UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Universitaria de Ingeniería Técnica de Telecomunicación

PROYECTO FIN DE CARRERA

Adaptación de una base de datos bibliográfica

Autor: Carlos Díaz Moreno
Tutor: Waldo Pérez Aguiar

Fecha: Julio de 2017

RESUMEN

El presente proyecto final de carrera titulado “Adaptación de una base de datos bibliográfica” se ha diseñado como una aplicación para Windows que gestione una base de datos bibliográfica y permita al usuario controlar la bibliografía utilizada para preparar esa actividad y finalmente poder generar un documento en Microsoft Word que incluya toda la bibliografía consultada ordenada por orden alfabético por autores. La aplicación se divide en dos partes diferenciadas:

Por un lado toda la gestión de la base de datos bibliográfica donde el usuario puede agregar hasta cuatro tipos distintos de documentos a las tablas: libros, capítulos de libros, artículos y actas de congreso. La aplicación cuenta con dos formas de introducir documentos en la base de datos: usando los formularios de mantenimiento o con una funcionalidad que importa documentos desde un fichero en formato RIS o CIW. Esta base de datos también contiene datos de los usuarios y los permisos que tienen para realizar las distintas operaciones que ofrece la aplicación.

Por otro lado la aplicación ofrece otra parte al usuario que crea actividades o “proyectos” nuevos, edita los ya existentes y borra los que no se necesitan. Después de introducir una breve descripción de la actividad, la aplicación ofrece la posibilidad de asociar documentos seleccionándolos de formularios que se rellenan con información de la base de datos. Una vez que se tiene el proyecto con toda su información es posible generar la lista de referencias bibliográficas en un archivo de WORD. Dependiendo del nivel que tenga el usuario la aplicación le permitirá realizar algunas operaciones, todas las operaciones posibles o sólo consultar proyectos.

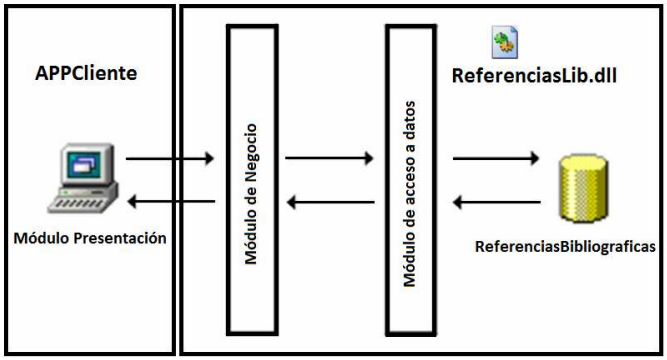
Para diseñar esta aplicación se ha partido del anterior proyecto de fin de carrera titulado: “Modificación de una base de datos Access de referencias bibliográficas”.

Después de estudiarlo se ha rediseñado la base de datos usando el modelo “Entidad – Relación”. La BBDD se mantiene en Access. Sin embargo, para el diseño de la aplicación que gestiona la BBDD y realice todas las funcionalidades se ha elegido desarrollar la aplicación para Windows en el entorno Microsoft .NET 2010 Express usando el lenguaje C# de Microsoft.

Para el diseño de la aplicación se ha aplicado la metodología de diseño orientado a objetos usando la herencia y el polimorfismo para implementar todos los objetos necesarios para el funcionamiento de la aplicación. El programa se divide en 3 capas:

- **Capa de presentación:** Es la que está en contacto con el usuario, donde están incluidos todos los formularios y funcionalidades relacionadas con ellos.
- **Capa de datos:** Es la que se encargará de atacar a la BBDD y realizar operaciones de consulta, inserción, modificación y borrado, así como traducir a los objetos la información obtenida en la base de datos.
- **Capa de negocio:** La capa que pone a disposición de la capa de aplicación las funcionalidades de la capa de datos. Permite la abstracción de la estructura de la BBDD.

Las capas de datos y de negocio están incluidas en una DLL llamada ReferenciasLib.dll.



El PC donde se ejecuta la aplicación deberá tener los siguientes requisitos:

- Sistema Operativo Microsoft Windows XP en adelante.

