

Documentación del Proyecto de Bases de datos Embebidas con SQLite

Módulo: Acceso a datos (AED)



Por Ayoze Amaro Estévez

IES Domingo Pérez Minik

Departamento de Informática

2º CFGS Desarrollo de Aplicaciones Multiplataforma

10 de Febrero de 2020

ÍNDICE

1.	<u>¿Qué es SQLite?</u>	1
2.	<u>¿Qué es DB Browser?</u>	3
2.1.	<u>Descargar el DB Browser</u>	4
2.2.	<u>Instalación del DB Browser</u>	6
3.	<u>Creación de una Base de Datos</u>	10
3.1.	<u>Insertar/Modificar/Eliminar registros</u>	15
3.2.	<u>Agregar claves foráneas</u>	17
4.	<u>Librerías o dependencias para el IDE</u>	18
5.	<u>Realizar la conexión con la BD</u>	19
6.	<u>Funcionamiento del CRUD</u>	20
7.	<u>Bibliografía</u>	27

1 ¿QUE ES SQLITE?

Se trata de un **sistema de gestión de bases de datos relacional**, con la intención de ser minimalista. **Está escrito en C** y es un proyecto de **dominio público**, por lo que se puede utilizar libremente.

Una de sus diferencias principales frente a otros sistemas de gestión de base de datos cliente-servidor, es que su motor no es un proceso independiente con el que el programa principal se comunica. En su lugar la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza **la funcionalidad de SQLite a través de llamadas simples a subrutinas y funcionales**. Esto hace que se reduzca la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos, esto es, las definiciones, tablas, índices y los propios datos, son guardados como un solo fichero estándar en la máquina host anfitrión. Esto se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En la versión 3 de SQLite se permiten base de datos de hasta 2 Terabytes, además de permitir campos tipo BLOB. **Mozilla Firefox, Clementine, Skype o XBMC**, por mencionar algunos proyectos, utilizan **este gestor de base de datos**.



Características sobre SQLite

La biblioteca implementa la mayor parte del **estándar SQL-92**, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, y durabilidad (ACID), triggers y la mayor parte de las consultas complejas.

SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede **insertar un string en una columna de tipo entero** (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). Algunos usuarios consideran esto como una innovación que hace que la base de datos sea mucho más útil, sobre todo al ser utilizada desde un lenguaje de scripting de tipos dinámicos. Otros usuarios lo ven como un gran inconveniente, ya que la técnica no es portable a otras bases de datos SQL. **SQLite no trataba de transformar los datos al tipo de la columna hasta la versión 3.**

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura solo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, **el acceso de escritura falla devolviendo un código de error** (o puede automáticamente reintentarse hasta que expira un tiempo de expiración configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales. Sin embargo, podría producirse un interbloqueo debido al multihilo. Este punto fue tratado en **la versión 3.3.4**, desarrollada el 11 de febrero de 2006.

Existe un programa independiente de nombre sqlite que puede ser utilizado para consultar y gestionar los ficheros de base de datos SQLite. También sirve como ejemplo para **la escritura de aplicaciones utilizando la biblioteca SQLite.**

2 ¿QUE ES DB BROWSER?

DB Browser for SQLite es una aplicación gratuita y de código abierto diseñada para facilitar la creación y administración de las bases de datos con **SQLite**. Mientras que para poder trabajar con estas bases de datos es necesario aprenderse una gran cantidad de comandos SQL, aumentando la probabilidad de que algo salga mal y hagamos que nuestra base de datos deje de funcionar correctamente.

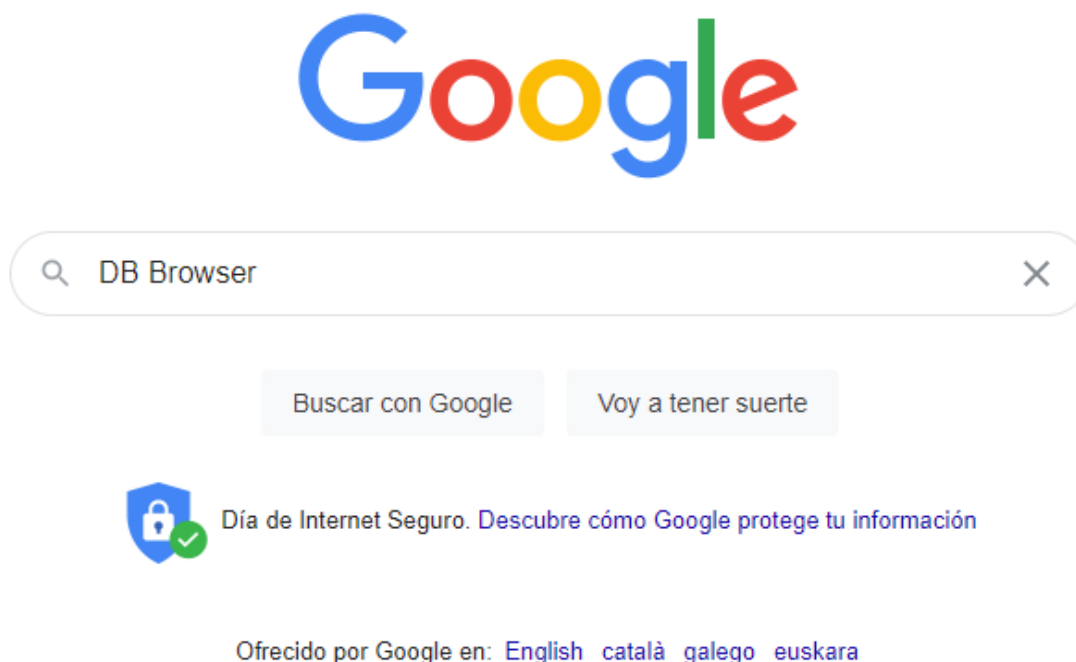
Esta aplicación (llamada inicialmente **Database Browser for SQLite**) nació como una alternativa al software Arca Database Xtra, una herramienta comercial para **facilitar la creación y edición de las bases de datos SQL**. Esta herramienta nació en 2012 preparada para trabajar con las bases de datos **SQLite 2.x**, sin embargo, a medida que ha ido pasando el tiempo se ha convertido en una de las herramientas imprescindibles cuando trabajamos con bases de datos.

DB Browser for SQLite funciona con una interfaz muy clara y sencilla de utilizar, similar basada en tablas como las que podemos encontrar en Excel de manera que tanto usuarios sin mucha experiencia en la creación y administración de bases de datos, como los desarrolladores más avanzados puedan trabajar cómodamente con sus bases de datos.



2.1 DESCARGAR EL DB BROWSER FOR SQLITE

A continuación, procederemos a descargar el **DB Browser for SQLite** ya que nos ofrece unas funcionalidades con las cuáles podremos crear nuestras bases de datos de una manera más sencilla teniendo para ello una interfaz gráfica pulida y cuidada. Comenzamos abriendo nuestro navegador web y escribiendo “**DB Browser**” en nuestro buscador predeterminado.



Tras su búsqueda en este caso en Google nos mostrará muchísimas páginas con información acerca de esta aplicación completamente gratuita de la cual haremos uso para gestionar nuestra base de datos **SQLite**. No obstante, retomando el tema seleccionaremos la primera página que se muestra (<https://sqlitebrowser.org>)

<https://sqlitebrowser.org> ▾ Traducir esta página

DB Browser for SQLite

What it is. **DB Browser** for SQLite (DB4S) is a high quality, visual, open source tool to create, design, and edit database files compatible with ...

Downloads

Note - If for any reason the standard Windows release does ...

Version 3.12.1 released

Version 3.12.1 released. 2020-11-09. This is the first bug fix ...

[Más resultados de sqlitebrowser.org »](#)

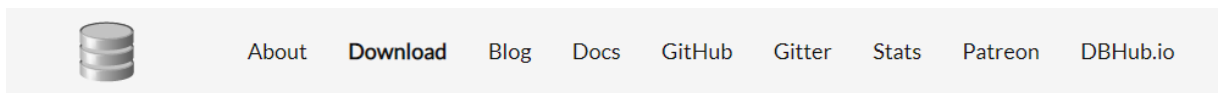
Version 3.12.0 released

SQLite 3.25.0 added support for renaming columns with the ...

About

The original version was used as a free companion tool to the Arca ...

Tras introducirnos en la página web del DB Browser tendremos una serie de pestañas con información acerca de la aplicación y con la opción de descarga para nuestro PC. Tras situarnos allí solo deberemos escoger nuestro sistema operativo y seguir los pasos que se nos proporcionan.



Downloads

([Please consider sponsoring us on Patreon](#) 😊)

Windows

Our latest release (3.12.1) for Windows:

- [DB Browser for SQLite - Standard installer for 32-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 32-bit Windows](#)
- [DB Browser for SQLite - Standard installer for 64-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 64-bit Windows](#)

Windows PortableApp

There is a PortableApp available, but it's still the previous (3.12.0) release version. It should be updated to 3.12.1 over the next few days:

- [DB Browser for SQLite - PortableApp](#)

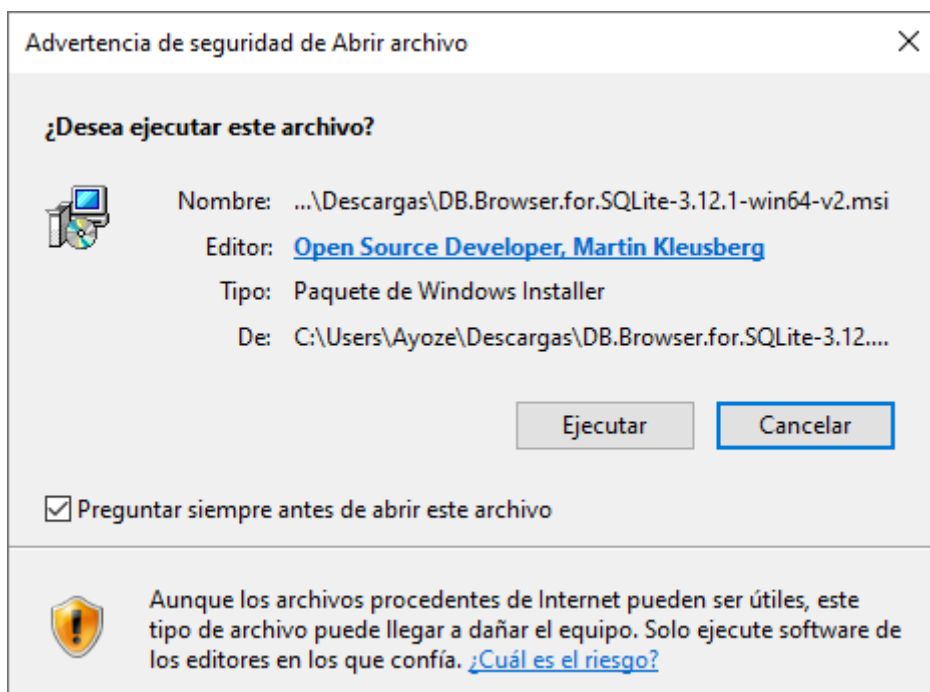
Note - If for any reason the standard Windows release does not work (e.g. gives an error), try a nightly build ([below](#)).

