Instituto Tecnológico de Costa Rica

API MultiDataBase

C# , REST , JSON

Bases de Datos II

Integrantes

1. Allan Rojas Durán.
2. Carlos Arguello Calderon.
3. Gerardo Villalobos Villalobos

Abril, 2016

Los archivos de la aplicación estarán disponibles en :
https://github.com/aallanrd/primer-proyecto-dbII

Diseño y Arquitectura

Servicio Web

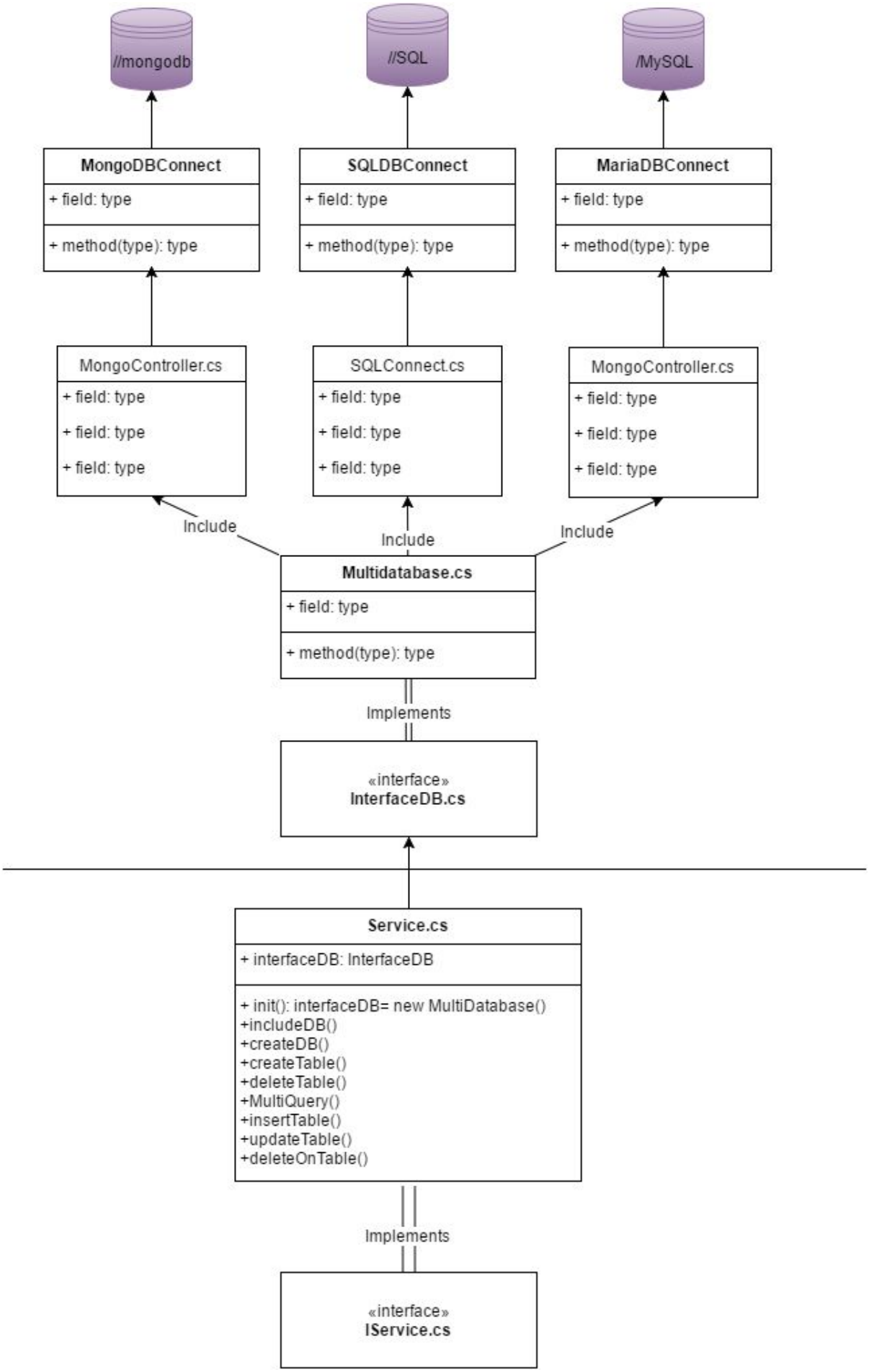
Controladores
InterfaceDB.cs: Expone el contrato de funciones para la clase multidatabase, que en si contendrá el cuerpo de las funciones del API principal.
Multidatabase.cs: InterfaceDB : Implementa la lista de funciones de la interface, creando la instancia de cada controlador para llamar a los diferentes métodos de cualquier DB del multidatabase administrator.
MariaDB.cs, MongoDB.cs,SQL Controller.cs : Crean la instancia de su proveedor respectivo, para estar realizando consultas con una sola instancia de proveedor, que se encarga de abrir y cerrar la conexión y algunas operaciones básicas, sin embargo en este controlador se realizan las operaciones más pesadas propias de nuestro sistema y además del procesamiento de sus respuestas.

