Biblioteca

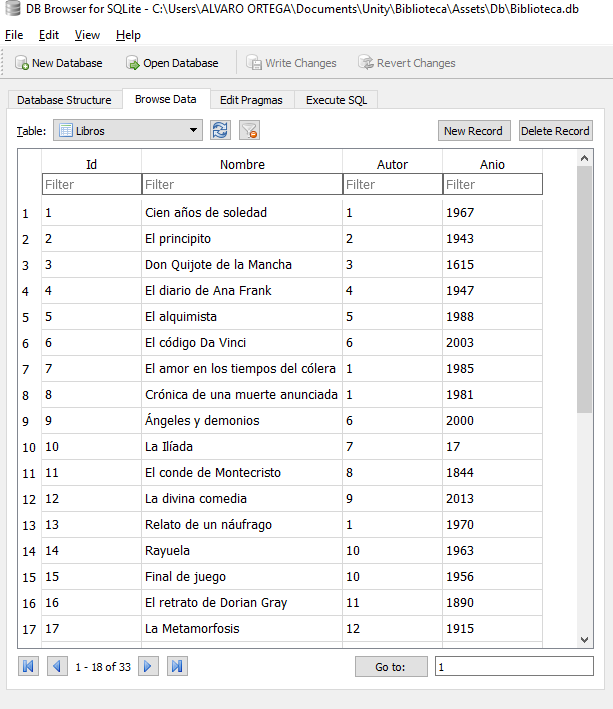
El programa consiste en un pequeño sistema de sugerencias de libros en donde se actualiza y se muestran diferentes sugerencias, o se actualiza individualmente, usando programación reactiva.

Se utilizó:

* Unity3D (Motor gráfico para videojuegos)
* UniRx (Plugin para la programación reactiva)
* SQlite (Plugin para manejar la base de datos)
* Biblioteca.db (Bases de datos local)
* C# (Lenguaje de programación)

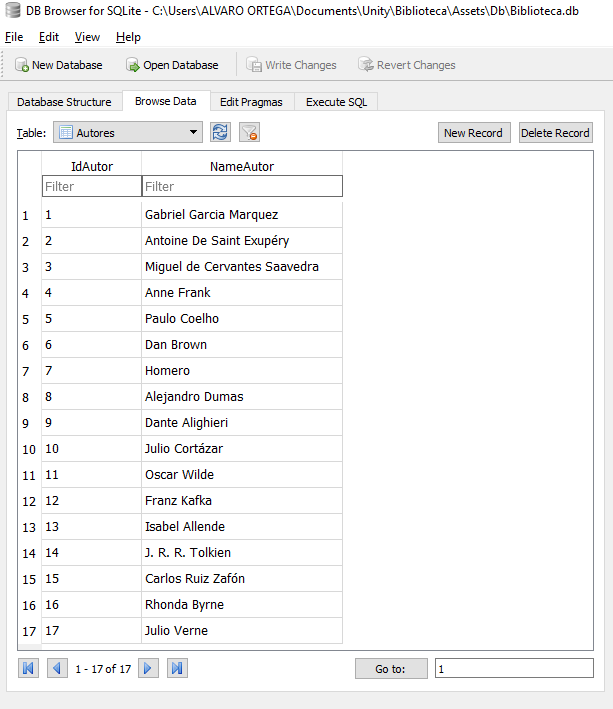
**Base de datos:**  Se creó una base de datos local con información correspondiente a los libros organizando esa información en dos tablas, Libros y Autores.

Libros: Información de publicación del libro.



Donde en el campo autor es una llave foránea de la tabla Autores

Autores: Con nombres de los autores.

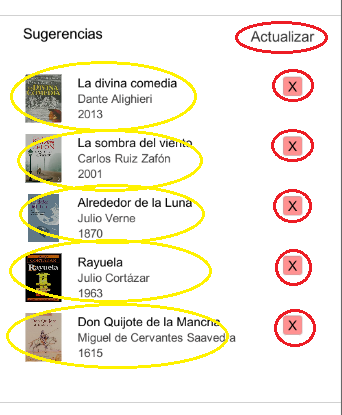


Manejando la base de datos:

1. void sugerir(){
2. conn = "URI=file:" + Application.dataPath + path;
3. dbconn = new SqliteConnection (conn);
4. dbconn.Open ();
5. IDbCommand dbcmd = dbconn.CreateCommand ();
6. string comando="select \* from Libros";
7. dbcmd.CommandText = comando;
8. IDataReader lector = dbcmd.ExecuteReader ();
9. float posiciony = posicionInicial;
10. int num = 0;
11. while (lector.Read() && num<5) {
12. InfoLibros.Add (guardar (lector));
13. }
14. // (continuando código…)
15. Si quieres ver el código completo entra a [Biblioteca](https://github.com/Alvarhito/Biblioteca)/[Assets](https://github.com/Alvarhito/Biblioteca/tree/master/Assets)/[Scripts](https://github.com/Alvarhito/Biblioteca/tree/master/Assets/Scripts)/**Sugerencias.cs**
16. }

Como vemos en el código, nos conectamos a la base de datos con “SqliteConnection”, y hacemos una consulta, buscando la información de todos los libros.

**Interfaz de usuario:** Esto es lo que se muestra al usuario, en los cuales hay botones (señalados en rojo), y son en los que aplicaremos la programación reactiva, y los utilizaremos como **observables**, que serán observado por los **observadores**, que en este caso son los objetos de libro (señalados en amarillo).



Aplicándola de la siguiente manera:

1. using UniRx;
2. .
3. .
4. .
5. void Start () {
6. bactualiza.OnClickAsObservable ().Subscribe (\_ => {
7. actualizarTodos();
8. });
9. sugerir ();
10. }
11. // (continuando código…)
12. Si quieres ver el código completo entra a [Biblioteca](https://github.com/Alvarhito/Biblioteca)/[Assets](https://github.com/Alvarhito/Biblioteca/tree/master/Assets)/[Scripts](https://github.com/Alvarhito/Biblioteca/tree/master/Assets/Scripts)/**Sugerencias.cs**

La variable **bactualiza** es la referencia al botón de actualizar. Con la librería de unity3D para la programación reactiva (UniRx), añadimos el botón el clic del actualizar como observable, y utilizo el operador “Subscribe”, para subscribir el observable a una función que actualizará los 5 libros sugeridos.

El anterior código solo era para el botón se actualizar. El siguiente código muestra algo parecido pero ahora para las “x”s de cada libro, para actualizarlo individualmente:

1. void Start () {
2. .
3. .
4. .
5. eliminar.OnClickAsObservable ().Subscribe (\_ => {
6. elimina ();
7. });
8. }
9. // (continuando código…)
10. Si quieres ver el código completo entra a [Biblioteca](https://github.com/Alvarhito/Biblioteca)/[Assets](https://github.com/Alvarhito/Biblioteca/tree/master/Assets)/[Scripts](https://github.com/Alvarhito/Biblioteca/tree/master/Assets/Scripts)/**Libro.cs**

En la clase “Libro”, hacemos cada botón de eliminar como observable, y se subscribe a una función que actualiza solo el libro en que fue clicleado.