Nightly builds often fix bugs reported after the last release. 😊

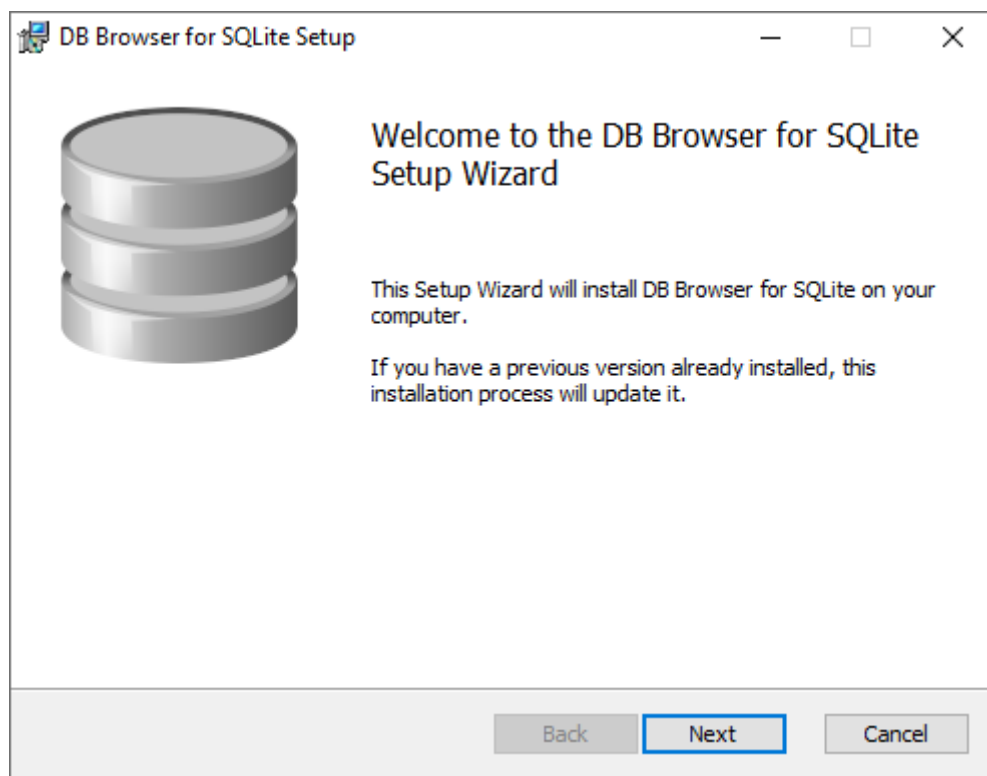
En este caso, el ordenador dispone de Windows y tiene una arquitectura de 64-bit. Por ello, optaremos por descargar el **“DB Browser for SQLite - Standard installer for 64-bit Windows”**. Tras darle click nos los descargará automáticamente y ya podremos proceder a su instalación.

2.2 INSTALACIÓN DEL DB BROWSER

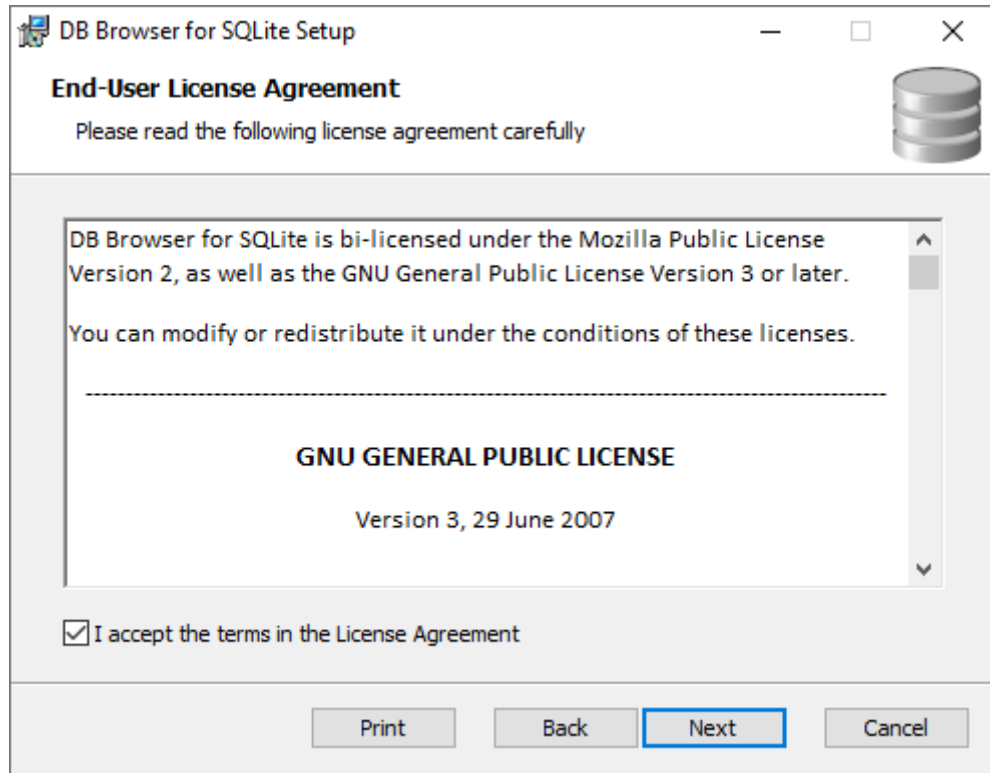
Tras la descarga del archivo que hemos realizado en el paso anterior. Procederemos a ejecutarlo y a seguir los pasos que veremos a continuación mediante imágenes.



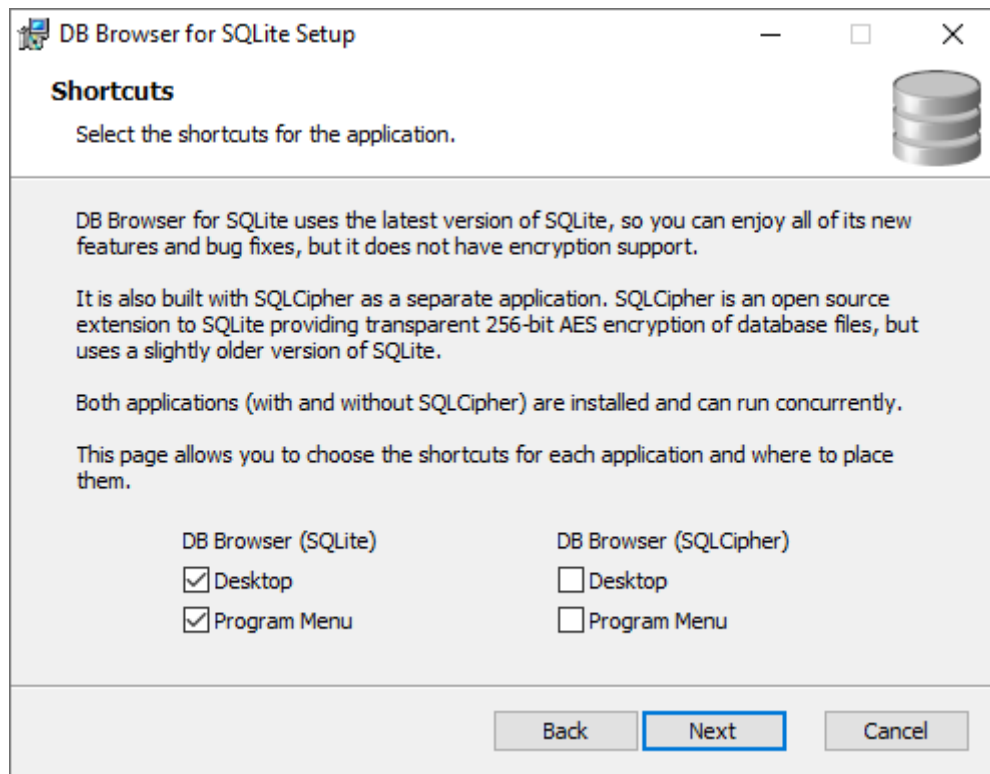
Paso 1: Apretaremos “next” tras leer el asistente de instalación del DB Browser



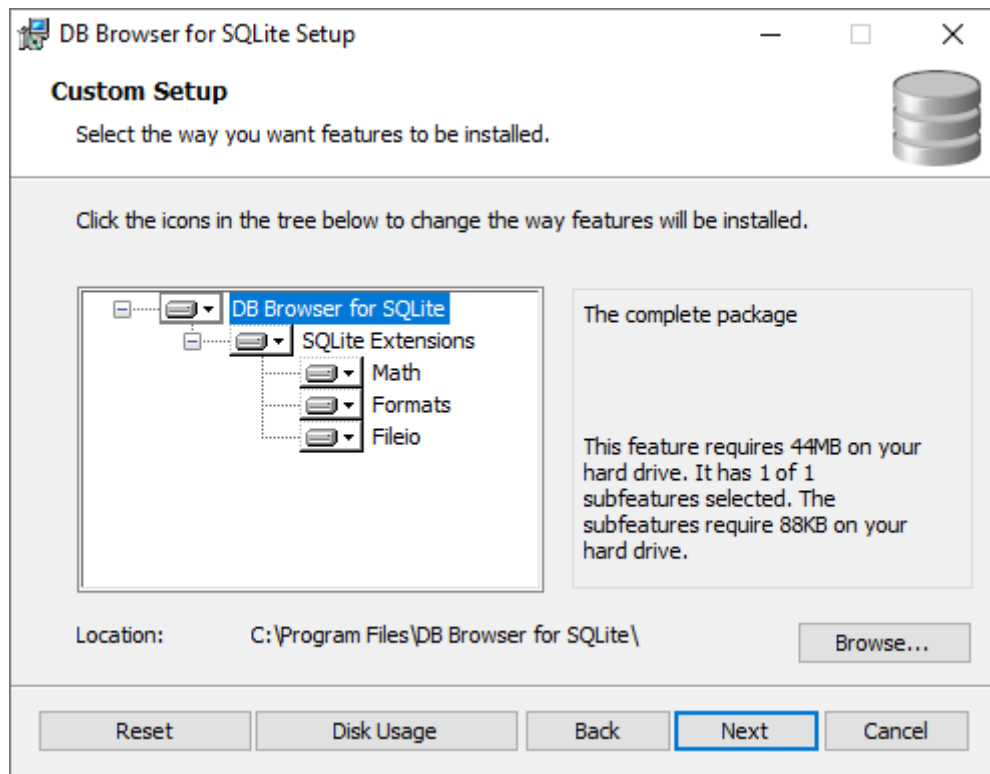
Paso 2: Aceptaremos los términos de la licencia