Proveedores
Connect MariaDB,Connect MongoDB,Connect SQL utilizan los drivers o referencias generales de sus conexiones para poder abrir una conexión con su base de datos respectiva o con la que se le ordene establecer conexión desde el controlador.
Servicio
- IServicio
- Servicio

Proyecto Web

Service Reference
Accede por medio de discover a los servicios que estén disponibles en la maquina, como nuestro servicio web esta corriendo en la misma máquina
Controladores
- ControlApp
  - Api/Ruta

Vistas
- Control
  - VIew/Ruta


//mongodb
//SQL
/MySQL
MongoDBConnect
+ field: type
+ method(type): type
SQLDBConnect
+ field: type
+ method(type): type
MariaDBConnect
+ field: type
+ method(type): type
MongoController.cs
+ field: type
+ field: type
+ field: type
SQLConnect.cs
+ field: type
+ field: type
+ field: type
MongoController.cs
+ field: type
+ field: type
+ field: type
Include
Include
Include
Multidatabase.cs
+ field: type
+ method(type): type
Implements
«interface»
InterfaceDB.cs
Service.cs
+ interfaceDB: InterfaceDB
+ init(): interfaceDB= new MultiDatabase()
+includeDB()
+createDB()
+createTable()
+deleteTable()
+MultiQuery()
+insertTable()
+updateTable()
+deleteOnTable()
Implements
«interface»
IService.cs

Librerías utilizadas

Conectando con MariaDB .

MYSQL Connector

Desde la versión 6.7, conector / Net ya no incluirá el MySQL para la integración de Visual Studio. Esa funcionalidad ya está disponible en un producto separado llamado MySQL para Visual Studio disponible utilizando el instalador de MySQL para Windows

Creación de la Clase

Es siempre una mejor idea de crear una nueva clase de conexión a la base de datos y para separar el código real del código que tendrá acceso a la base de datos. Esto ayudará a mantener nuestro código limpio, fácil de leer y más eficiente. Vamos a empezar por la adición de la biblioteca MySQL Connector:

Ocultar código de la copia

```
// Añadir biblioteca de MySQL
 usando MySql.Data.MySqlClient;
```

Luego de declarar e inicializar las variables que vamos a utilizar:

- `conexión` : se utiliza para abrir una conexión a la base de datos.
- `servidor` : indica dónde se encuentra alojado nuestro servidor, en nuestro caso, es localhost.
- `la base de datos` : es el nombre de la base de datos vamos a utilizar, en nuestro caso se trata de la base de datos ya creada anteriormente que es `connectcsharptomysql` .
- `uid` : es nuestro nombre de usuario de MySQL.
- `password` : es nuestra contraseña de MySQL.
- `connectionString` : contiene la cadena de conexión para conectarse a la base de datos, y se le asignará a la variable de conexión.

```
## using MySQL.Data.
```

Conectando con SQL

https://msdn.microsoft.com/es-es/library/system.data.sqlclient(v=vs.110).aspx

System.Data.SqlClient: El System.Data.SqlClient espacio de nombres es el.NET Framework Data Provider para SQL Server.

Esta librería contiene estos métodos utilizados en este proyecto y son:

1- SQLConnect: Representa una conexión abierta a una base de datos de SQL server. Esta clase no puede heredarse.

Ejemplo:

-Acá se puede ver que se utilizar los parámetros de uid,pass, server, port, database para que pueda funcionar y que los demás métodos establezcan la conexión descrita.

```
public aSQLController()
        {
conexion = new SQLConnect("root","CAAC89","DESKTOP-FV57AJ9",1433,
"CAAC89");
        }
public aSQLController(string uid, string pass, string server, int port,
string database)
        {
conexion = new SQLConnect( uid,  pass,  server,  port,  database);
        }
```

2- **SqlCommand**: Representa un procedimiento almacenado o una instrucción de Transact-SQL que se ejecuta en una base de datos de SQL server. Esta clase no puede heredarse.