- Framework .NET 4 de Microsoft
- Office 2003 en adelante, en concreto Access y Word

BIBLIOGRAFÍA

- GIL PERALES, O. *Modificación de una Base de Datos Access de Referencias Bibliográficas.* Proyecto Fin de Carrera. Universidad Politécnica de Madrid. 2005

- KALISKI, B. *PKCS #5: Password-Based Cryptography Specification Version 2.0,* Laboratorios RSA, Septiembre 2000.

- MORIARTI K., KALISKI, B., RUSCH, A. PKCS #5: Password-Based Cryptography Specification Version 2.1, Internet Engineering Task Force (IETF), Enero 2017.

- RIS Format Documentation. Adding a "Direct Export" Button to Your Web Page or Web Applicatio. [documentación]. Thomson Reuters ResearchSoft.

ABSTRACT

This final degree project, titled “Adaptation of a bibliographic database”, has been designed as a Windows application that can manage a bibliographic database. It allows the user to control the bibliography he has used to prepare an activity or course and to generate one Microsoft Word document with all the bibliographic references, alphabetically ordered by authors. The application is divided into two parts:

The first part of the application manages the database where four different types of documents can be added to the database tables: books, book chapters, journals and white paper. There are two methods to introduce documents in database tables supported by the application: using table management forms or importing bibliographic references from files with a RIS or CIW format. This database also includes information about users and their permissions to execute the different operations provided by the application.

The second part of the application allows the user to create and manage different “projects” (activities or courses). After introducing a brief description of the activity, the application will offer to include documents in the database that can be referenced by selecting them from a list form. When the project is ended, with all the info and references, it is possible to generate a MS Word file with all references ordered by author. The application allows the user to do some operations like manage the data base, to create projects or consult projects only, depending on the user permissions.

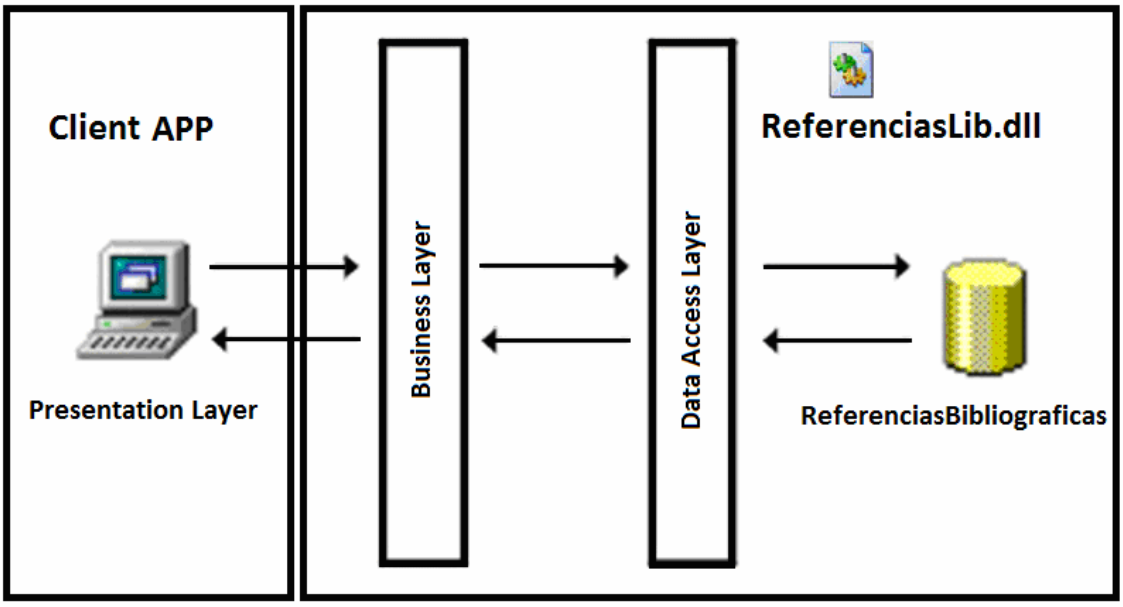
The first step to design the application was to study the previous final degree project titled: “Modification of a MS Access database with bibliographic references”. After that, the database was re-designed using an Entity – Relation

model and keeping it in MS Access. However, in order to develop the application, it has been selected the Microsoft .NET 2010 Express framework and the C# programming language.