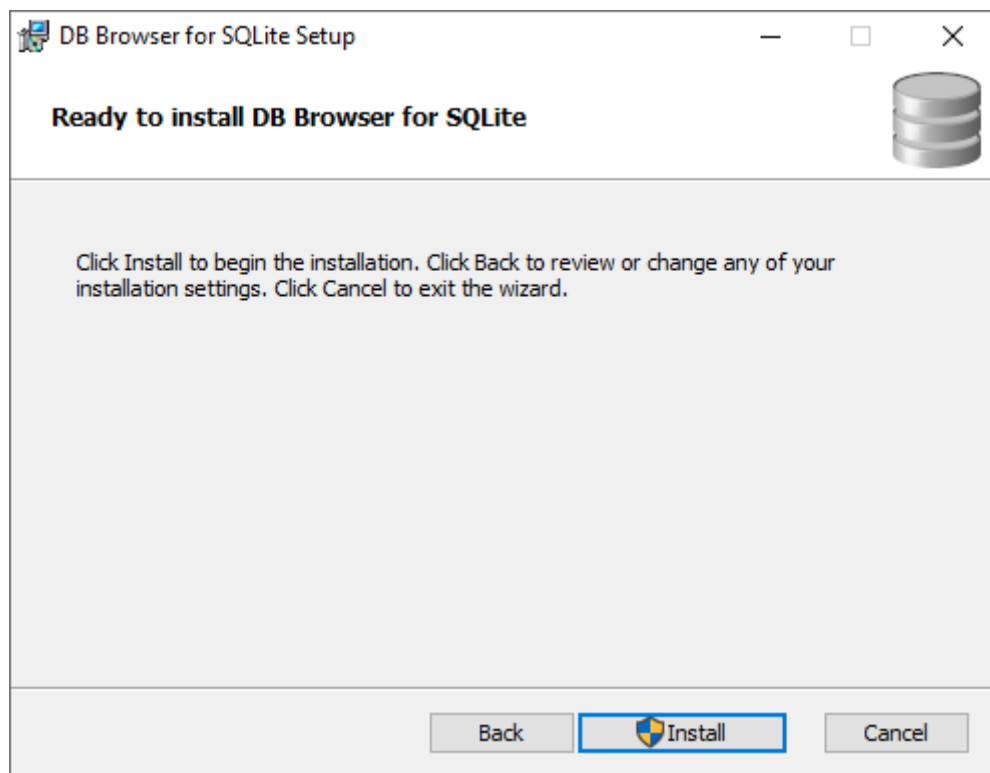
Paso 3: Seleccionaremos las opciones disponibles para SQLite



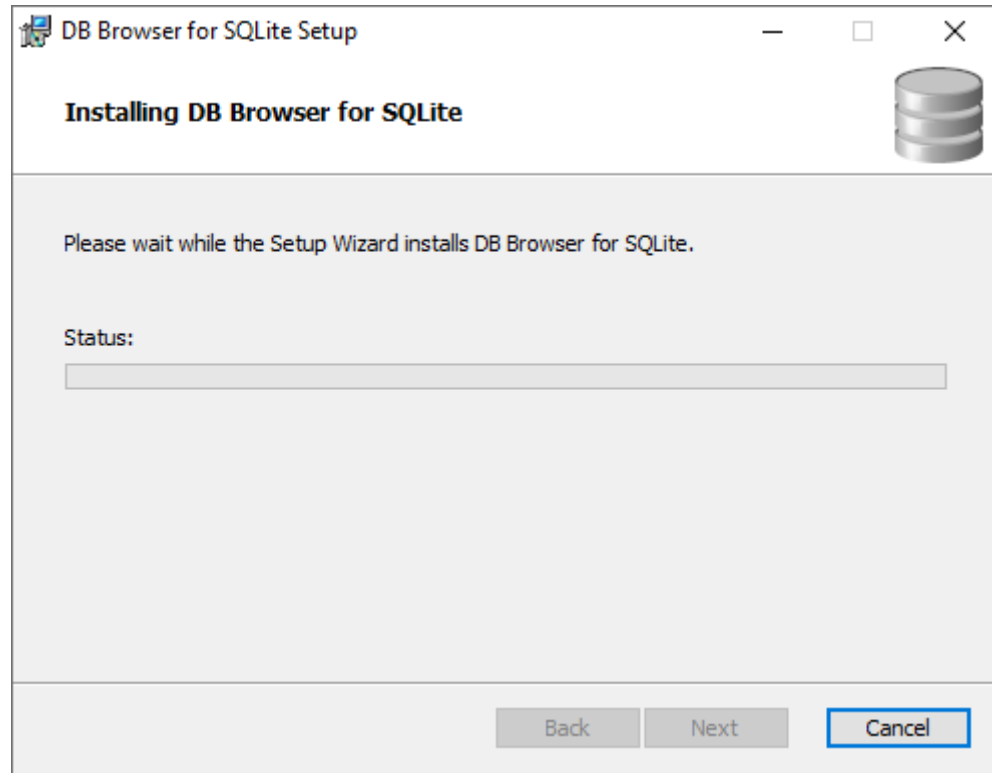
Paso 4: Seleccionaremos la carpeta de instalación



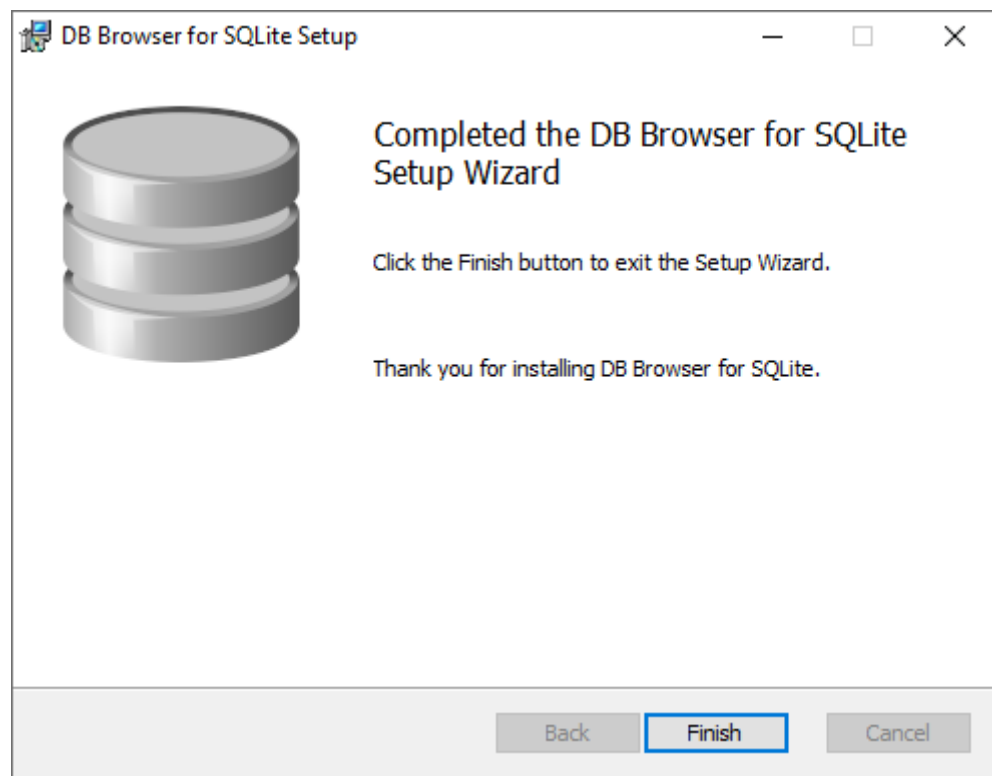
Paso 5: Apretaremos a “Install” y se iniciará la instalación