Ejemplo:

Este es un ejemplo se ejecuta esa cadena que redirigida a una conexión específica de un servidor “X”.

```
SqlCommand command = new SqlCommand("SELECT TOP 3 * FROM Dogs1 ORDER
BY Weight", conexión)
```

Posibles errores
Se requiere descargar la librería compatible con visual studio sql server; aunque ya vienen por defecto por ser de Microsoft.

Cadenas de Sql server funcionales

`-SELECT *FROM sys.databases :` Este comando en Sql Server sirve para consultar todas las bases de datos existentes que hay en sql server.

`-CREATE DATABASE + database_name:` Esta cadena sirve para crear una BD dentro de un servidor.

`-CREATE TABLE[dbo]. + table_name + colums:` Esta cadena sirve para crear una tabla en la base de datos donde table_name es el nombre de la base de datos y columns son la columna que encerrado dentro de 2 paréntesis “(variable tipo longitud)” y separados por coma si se desea más parámetros dentro de una tabla.

`-DROP TABLE[dbo]. + table_name:` Esta cadena sirve para borrar una tabla en la base de datos donde table_name es el nombre de la base de datos.

`-INSERT dbo+table_name+VALUES+colums:` Este cadena sirve para agregar valores a una tabla ya creada; table_name es el nombre de la tabla a utilizar y columns son las valores a crear y van separados por coma dentro del método creado por si se quiere agregar mas datos.

`-"UPDATE dbo." + table_name + "set" + columnaParametroOriginal+"="+columnParametroACT +"WHERE"+"("+llaveCondicional+"="+llaveDatoIngresado+")” :` Esta cadena actualiza una tabla donde se utiliza un parametro “X” y se utiliza una condición que debe tener el parametro de la tabla para que pueda funcionar.

`- "DELETE dbo." + table_name + "WHERE" +"("+columnaParametroOriginal+"="+columnParametroACT+")" :` Esta cadena sirve para borrar datos o valores de una tabla donde el parámetro de una “x” sea igual al que escribió el usuario.

Interfaz gráfica
-Para poder trabajar sql server se debe trabajar en la versión de Microsoft para mayor compatibilidad con Visual Studio.

Descripción

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

Autentificación

Se da de 2 formas para ingresar al sistema de gestión de datos:

1- Windows autentificación: Es una forma de ingresar sin contraseña solo con el usuario.

2-Sql Server autentificación: Es una forma de ingresar con usuario y contraseña.

Conectar a MongoDB

-- Iniciar instancia de MongoDB
https://docs.mongodb.org/manual/tutorial/manage-mongodb-processes/

```
mongod --dbpath D:/Entornos/MongoDB/srv/mongodb/ --port 27017
```

http://mongodb.github.io/mongo-csharp-driver/2.0/getting_started/quick_tour/

--Interfaz Gráfica Mongo
https://robomongo.org/download

-- Error con autenticación
Añadir Usuario :

```
db.createUser( { user:"aallanrd", pwd: "abc1234", roles : [{ role : "userAdminAnyDatabase", db: "admin"}] })
```

https://docs.mongodb.org/manual/tutorial/enable-authentication/

-- Error con librerías

```
using MongoDB.Bson;
using MongoDB.Driver;
```

```
            var credential = MongoCredential.
                  CreateMongoCRCredential("allandb", "aallanrd",
            "abc1234");

            var settings = new MongoClientSettings
            {
                Credentials = new[] { credential }
            };

             _client = new MongoClient(settings);
            var database = _client.GetDatabase("allandb");
            if (database!=null)
            {
                var collection =
database.GetCollection<BsonDocument>("persons");
           }
```

Para trabajar con MongoDB es necesario descargar el controlador MongoDB C# que soporta MongoDB.

Añadir a las referencias del proyecto a las siguientes DLL:

1. MongoDB.Bson.dll
2. MongoDB.Driver.dll
3. MongoDB.Driver.Core.dll

Añadir las siguientes instrucciones using en su programa en C#:

```
using MongoDB.Bson;
using MongoDB.Driver;
```

Trabajar con MongoDB es lo mismo que trabajar en LINQ y Entity Framework. MongoDB contiene las mismas características y el mismo estilo de trabajo que tiene LINQ. Esta es una de las grandes características de MongoDB.

Al igual que una conexión de SQL que están obligados a crear un cliente MongoDB.

Usted puede utilizar el método InsertOneAsync y InsertManyAsync para agregar documentos a una colección en MongoDB. Si intenta añadir documentos a una colección que no existe, MongoDB creará la colección para usted.