Object-oriented programming method was applied for the application design using inheritance and polymorphism features to create all objects. The program has been divided into three layers:

- **Presentation Layer:** this layer is the only one that is accesible by the user. It includes all forms and their funcionalities.
- **Data Access Layer (DAL):** is the layer that knows the database structures and uses them. All the Selects, Inserts, Deletes and Updates are made in this layer. It translates orders into queries and converts the results into objects with the info.
- **Business Layer:** is the layer that is between the Data Access Layer and Presentation Layer. All funcionalities given by DAL are provided by the Business Layer with the format known by the presentation layer. It permits the database structure abstraction.

Data Access Layer and Business layer are included in a DLL called ReferenciasLib.dll.



The following requirements are needed in order to run the application on a PC:

- Microsoft Windows XP Operative System or later
- Microsoft Framework .NET 4
- Microsoft Office 2003 or later

BIBLIOGRAPHY

- GIL PERALES, O. *Modificación de una Base de Datos Access de Referencias Bibliográficas.* Proyecto Fin de Carrera. Universidad Politécnica de Madrid. 2005

- KALISKI, B. *PKCS #5: Password-Based Cryptography Specification Version 2.0,* Laboratorios RSA, Septiembre 2000.

- MORIARTI K., KALISKI, B., RUSCH, A. PKCS #5: Password-Based Cryptography Specification Version 2.1, Internet Engineering Task Force (IETF), Enero 2017.

- RIS Format Documentation. Adding a "Direct Export" Button to Your Web Page or Web Applicatio. [documentación]. Thomson Reuters ResearchSoft.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En esta introducción se van a explicar los objetivos que se han perseguido en este proyecto final de carrera (PFC en adelante), la metodología que se ha utilizado para llevarlos a cabo, las herramientas que se han usado para su desarrollo y, por último, la estructura y los contenidos de esta memoria.

1.1 Objetivos

El PFC *"Adaptación de una base de datos bibliográfica"* tiene como objetivo realizar una mejora respecto de un proyecto anterior que se hizo para este mismo departamento. Para ello se ha desarrollado una aplicación para Windows en el entorno MS .NET 2010 Express y con el lenguaje C# que sea capaz de generar un documento Word con la lista de referencias bibliográficas correspondientes a una asignatura o curso de los que imparte cada profesor. La aplicación contará con una base de datos en MS Access que almacenará todas las referencias según su tipo y el resto de datos que necesite. Además, como mejora propuesta por los mismos profesores este *"software"* está preparado para poder importar registros de documentos en la base de datos a través de la lectura de un archivo generado externamente.

El objetivo principal es permitir a los profesores del departamento generar fácilmente el listado de referencias bibliográficas cuando están preparando libros y apuntes para asignaturas o cursos usando un entorno más potente que el *"Visual Basic for Applications"* que se usa en *"Access"*.

1.2 Metodología

El PFC ha constado de 6 fases:

1ª Estudio de la anterior aplicación de *"Access"* para repararla y hacer un evolutivo.

El PFC en su origen era un estudio, reparación y realización de un evolutivo que permitiera que la aplicación en *"Access"* leyera un *"Excel"* con nuevos documentos y los incluyera en la base de datos.

Se hizo el estudio y se comenzó a reparar la aplicación pero enseguida se vio que sería mejor realizar otro diseño con una tecnología más completa. Sobre todo para la mejora que se pedía, el entorno "*Microsoft .Net 2010"* tiene una "*framework"* que facilita mucho el diseño.

La base de datos, después de estudiarse un cambio, se decidió mantenerla en *"Access"* porque es una aplicación que no va a usar mucha gente a la vez y a la hora de mantener no es difícil de editar.

2ª Toma de requerimientos para el diseño de la nueva aplicación.