Paso 6: Esperaremos a que se instale todo correctamente

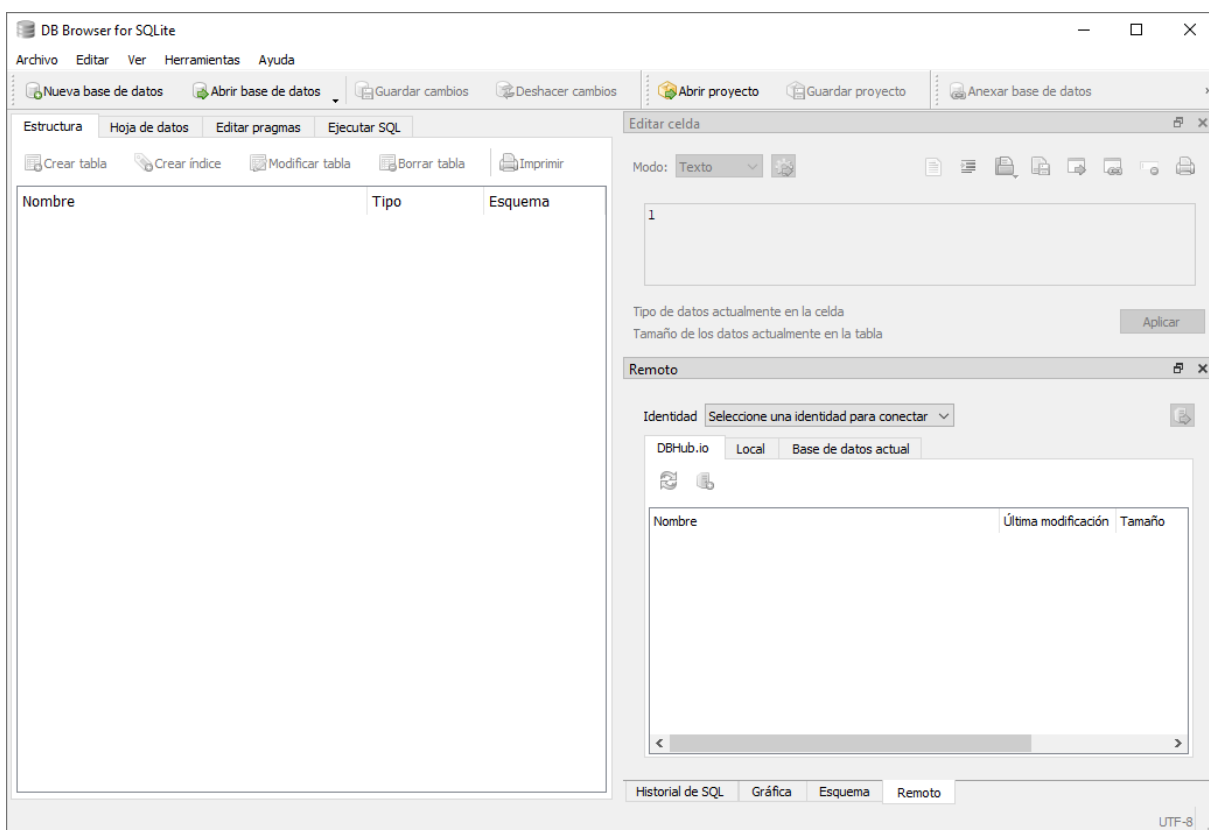


Paso 7: Finalización de la instalación

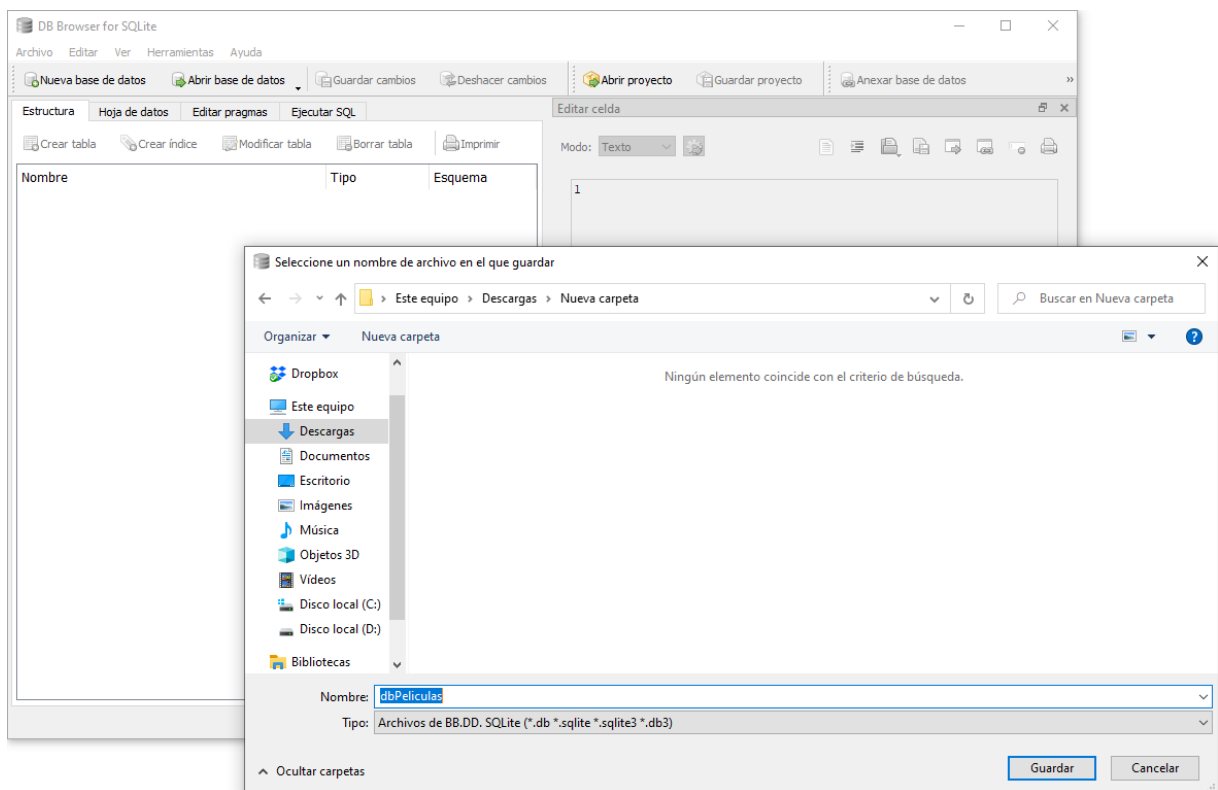


3 CREACIÓN DE LA BASE DE DATOS

Tras iniciar el DB Browser, vemos como tiene una interfaz cuidada y llamativa a la vista del usuario donde podrá gestionar las bases de datos SQLite de una manera cómoda y sencilla. En esta imagen, se nos pone la opción de la Creación de una nueva base de datos en la esquina superior izquierda donde nos ilustran un botón llamado **“Nueva base de datos”**

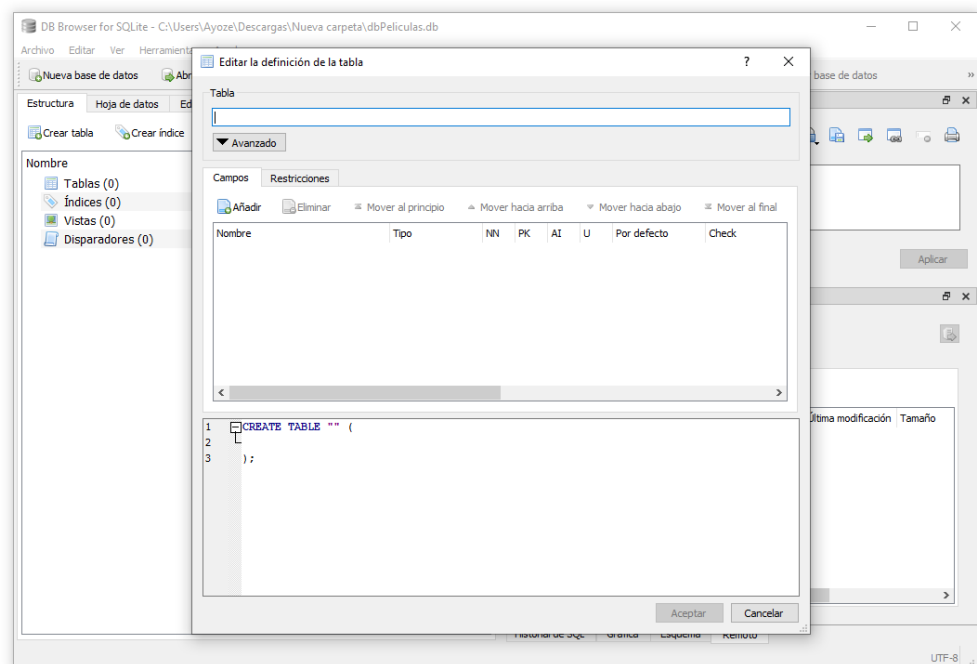


Tras pulsar este botón se nos mostrará una ventana con el explorador de archivos para ponerle un nombre a nuestra base de datos y así empezar a trabajar sobre ella. Para ello, solo debemos ver la siguiente captura donde está el explorador de archivos abierto y de nombre le pondremos **“dbPelículas”** que se guardará con una extensión **.db**

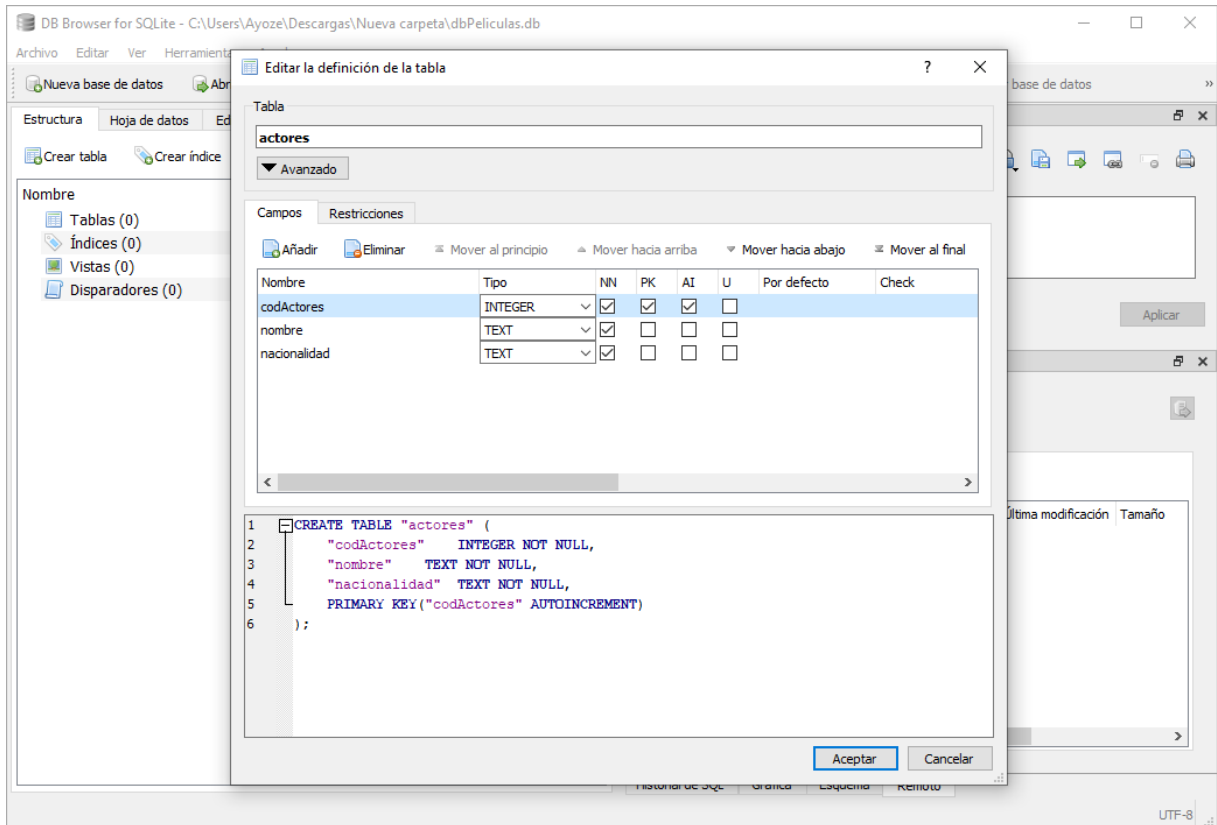


En la siguiente captura podemos ver como tras crear la base de datos “**dbPelículas**” se nos ha creado este archivo junto con otro temporal que está a la espera del guardado de cambios.