Si el documento que se pasa al método InsertOneAsync no contiene el campo _id, el controlador añade automáticamente el campo al documento y establece el valor del campo a un ObjectId generado.

Se pueden utilizar los métodos Find y FindAsync para emitir una consulta para recuperar datos de una colección en MongoDB. Todas las consultas en MongoDB tienen el alcance de una sola colección.

En las consultas se pueden devolver todos los documentos de una colección o sólo los documentos que coinciden con un filtro o criterios especificados. Se puede especificar el filtro o los criterios en un BsonDocument y transmitirlo como un parámetro a los métodos Find y FindAsync.

Usted puede utilizar los métodos UpdateOneAsync, UpdateManyAsync, y el método ReplaceOneAsync para actualizar los documentos de una colección.
Los métodos aceptan los siguientes parámetros:
-un documento de filtro para que coincida con los documentos a actualizar,
-un documento de actualización para especificar la modificación de realizar, y
-un parámetro de opciones (opcional).

Si no hay ningún documento que coincida con la condición de actualización, el comportamiento por defecto del método de actualización es no hacer nada.

Usted puede utilizar el método UpdateOneAsync y DeleteAsync para eliminar documentos de una colección. El método toma un documento de condiciones que determina los documentos a eliminar.

En MongoDB, las operaciones de escritura son atómicas en el nivel de un solo documento. Si una sola operación elimina varios documentos de una colección, la operación puede intercalarse con otras operaciones de escritura en esa colección.

MongoDB puede realizar operaciones de agregados, tales como la agrupación por una clave especificada y evaluación de un total o un recuento de cada grupo distinto.

Los índices pueden apoyar la ejecución eficiente de las consultas. Sin índices, MongoDB debe realizar una exploración de la colección, es decir escanear todos los documentos en una colección, para seleccionar aquellos que cumplan la instrucción de consulta. Si existe un índice apropiado para una consulta, MongoDB puede utilizar el índice para limitar el número de documentos que debe inspeccionar.

Características

El API debe incluir:

1. **/includeDB**

   Incluir nuevo servidor/base de datos:

   Una ruta para poder **incluir un nuevo servidor/base de datos**, esta debe retornar un mensaje si fue agregada correctamente o no.

   a. Ruta"/addDatabase"
   b. Parámetros

   {
   database_type: "MariaDB/MongoDB/SQLServer",
   user: "user",
   pass: "pass",
   server: "server",
   protocol: "tcp",
   port: "4100",
   alias: "alias"
   }

2. **/createDB**

   Crear Base de Datos

   Una ruta para crear una base de datos (en el caso de MongoDB)
   a. Ruta"/createDatabase"
   b. Parámetros { database_name: "database" }

3. **/createTable**

   Una ruta para crear tablas
   a. Ruta"/createTable"
   b. Parámetros
   { nombre: "nombre",
   columnas: [{ alias:"alias",
   nombre:"nombre",
   tipo: "tipo",
   null: true/false },...]
   c. Entre los tipos están
   i. Int
   ii. Double
   iii. String

d. Noten que cada columna puede estar en una tabla distinta

4. **/deleteTable**
   Borrar Tablas

   Una ruta para borrar una tabla de la base de datos

5. **/multipleQuery**
   Hacer Multiple Query

   Una ruta para poder hacer query a una o varias tablas

   a. . Columnas a Seleccionar

   ii. Tablas a utilizar (solo se permite Inner Join)

   1. Columnas para hacer el Join de ser necesario .
   2. Estas deben ser del mismotipo, pero no del mismo nombre

   iii. Alguna forma de filtro complejo por columna debe incluir al menos

   1. Igualdad
   2. Desigualdad
   3. Mayor
   4. Menor

   iv. Alguna forma de hacer un ordenamiento, por múltiples columnas

   v. Alguna forma de hacer una agregación

6. **/insertValuesTable**
   Insertar Valores a Tabla

   Se deben proveer al menos todas las columnas que no pueden ser nulas.

7. **/updateValuesTable**
   Actualizar Valores de Tabla

   Se debe poder especificar la fila

8. **/deleteValuesTable**
   Borrar Valores de Tabla.

9. **/getConnections**
   Obtener BD Conectadas

   Una ruta para poder obtener los alias de las bases de datos que están conectadas.

Bibliografía.

https://msdn.microsoft.com/es-es/mt429383
https://es.wikipedia.org/wiki/Microsoft_SQL_Server
https://github.com/mongodb/mongo-csharp-driver/releases