Comparativa capa de datos

SQLite – MongoDB

Así como resumen ejecutivo se obtuvo un mejor desempeño con MongoDB en toda la implementación respecto a la facilidad para desarrollar el código de las operaciones CRUD. Sin embargo, la mayor dificultad estuvo a la hora de buscar consultas muy específicas entre tablas “lejanas” una de la otra. Un problema particular apareció al manipular la tabla “Subsidios” por el tema del ObjectId que maneja Mongo por defecto, una solución fue emplear el campo ID\_Subidio que ya se había trabajado con SQLite, para ello, se puso a funcionar con un método Random que asignaba una cadena de caracteres de longitud (25) cada vez que se registraba un nuevo subsidio. A través de ese campo se hacia la conexión lógica entre tablas manipulando los modelos creados.

Siguiendo esta misma lógica era mucho más fácil referirse a registros dentro de la base de datos en muchos casos, algunas veces en SQLite tocaba implementar una consulta SQL muy especifica y extensa que a veces dificultaba el proceso, en cambio en Mongo se usaron funciones como Find, Async, Filter, Any… etc, para dar con el registro concreto y efectuar alguna operación. Adicionalmente, nos resultó más sencillo la línea de código para las operaciones, dado que ya venia implementada de forma más práctica solo requería de dos parámetros, un filter y un update, es decir, indicar los registros a modificar y luego pasarle los datos modificados.

|  |  |  |
| --- | --- | --- |
| **Operación CRUD** | **SQLite (SQL)** | **MongoDB** |
| Create | Se establece una cadena de conexión con la base de datos, luego a través de consultas SQL acompañadas de funciones como INSERT, DELETE o UPDATE se establece la operación. | InsertOne fue la función para añadir, simplemente se hacia la referencia a la tabla de inserción y usaba la función con los datos extraídos de textBox y listBox que el usuario manipula. Uso de condicionales que verificaban la existencia de registros en la base dee datos para evitar duplicados. |
| Read | A través de listas llamadas con consultas SQL se genera una estructura de llamada a partir de listBox y textBox, donde obtienes datos específicos a través de una selección del usuario. Las tablas conectadas por consultas. | Se usaron varias funciones de búsqueda para mostrar en textbox la información correspondiente a un registro seleccionado en un listBox, se usaron muchos .Find para ello, esto incluyo uso de muchos de los atributos de los modelos. |
| Update | Se establece una cadena de conexión con la base de datos, luego a través de consultas SQL acompañadas de funciones como INSERT, DELETE o UPDATE se establece la operación. | UpdateOneAsync, Update y UpdateMany fueron la clave. Muy sencillo, se le indicaba un filtro de búsqueda y luego un update con los nuevos datos. Se establecieron condicionales para actualizar todos los registros similares compartidos por varias tablas. |
| Delete | Se establece una cadena de conexión con la base de datos, luego a través de consultas SQL acompañadas de funciones como INSERT, DELETE o UPDATE se establece la operación. | DeleteOne fue la función para eliminar, simplemente se hacia la referencia a la tabla de inserción y usaba la función con los datos extraídos de textBox y listBox que el usuario manipula. Además, se usaron condicionales para impedir eliminar registros cuyo valor estuviera incluido en otras tablas. |

Siguiendo lo racional es mas sencillo el tema del SQL por el tema de las llaves primarias y foráneas, siendo mucho más fácil hallar registros específicos entre tablas. Pero en lo personal se disfruto mas el uso de Mongo para este dominio de problema, por la facilidad para ejecutar algunas operaciones CRUD, el tipo de conexión con la base de datos y el manejo del ObjectID para algunos procedimientos.