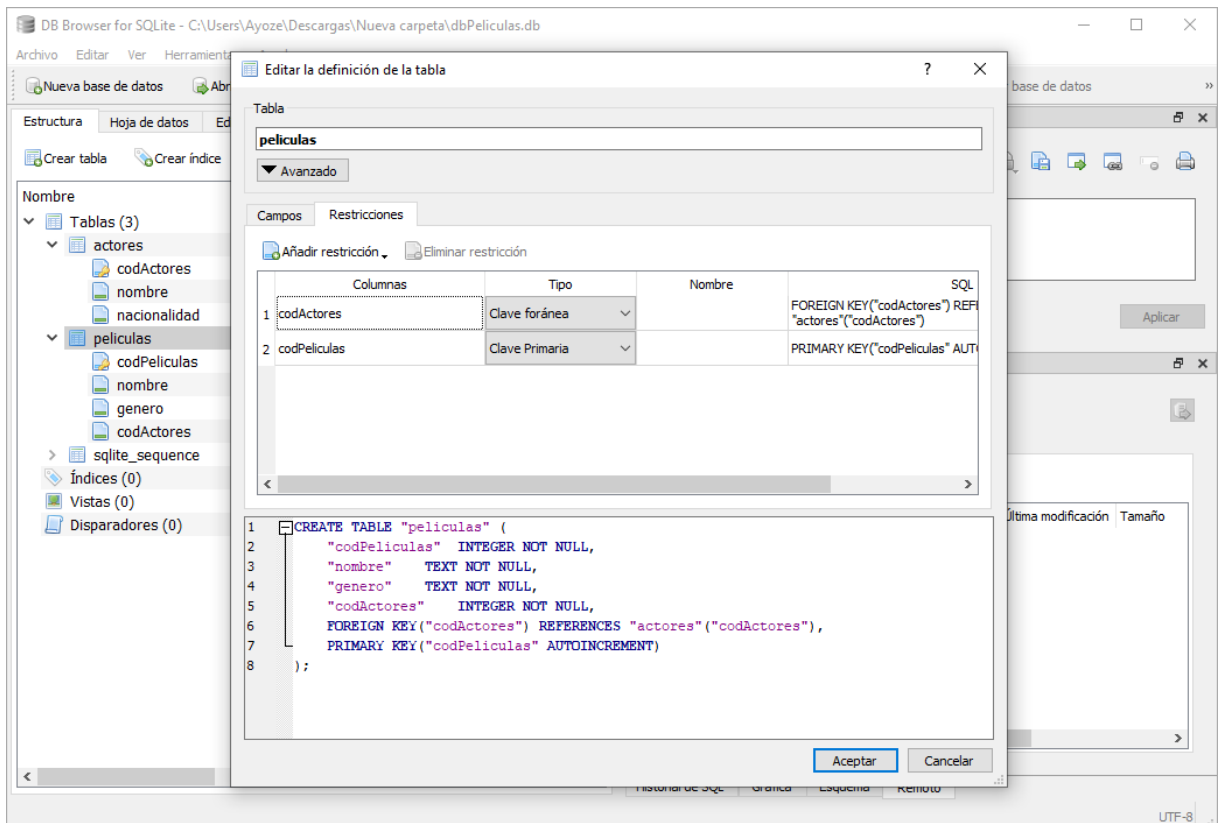
Paso 1: Creación de las tablas



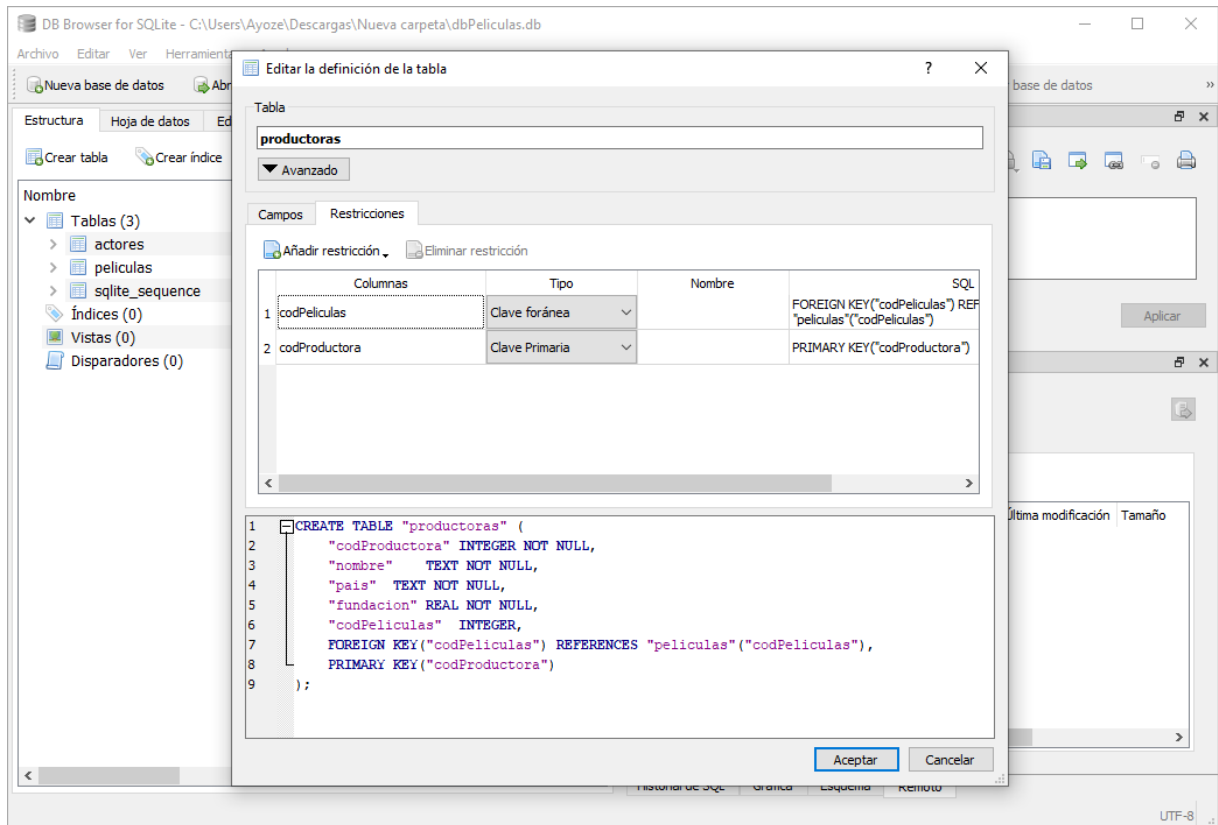
Paso 2: Añadimos los campos de la primera tabla “actores”



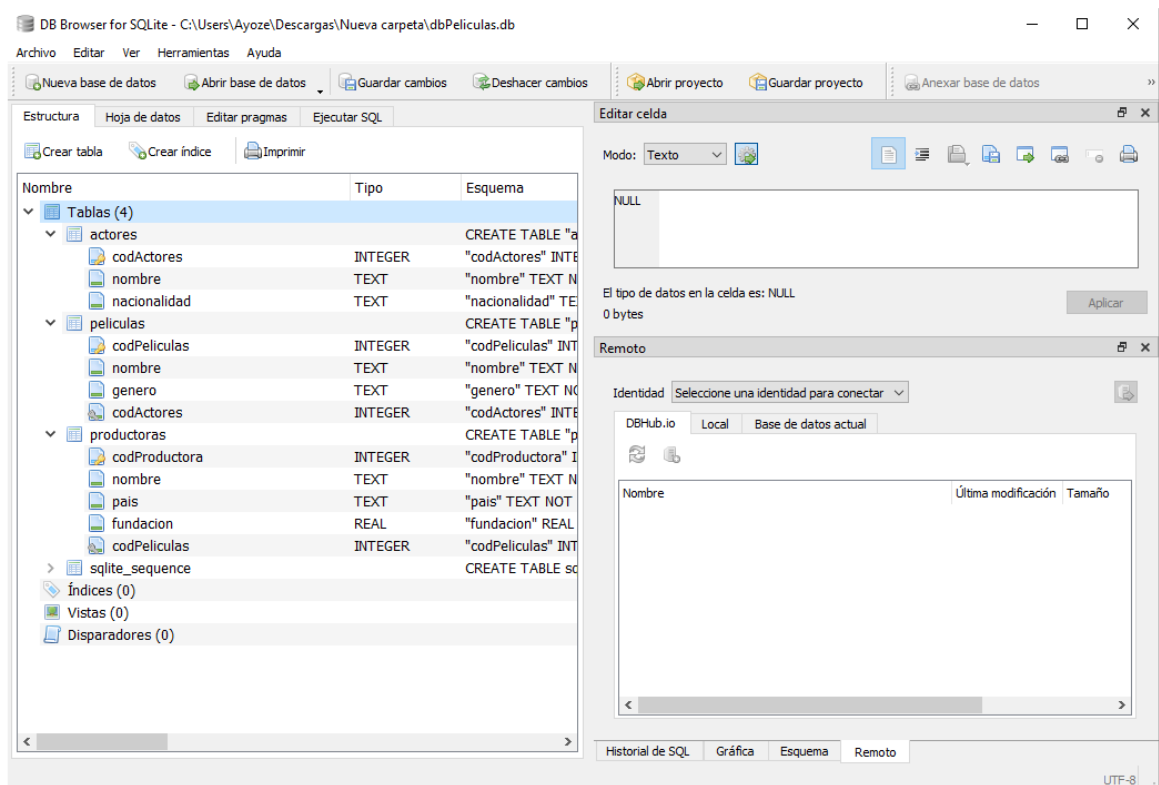
Paso 3: Añadimos los campos de la segunda tabla “películas”. Además de las claves foráneas necesarias



Paso 4: Añadimos los campos de la tercera tabla “productoras”



Paso 5: Añadimos los campos de la tercera tabla “productoras”

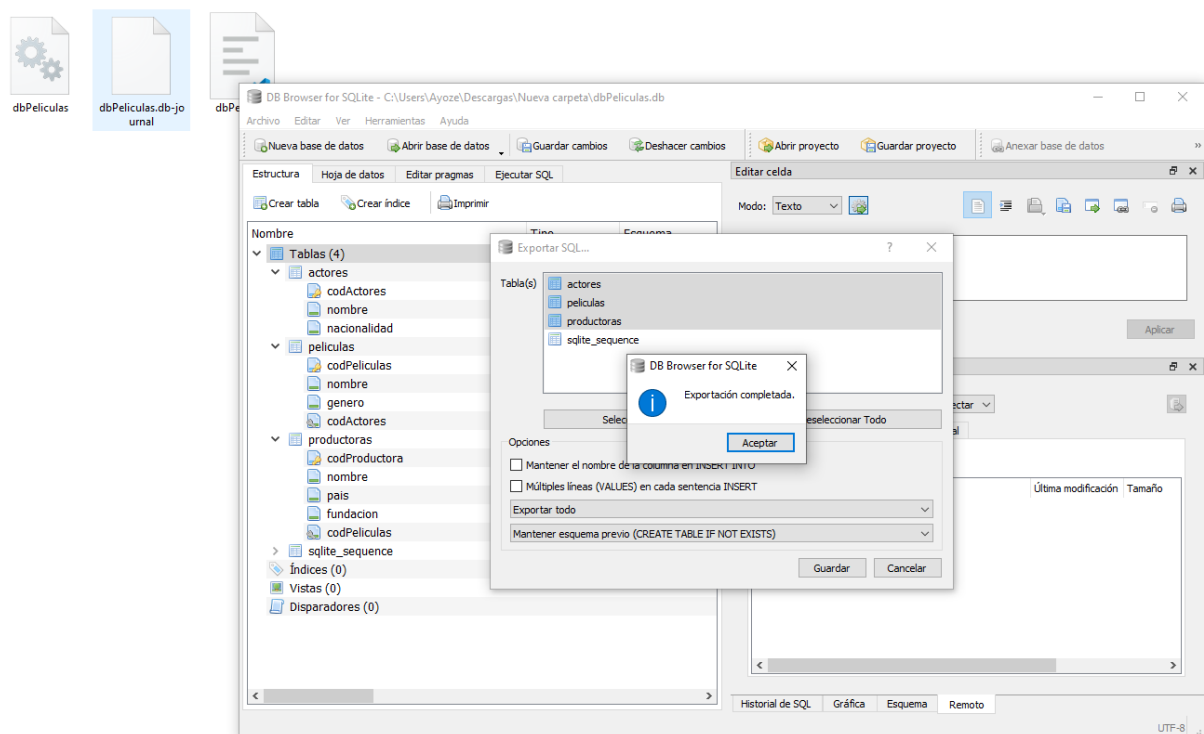


Exportar nuestra base de datos

El DB Browser tiene una serie de opciones las cuáles no pasan para nada desapercibidas. Una de ellas es la exportación de la base de datos a **.SQL**. Este script podrá ser leído por cualquier gestor de bases de datos que trabaje con **SQL** lo que implica una ayuda al usuario ya que crear las bases de datos en **DB Browser** resulta más sencillo que hacerlo mediante comandos con **SQLite** o en su defecto grandes gestores como **MySQL**, **SQL Server**, etc.

