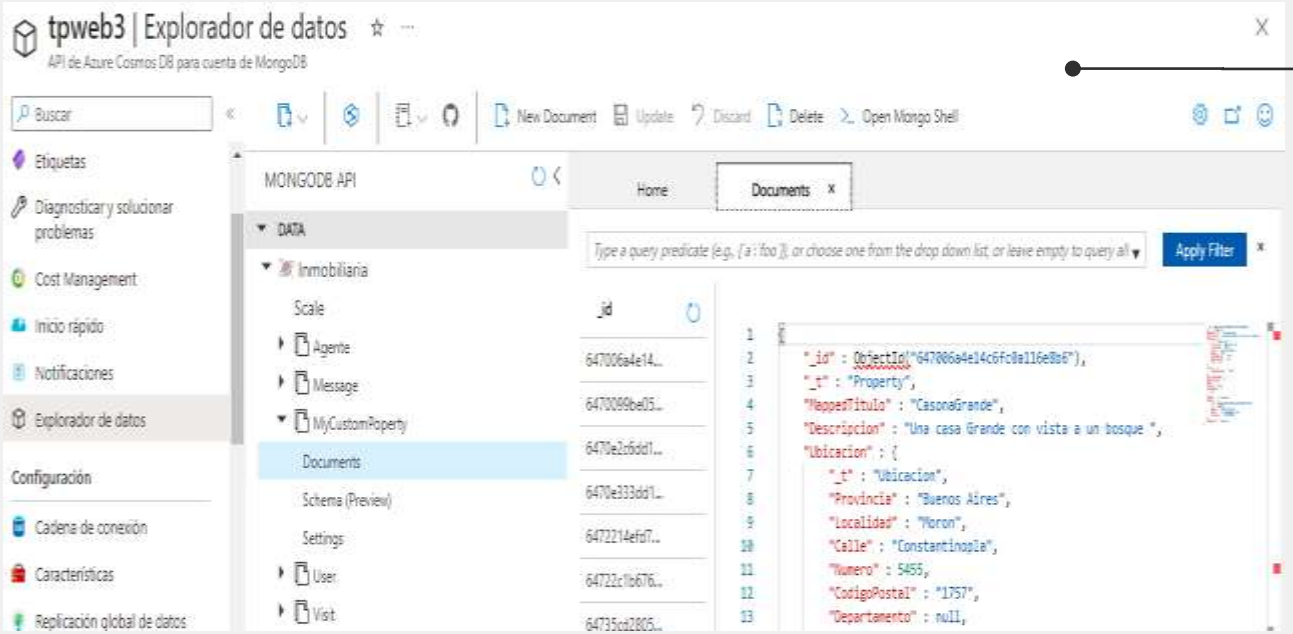
A continuación, elegimos una arquitectura, en nuestro caso la primera



D

Completamos los datos y seleccionamos **Revisar y Crear**

Pasos de como creamos nuestra base de datos NoSQL desde Azure



Vista de la base de datos

Cadena de conexión principal que utilizaremos en nuestro contexto

MongoFramework

¿Qué es y cómo funciona?

MongoFramework es una biblioteca de mapeo de objetos (Object-Document Mapper o ODM) para MongoDB en el entorno de desarrollo de .NET .Proporciona una capa de abstracción sobre la biblioteca oficial de MongoDB(Driver) para facilitar la interacción y el mapeo de objetos entre una aplicación .NET y una base de datos MongoDB.

MongoFramework intenta traer algunas de las buenas características de Entity Framework al mundo de MongoDB.

Al igual que en Entity Framework, MongoFramework basa su uso en los contextos, específicamente MongoDbContext.

La siguiente imagen nos sirve para representar como se puede crear un contexto específico para la aplicación extendiendo del MongoDbContext y como es en EntityFramework

MongoFramework

```
namespace PruebaMongo.Repository;

11 referencias
public class AppContext : MongoDbContext
{

    5 referencias
    public AppContext(IMongoDbConnection connection) : base(connection)  {}


    6 referencias
    public MongoDbSet<Property> Propiedades { get; set; }
    2 referencias
    public MongoDbSet<Agente> Agentes { get; set; }
    3 referencias
    public MongoDbSet<User> Users { get; set; }
    1 referencia
    public MongoDbSet<Contacto> Messages { get; set; }

    1 referencia
    public MongoDbSet<Visit> Visita { get; set; }
}
```

EntityFramework

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

namespace Clase7.EF.IslaDelTesoro.Data.Entidades
{
    6 referencias
    public partial class PW320231CEFIslaDelTesoroContext : DbContext
    {
        0 referencias
        public PW320231CEFIslaDelTesoroContext()
        {
        }

        0 referencias
        public PW320231CEFIslaDelTesoroContext(DbContextOptions<PW320231CEFIslaDelTesoroContext> options)
            : base(options)
        {
        }

        7 referencias
        public virtual DbSet<Tesoro> Tesoros { get; set; } = null!;
```

Conexión con la base de datos

Si bien en su mayoría se siente igual que crear contextos en Entity Framework, todavía hay una serie de diferencias, siendo la conexión la más relevante.

IMongoDbConnection es la infraestructura central que permite la conexión a MongoDB y se requiere para instanciar un contexto.

MongoFramework

```
using MongoDB.Driver;
using MongoFramework;

namespace PruebaMongo.Repository;

5 referencias
public static class ConnectionFactory
{
    5 referencias
    public static IMongoDbConnection GetConnection()
    {

        return  MongoDbConnection.FromConnectionString("mongodb://tpweb3:zrBcQovBkgZmRlyGKnB9nSBV4iDbnQNVR1QqwsaHdjDBygm7Kie8F8

    }
}
```

EntityFramework

```
0 referencias
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseSqlServer("Server=.;Database=PW3-2023-1C-EF-IslaDelTesoro;Trusted_Connection=True;");
    }
}
```