Se realizaron varias reuniones con los profesores del departamento para poder canalizar sus necesidades hacia una aplicación. Se decidió que la aplicación contaría con:

- Crear los llamados "proyectos" que serán asignaturas, curso u otras actividades que un profesor debe prepararse. Estos proyectos se asignarán a uno o varios usuarios para que puedan abrirlos. Cada proyecto hará referencia a uno o varios documentos que pueden ser:
    - Artículos
    - Libros
    - Capítulos
    - Actas de congreso. Incluyen también los "*White paper*" que son documentos en forma de manual que explican cómo resolver un determinado problema.

- Una funcionalidad de inicio y fin de sesión con varios tipos de usuarios. La aplicación permitirá a cada usuario realizar unas operaciones u otras en función de su nivel de usuario. Se definieron estos tipos de usuario:
    - Administrador: Tienen acceso a todas las funcionalidades de la aplicación: Administran la base de datos, dan de alta o borran proyectos, documentos, usuarios.
    - Gestor: Tienen acceso a las funcionalidades relacionadas con los proyectos: Dan de alta y borran proyectos y documentos.
    - Usuario: Tiene acceso a los proyectos que un gestor o administrador le haya asignado en modo "sólo lectura".
- Una base de datos en *"Access 2003"* que contenga los proyectos, documentos, usuarios y todo lo que necesite para funcionar. Tiene que ser posible realizar su mantenimiento con la aplicación por parte de los usuarios con perfil "Administrador"
- Una funcionalidad para poder importar más registros para las tablas de documentos desde un fichero y poder hacer más cómoda y rápida la operación. Después de mucho barajar opciones se decidió usar dos formatos posibles de archivos que se basan en archivos de texto plano que cuentan con etiquetas identificativas de lo que es cada campo:
    - El formato RIS que es un estándar bibliográfico. La base de datos de Proquest exporta a este formato.
    - El formato CIW propio de EndNotes, muy parecido al RIS pero con otras etiquetas. La base de datos de Web Of Science exporta a este formato.
- Una funcionalidad para, una vez que esté definido el proyecto con todos sus datos, poder generar un listado de todas las referencias bibliográficas en un archivo de Word.

3ª Diseño e implementación de la base de datos ReferenciasBibliográficas.mdb.

Se diseña usando el modelo "Entidad-Relación". Se decide que cada entidad tenga un identificador auto-numérico por la facilidad con la que se manejan y el hecho de delegar en el sistema la tarea de generar un número único.

4ª Diseño e implementación de la DLL de capa de datos ReferenciasLib.dll.

Una vez que se tiene claro las operaciones que se necesitan hacer con la base de datos, se diseña una "librería de vínculos dinámicos" (DLL) usando la metodología de "programación orientada a objetos" (OOP) aprovechando las ventajas de la herencia el polimorfismo para definir cada uno de los documentos posibles y otros tipos de datos y se crean dos capas:

- Capa de datos: Encargada de la conexión a la base de datos, realizar las operaciones de lectura, escritura y borrado necesarias para el funcionamiento de la aplicación. Esta capa no será visible para la aplicación, se tendrá que hacer uso de los métodos que proporciona la capa de negocio.
- Capa de negocio: Es la encargada de proporcionar servicios a la aplicación para llamar a la capa de datos y que las operaciones en base de datos se sucedan. Contiene las funciones que se exportan.

5ª Diseño de la aplicación AppCliente.exe

Una vez que se cuenta con la base de datos y la DLL que permite realizar todas las operaciones es hora de implementar la aplicación que usará cada profesor del departamento. Tendrá un control de usuario con encriptado de clave usando el "algoritmo PBKDF2", la funcionalidad de importar bibliografía a partir del archivo de formato RIS y se usará la librería "*Microsoft.Office.Interop.Word*" para generar el archivo *"Word"* con la bibliografía. Además se creará un menú con todas las opciones y todos los formularios en forma de ventana necesarios.

6ª Batería de pruebas del conjunto

Se definirán una serie de pruebas a realizar que ayudarán a detectar posibles fallos.