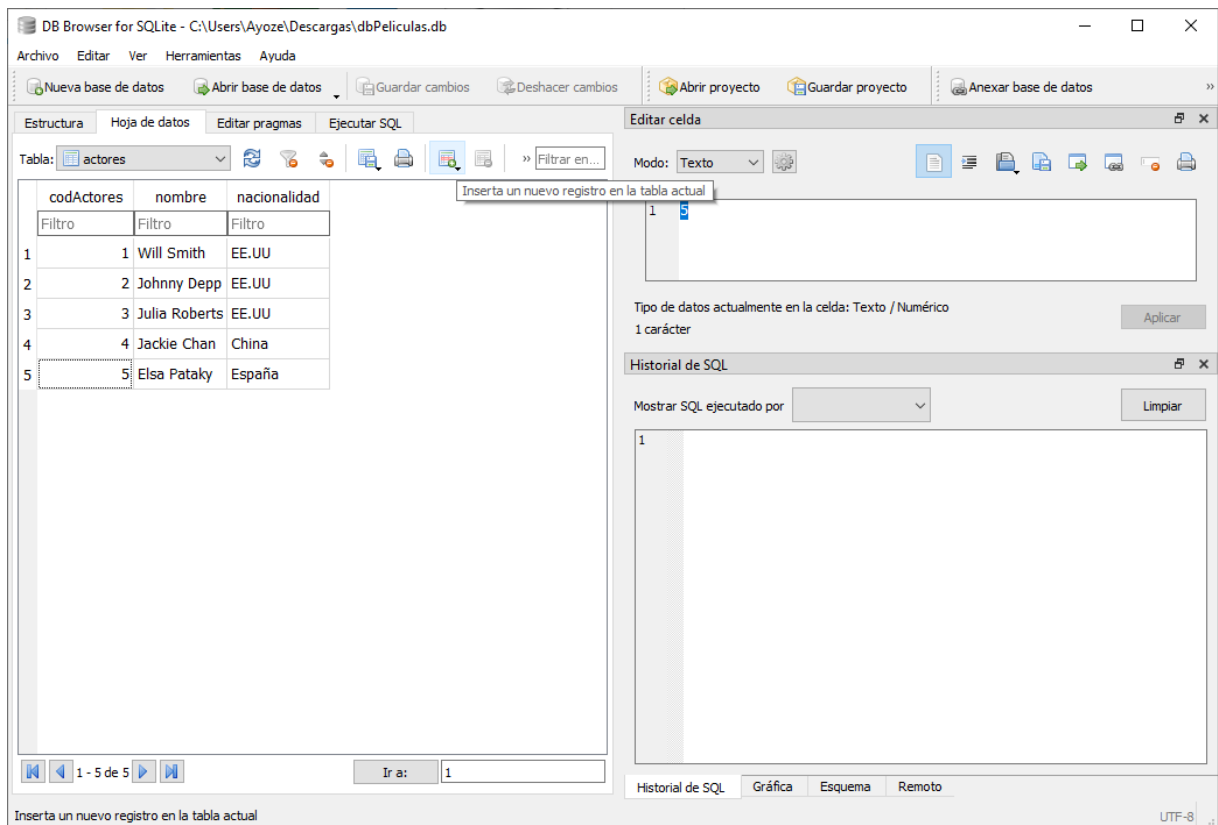
Esta exportación se haría de la siguiente manera. Comenzando por irnos al menú del programa **“Archivo”** desde aquí nos aldrán muchas opciones aunque nosotros solo necesitaremos la de **“Exportar”**. Tras pulsar sobre ella, seguiremos como se muestra en las siguientes imágenes.

Seleccionamos las tablas que correspondan y confirmaremos la exportación de la base de datos



3.1 INSERTAR/MODIFICAR/ELIMINAR REGISTROS

Para la insercción de registros, deberemos irnos a la pestaña “**Hoja de datos**” allí nos mostrará todos aquellos resgitros disponibles en la base de datos y para agregarlos solo deberemos seleccionar la tabla y agregar un nuevo registro en el botón “**Inserta un nuevo registro en la tabla actual**”.

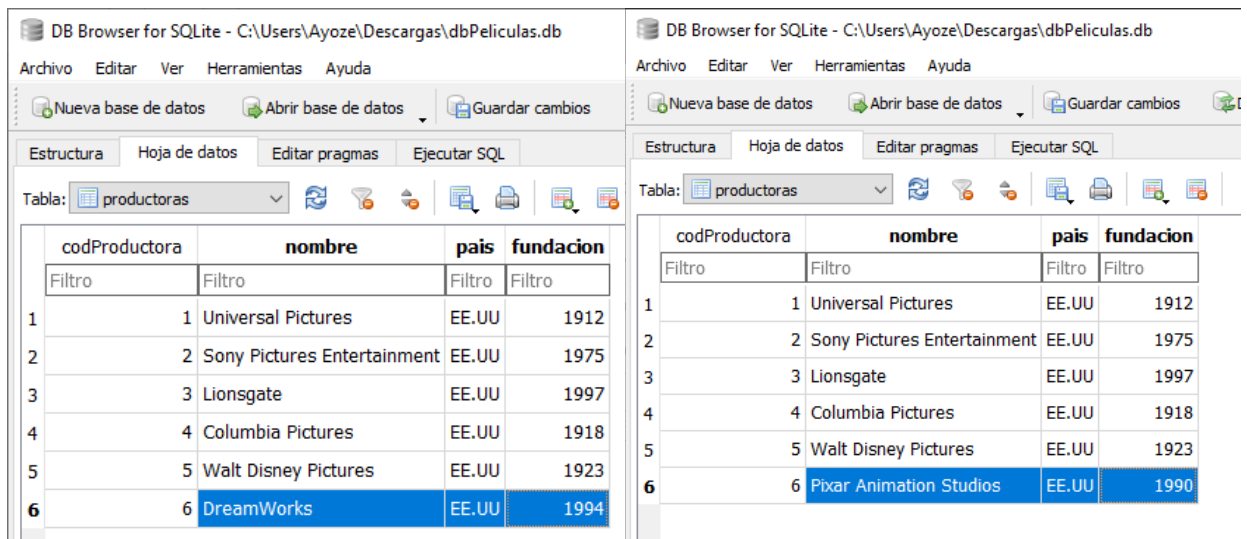


A continuación, vemos como tras añadir un nuevo resgitro el ID se incrementa solo automáticamente, el nombre y el país lo hemos agregado nosotros escribiendo “**Brad Pitt**” y “**EE.UU**” respectivamente.

6	6	Brad Pitt	EE.UU
---	---	-----------	-------

Para la modificación de los registros solo deberemos clicar dos veces sobre el campo del registro y tendremos la oportunidad de escribir un nuevo nombre/país siendo la tabla de actores la elegida, ya que en otras tablas los campos son diferentes.

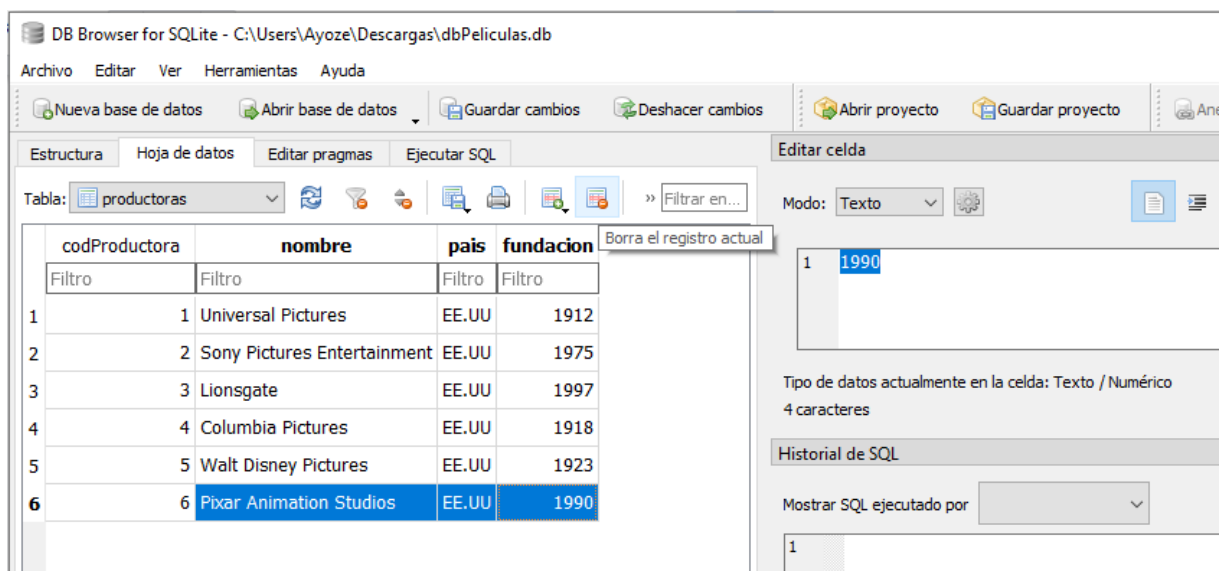
En este caso, el ejemplo lo haremos con el último registro de la tabla “**productoras**” El cual editaremos y borraremos como última opción que nos permite hacer el **DB Browser de SQLite**.



	codProductora	nombre	pais	fundacion
	Filtro	Filtro	Filtro	Filtro
1	1	Universal Pictures	EE.UU	1912
2	2	Sony Pictures Entertainment	EE.UU	1975
3	3	Lionsgate	EE.UU	1997
4	4	Columbia Pictures	EE.UU	1918
5	5	Walt Disney Pictures	EE.UU	1923
6	6	DreamWorks	EE.UU	1994

	codProductora	nombre	pais	fundacion
	Filtro	Filtro	Filtro	Filtro
1	1	Universal Pictures	EE.UU	1912
2	2	Sony Pictures Entertainment	EE.UU	1975
3	3	Lionsgate	EE.UU	1997
4	4	Columbia Pictures	EE.UU	1918
5	5	Walt Disney Pictures	EE.UU	1923
6	6	Pixar Animation Studios	EE.UU	1990

La eliminación de un registro se hará seleccionando la línea del registro y pulsando el botón que nos ofrece el **DB Browser** para ello.



	codProductora	nombre	pais	fundacion
	Filtro	Filtro	Filtro	Filtro
1	1	Universal Pictures	EE.UU	1912
2	2	Sony Pictures Entertainment	EE.UU	1975
3	3	Lionsgate	EE.UU	1997
4	4	Columbia Pictures	EE.UU	1918
5	5	Walt Disney Pictures	EE.UU	1923
6	6	Pixar Animation Studios	EE.UU	1990

Editar celda

1

1990

Tipo de datos actualmente en la celda: Texto / Numérico

4 caracteres

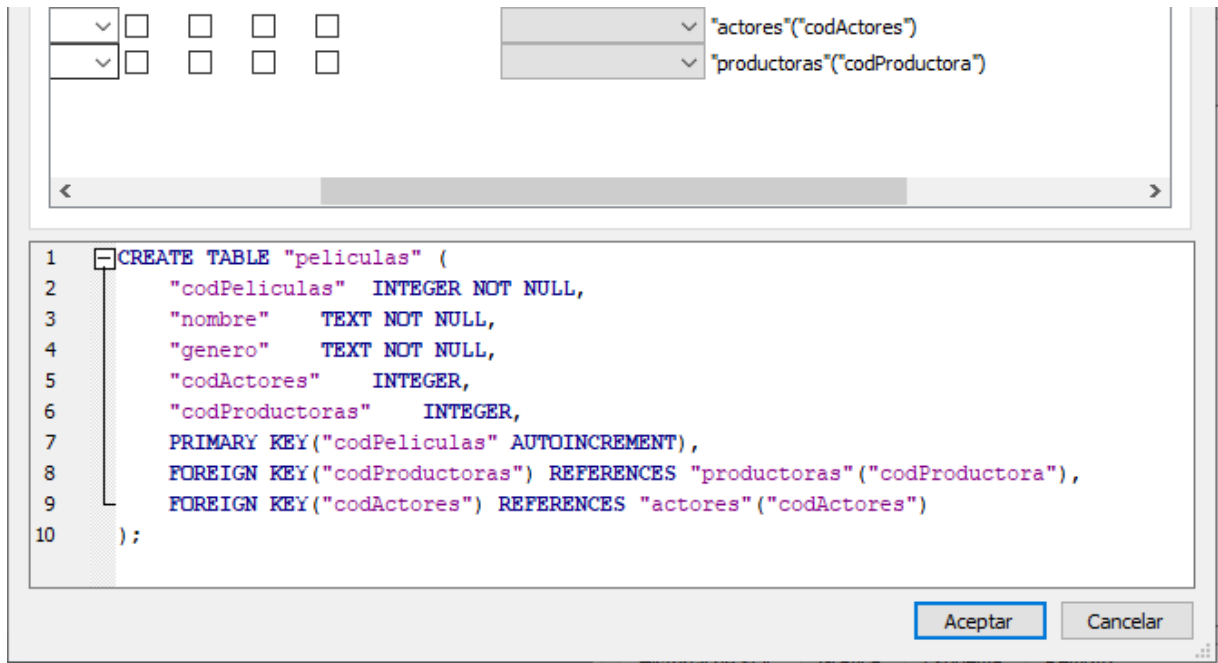
Historial de SQL

Mostrar SQL ejecutado por

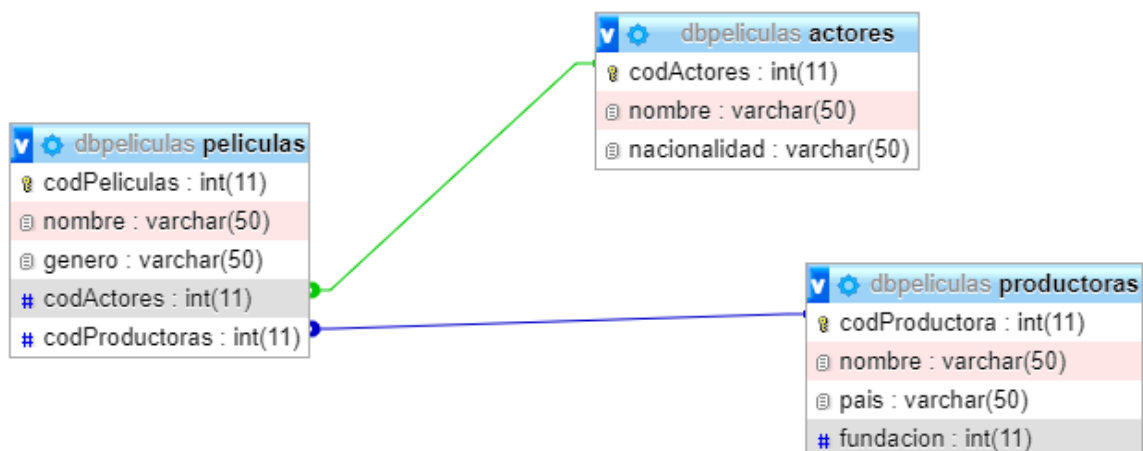
1

3.2 AGREGAR CLAVES FORÁNEAS

A la hora de insertar los campos en una tabla, podemos conectar ambas tablas mediante una clave foránea. En este caso, hemos utilizado el ejemplo de la tabla **“películas”** donde dispone de 2 claves foráneas para la conexión de las tablas.



Como vemos podemos **agregar claves foráneas** de una manera muy sencilla e intuitiva para el usuario que se adentre en este nuevo framework. La base de datos finalmente acaba con esta distribución de tablas.



4 LIBRERÍAS O DEPENDENCIAS PARA EL IDE

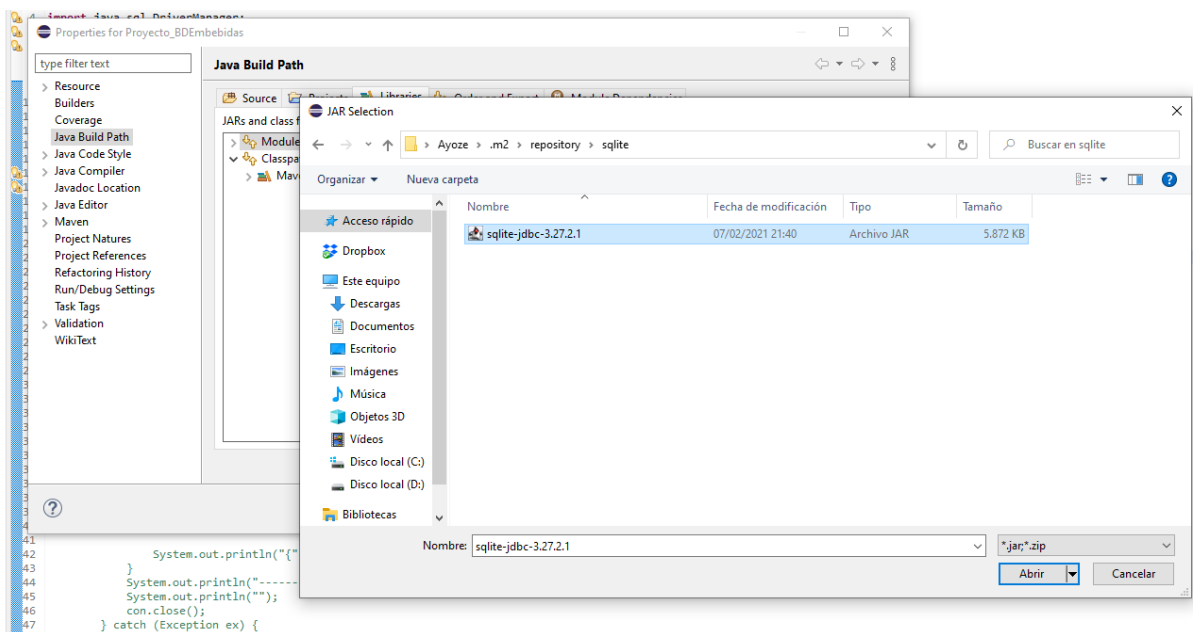
Una de las cosas fundamentales para el funcionamiento de nuestro proyecto Java es añadir la librería o dependencia correcta en nuestro proyecto para así poder ejecutar una correcta conexión con la base de datos. Para ello, se plantean dos opciones:

1. Incluimos la dependencia en nuestro pom.xml

```
<!-- SQLITE -->
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.27.2.1</version>
</dependency>
```

Esta dependencia podemos obtenerla en la página [Maven Repository](#), la cual almacena cientos de dependencias libres de uso para nuestros proyectos. La versión escogida ha sido una de las últimas en salir, la **3.27.2.1**

2. Añadimos la librería .JAR del SQLite



También está disponible la opción de descargar la librería y añadiéndola de manera manual a nuestro proyecto lo cual será igual de válido que la dependencia del **pom.xml**, existente en los proyectos Maven.

5 REALIZAR LA CONEXIÓN CON LA BD

Tras agregar la dependencia o la librería a nuestro proyecto tendremos que comprobar si todo ha salido bien, para ello, colocaremos el archivo de nuestra base de datos **“dbPelículas.db”** dentro de los recursos de nuestro proyecto en una carpeta que hemos llamado **“report”**.

Aquí se encontrará nuestra base de datos gestionada desde nuestro proyecto para así no sufrir ningún problema de localización de la misma. Ya para la comprobación de la conexión a la base de datos será necesario lo siguiente:

- La clase con la dependencia.
 - `Class.forName("org.sqlite.JDBC");`
- La ruta de localización de la BD
 - `C:\\Users\\Ayoze\\eclipse-workspace\\Proyecto_BDEmbebidas\\src\\main\\resources\\report\\dbPelículas.db`
- El conector JDBC
 - `jdbc:sqlite:`

Teniendo estas tres líneas de código podemos comprobar la conexión de nuestra base de datos de la siguiente manera:

Función que mostrará un mensaje por consola si la conexión se realiza satisfactoriamente

```
public static void main(String[] args) {  
    try {  
        Class.forName("org.sqlite.JDBC");  
        Connection con = DriverManager.getConnection(url + nom);  
        System.out.println("¡CONEXIÓN CORRECTA!");  
    } catch (Exception ex) {  
        System.out.println("ERROR: " + ex.getMessage());  
    }  
}
```

6 FUNCIONAMIENTO DEL CRUD

A continuación, pasaremos a explicar el funcionamiento concreto de todo nuestro código que dividiremos en varios fragmentos para así no dejarnos nada atrás y detallar cada parte de relevancia para el funcionamiento de nuestro CRUD gestionando una base de datos SQLite.

Comenzaremos con el “main” desde aquí ejecutaremos una serie de menús donde mostraremos por consola al usuario las opciones disponibles dentro de nuestra aplicación. Entre ellas destacamos las siguientes:

```
System.out.println("1. Listar datos");
System.out.println("2. Insertar datos");
System.out.println("3. Eliminar datos");
System.out.println("4. Modificar datos");
System.out.println("5. Salir");
```

Estas opciones estarán disponibles para las 3 tablas que componen nuestra base de datos. Sin más dilación empecemos con la opción de listar datos, en la cual hemos creado 3 funciones en Java donde podamos recabar los datos de las tablas para así poder imprimirlos facilmente.

Listar los registros

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM actores ORDER BY codActores ASC";
PreparedStatement consult = con.prepareStatement(sql);
ResultSet result = consult.executeQuery();

while (result.next()) {
    String codActores = result.getString("codActores");
    String name = result.getString("nombreActor");
    String nationality = result.getString("nacionalidad");

    System.out.println("{ " + codActores + " }:" + " " + name + " - " + nationality);
}
con.close();
```

Como vemos en la anterior imagen con esa función es posible listar todos los registros de la tabla actores. En ella, hemos establecido una conexión a la base de datos, ejecutado la consulta necesaria y mostrado los datos por consola que hemos recibido de la consulta SQL.

Al igual que en esta imagen, haremos lo mismo pero con la tabla de películas, donde entran en juego más campos debido a las claves foráneas que tiene esta tabla. En este caso, hemos optado por recabar la información de las otras dos tablas para así cambiar el ID de actores por el nombre del registro y el ID de la productora por el nombre del registro.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM peliculas INNER JOIN actores ON actores.codActores = peliculas.codActores "
            + "INNER JOIN productoras ON productoras.codProductora = peliculas.codProductoras "
            + "ORDER BY codPeliculas ASC";
PreparedStatement consult = con.prepareStatement(sql);
ResultSet result = consult.executeQuery();

while (result.next()) {
    String codPeliculas = result.getString("codPeliculas");
    String name = result.getString("nombrePelicula");
    String genre = result.getString("genero");
    String actor = result.getString("nombreActor");
    String producer = result.getString("nombreProductora");

    System.out.println("{ " + codProducer + " }:" + " " + name + " - " + genre + " - " + actor + " - " + producer);
}
con.close();
```

Por último, realizaremos el mismo proceso con la tabla de productoras, la cuál dispone de menos campos y es igual de sencillo de mostrarlos al igual que hicimos anteriormente en la tabla actores.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM productoras ORDER BY codProductora ASC";
PreparedStatement consult = con.prepareStatement(sql);
ResultSet result = consult.executeQuery();

while (result.next()) {
    String codProducer = result.getString("codProductora");
    String name = result.getString("nombreProductora");
    String country = result.getString("pais");
    String foundation = result.getString("fundacion");

    System.out.println("{ " + codProducer + " }:" + " " + name + " - " + country + " - " + foundation);
}
con.close();
```

Insercción de los registros

Tras listar los datos, pasaremos a crear los registros, lo cual se hace sumamente sencillo ya que solo deberemos pedir los datos por consola e insertarlos en nuestra base de datos.

En la imagen podemos ver los 3 insert que hemos creado para cada tabla. En este caso, es necesario **“try-catch”** para la detección de errores, aunque en este informe no profundizaremos tanto y dejaremos el código funcional en nuestro CRUD.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "INSERT INTO actores (nombreActor, nacionalidad) VALUES (?,?)";
PreparedStatement consult = con.prepareStatement(sql);

System.out.print("Nombre del actor: ");
String name = sc.nextLine();
System.out.print("Nacionalidad: ");
String nationality = sc.nextLine();

consult.setString(1, name);
consult.setString(2, nationality);
consult.executeUpdate();
con.close();

Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "INSERT INTO peliculas (nombrePelicula, genero, codActores, codProductoras) VALUES (?, ?, ?, ?)";
PreparedStatement consult = con.prepareStatement(sql);

System.out.print("Nombre de la película: ");
String name = sc.nextLine();
System.out.print("Género: ");
String genre = sc.nextLine();
System.out.print("Código del actor: ");
String codActor = sc.nextLine();
System.out.print("Código de la productora: ");
String codProducer = sc.nextLine();

consult.setString(1, name);
consult.setString(2, genre);
consult.setString(3, codActor);
consult.setString(4, codProducer);
consult.executeUpdate();
con.close();

Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "INSERT INTO productoras (nombreProductora, pais, fundacion) VALUES (?, ?, ?)";
PreparedStatement consult = con.prepareStatement(sql);

System.out.print("Nombre de la productora: ");
String name = sc.nextLine();
System.out.print("País: ");
String country = sc.nextLine();
System.out.print("Fundación: ");
String foundation = sc.nextLine();

consult.setString(1, name);
consult.setString(2, country);
consult.setString(3, foundation);
consult.executeUpdate();
con.close();
```


Modificar los registros

Es el momento de proceder a modificar los datos en las tablas de nuestra base de datos. En este caso, empezaremos con el ejemplo de la tabla actores, la cual pediremos previamente al usuario que indique el registro que desea modificar y tras su elección le permitiremos insertar los campos mostrándole los valores actuales del registro.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql1 = "UPDATE actores SET nombreActor=?, nacionalidad=? WHERE codActores = " + codActor;
PreparedStatement consult = con.prepareStatement(sql1);

String sql2 = "SELECT * FROM actores WHERE codActores = " + codActor;
PreparedStatement modify = con.prepareStatement(sql2);
ResultSet result = modify.executeQuery();

if (result.next()) {
    System.out.print("Nombre del actor (Actual: " + result.getString("nombreActor") + "): ");
    String newNomActor = sc.nextLine();
    System.out.print("Nacionalidad (Actual: " + result.getString("nacionalidad") + "): ");
    String newNationality = sc.nextLine();

    consult.setString(1, newNomActor);
    consult.setString(2, newNationality);
}
consult.executeUpdate();
con.close();
```

Esta imagen podemos ver como se haría la modificación en la tabla de películas.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql1 = "UPDATE peliculas SET nombrePelicula=?, genero=?, codActores=?, codProductoras=? WHERE codPeliculas = " + codFilm;
PreparedStatement consult = con.prepareStatement(sql1);

String sql2 = "SELECT * FROM peliculas INNER JOIN actores ON actores.codActores = peliculas.codActores "
    + "INNER JOIN productoras ON productoras.codProductora = peliculas.codProductoras WHERE codPeliculas = " + codFilm;
PreparedStatement modify = con.prepareStatement(sql2);
ResultSet result = modify.executeQuery();

if (result.next()) {
    System.out.print("Nombre del actor (Actual: " + result.getString("nombrePelicula") + "): ");
    String newNomFilm = sc.nextLine();
    System.out.print("Género (Actual: " + result.getString("genero") + "): ");
    String newGenre = sc.nextLine();
    System.out.print("Código de Actor (Actual: " + result.getString("nombreActor") + "): ");
    String newCodActor = sc.nextLine();
    System.out.print("Código de Productora (Actual: " + result.getString("nombreProductora") + "): ");
    String newCodProducer = sc.nextLine();

    consult.setString(1, newNomFilm);
    consult.setString(2, newGenre);
    consult.setString(3, Integer.parseInt(newCodActor));
    consult.setString(4, Integer.parseInt(newCodProducer));
}
consult.executeUpdate();
con.close();
```

Por último veremos como sería la modificación en la tabla de productoras. No obstante, es de igual manera que la tabla de películas ya que a pesar de que los campos son diferentes, su estructura es la misma.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql1 = "UPDATE productoras SET nombreProductora=?, pais=?, fundacion=? WHERE codProductora = " + codProducer;
PreparedStatement consult = con.prepareStatement(sql1);

String sql2 = "SELECT * FROM productoras WHERE codProductora = " + codProducer;
PreparedStatement modify = con.prepareStatement(sql2);
ResultSet result = modify.executeQuery();

if (result.next()) {
    System.out.print("Nombre de la productora (Actual: " + result.getString("nombreProductora") + "): ");
    String newNomProducer = sc.nextLine();
    System.out.print("País (Actual: " + result.getString("pais") + "): ");
    String newCountry = sc.nextLine();
    System.out.print("Fundación (Actual: " + result.getString("fundacion") + "): ");
    String newFoundation = sc.nextLine();

    consult.setString(1, newNomProducer);
    consult.setString(2, newCountry);
    consult.setString(3, Integer.parseInt(newFoundation));
}
consult.executeUpdate();
con.close();
```

Eliminar los registros

Como último punto de nuestro CRUD queda explicar el código de la eliminación de los registros. En este caso, retomaremos la tabla actores para realizar una consulta que llame a todos los actores y que el usuario tenga elección sobre uno de ellos. Una vez elegido el actor procederemos a eliminarlo tras la confirmación del usuario.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM actores WHERE codActores = " + codActor;
PreparedStatement consult = con.prepareStatement(sql);
ResultSet resultActor = consult.executeQuery();

if (resultActor.next()) {
    System.out.println("{ " + codActores + " } : " + name + " - " + nationality);
    System.out.print("¿Seguro que quiere borrar este actor? (SI = true o NO = false): ");
    String select = sc.nextLine();

    if (select.toLowerCase().equals("si")) {
        delete = true;
    }
}

consult.executeUpdate();
con.close();

if (delete == true) {
    String deleteSQL = "DELETE FROM actores WHERE codActores = " + codActor;
    PreparedStatement deleteActor = con.prepareStatement(deleteSQL);
    deleteActor.executeUpdate();
}
```

Además de ello, también realizaremos el delete en las tablas películas y productoras que se harán de una manera muy similar

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM peliculas INNER JOIN actores ON actores.codActores = peliculas.codActores "
    + "INNER JOIN productoras ON productoras.codProductora = peliculas.codProductoras WHERE codPeliculas = " + codFilm;
PreparedStatement consult = con.prepareStatement(sql);
ResultSet resultFilm = consult.executeQuery();

if (resultFilm.next()) {
    System.out.println("(" + codFilm + "):" + " " + name + " - " + country + " - " + actor + " - " + producer);
    System.out.print("¿Seguro que quiere borrar esta película? (SI = true o NO = false): ");
    String select = sc.nextLine();

    if (select.toLowerCase().equals("si")) {
        delete = true;
    }
}

consult.executeUpdate();
con.close();

if (delete == true) {
    String deleteSQL = "DELETE FROM peliculas WHERE codPeliculas = " + codFilm;
    PreparedStatement deleteFilm = con.prepareStatement(deleteSQL);
    deleteFilm.executeUpdate();
}
```

La anterior captura representa el borrado de registros en la tabla de películas y al siguiente muestra como se realizará el borrado de registros en la tabla productora.

```
Class.forName("org.sqlite.JDBC");
Connection con = DriverManager.getConnection(url + nom);
String sql = "SELECT * FROM productoras WHERE codProductora = " + codProducer;
PreparedStatement consult = con.prepareStatement(sql);
ResultSet resultProducer = consult.executeQuery();

if (resultProducer.next()) {
    System.out.println("(" + codProducers + "):" + " " + name + " - " + country + " - " + foundation);
    System.out.print("¿Seguro que quiere borrar esta productora? (SI = true o NO = false): ");
    String select = sc.nextLine();

    if (select.toLowerCase().equals("si")) {
        delete = true;
    }
}

consult.executeUpdate();
con.close();

if (delete == true) {
    String deleteSQL = "DELETE FROM productoras WHERE codProductora = " + codProducer;
    PreparedStatement deleteProducer = con.prepareStatement(deleteSQL);
    deleteProducer.executeUpdate();
}
```

Finalmente, debo agregar que este CRUD se encuentra disponible en mi **GitHub personal**, donde es posible descargarlo y probarlo para poder comprobar su correcto funcionamiento.



GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC.

Github fue desarrollado por Chris Wanstrath, P. J. Hyett, Tom Preston-Werner y Scott Chacon usando Ruby on Rails, y empezó en 2008. Aunque la compañía, Github, Inc, existía desde 2007.

Enlace hacia el proyecto: https://github.com/Ayoamaro/Proyecto_BDEmbebidas

The screenshot shows a GitHub repository page for 'Ayoamaro/Proyecto_BDEmbebidas'. At the top, there are navigation tabs for 'main', '1 branch', and '0 tags', along with buttons for 'Go to file', 'Add file', and 'Code'. Below this is a table of files and folders with their respective commit messages and timestamps. The 'README.md' file is selected, showing its content. The README describes a CRUD project in Java using SQLite, mentioning a database named 'dbPeliculas' with at least 2 tables. The SQLite logo is displayed at the bottom of the README content. On the right side, there are sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'Languages' section shows a bar chart with 'Java' at 100.0%.

File/Folder	Commit Message	Time Ago
.settings	Upload files	3 days ago
docs/images	New image	2 days ago
src/main	DELETE finished	2 days ago
.classpath	Upload files	3 days ago
.gitignore	Update .gitignore	3 days ago
README.md	Update README.md	2 days ago
pom.xml	Update pom.xml	16 hours ago

Proyecto BD Embebidas

Proyecto CRUD en Java que permita trabajar en una BD Embebida de SQLITE. Se realizará un proyecto en Java accediendo a la base de datos dbPeliculas que contendrá al menos 2 tablas con registros.

7 BIBLIOGRAFÍA

- Información sobre SQLite
 - <https://www.sqlite.org/>
 - <https://es.wikipedia.org/>
 - <https://academiaandroid.com/>
- Información sobre DB Browser for SQLite
 - <https://sqlitebrowser.org>
 - <https://www.redeszone.net/>
- Dependencias y librerías
 - <https://mvnrepository.com>
 - <https://jnj.site.com/>
 - <https://github.com/xerial/sqlite-jdbc>