1.3 Medios utilizados

Durante el desarrollo de este PFC se han utilizado distintas herramientas de desarrollo:

- El entorno de desarrollo Visual Studio .NET 2010 Express que contiene la *"Framework"* 4.0 de Microsoft e incluye además un compilador en C#. Esta herramienta se usó para desarrollar la DLL de capa de datos y la aplicación. Era la versión de Visual Studio más moderna cuando se empezó el PFC.
- El software Microsoft Access 2003 por petición expresa del departamento ya que era la versión con la que ellos contaban. Con este software se implementó la base de datos.

1.4 Estructura de la memoria

Para concluir con la introducción se describen los capítulos que incluye esta memoria:

- En el capítulo 1 se ha realizado una introducción al PFC en el que se incluyen los objetivos, la metodología y las herramientas que se han utilizado en su desarrollo.
- En el capítulo 2 se habla de los conceptos teóricos utilizados en la realización de este PFC:
    - Modelo *Entidad-Relación* para el diseño de bases de datos.
    - *Programación Orientada a Objetos* y el concepto de Librería de Enlace Dinámico DLL.
    - el *Algoritmo PBKDF2* para encriptación de contraseñas.
    - Formatos RIS y CIW para guardar bibliografías.
- En el capítulo 3 se explica cómo se implementó el PFC separando las distintas partes que lo conforman.
- En el capítulo 4 se detalla incluye el manual de usuario de la aplicación.
- Finalmente en el capítulo 5 se expone mejoras y ampliaciones que pueden hacerse sobre el proyecto.

2. CONCEPTOS TEÓRICOS

En este capítulo se van a enunciar los conceptos teóricos ya sean aprendidos durante la carrera o investigados para el diseño de la aplicación que conforma este PFC.

2.1 Modelo Entidad – Relación para BBDD

2.1.1 Breve introducción histórica sobre las bases de datos

El término “bases de datos” (desde ahora BBDD) fue escuchado por primera vez en un simposio celebrado en California en 1963. Se puede decir que una base de datos es un conjunto de información relacionada que se encuentra estructurada. Desde el punto de vista informático una base de datos lo forma un conjunto de datos almacenados en disco que permiten un acceso directo a ellos. El software que da de alta, consulta y modifica esa información es conocido como “gestor de BBDD”.

Los orígenes de las BBDD se remontan a las bibliotecas y demás archivos de las bibliotecas de la antigüedad. La idea va evolucionando hasta que en 1884 Herman Hollerith crea una máquina automática de tarjetas perforadas. Más tarde, en la década de los 50 del siglo XX se empiezan a usar cintas magnéticas para almacenar la información con el inconveniente de que el acceso se hace de forma secuencial.

En la década de los 60 se desarrollan las primeras generaciones de BBDD: BBDD en red, BBDD jerárquicas, donde era posible guardar estructuras de datos en listas y árboles; también esa década surge el IDS (“*Integrated Data Store*”), desarrollado por

Charles Bachman, uno de los primeros gestores de bases de datos que pertenecía al CODASYL (*"Conference on Data Systems Languages"*) un consorcio de industrias informáticas que tenía como objetivo la regularización de un lenguaje de programación estándar que pudiera ser utilizado en multitud de ordenadores. Trabajaron en distintos lenguajes, de ahí surgió el COBOL, pero nunca llegaron a establecer un estándar fijo.

En la década de los 70 Edgar Frank Codd, científico informático inglés define el modelo relacional de datos a la vez que publicó una serie de reglas para los sistemas de datos relacionales a través de su artículo "Un modelo relacional de datos para grandes bancos de datos compartidos". Este hecho dio paso al nacimiento de la segunda generación de gestores de BBDD. Esto produce la creación del software "*Relational Software System"* que desembocó en lo que hoy se conoce como Oracle, uno de los gestores de BBDD más famosos del mundo.

Más adelante, ya en la década de los 80 se desarrolla el lenguaje SQL ("*Structured Query Languaje*") que permite realizar consultas para recuperar la información y hacer cambios sobre los datos y la estructura y permite especificar diversos tipos de operaciones frente a la misma información un avance frente a los anteriores lenguajes que realizaban todo por transacciones.

A partir de estos 3 pilares: el modelo relacional, el gestor Oracle y el lenguaje SQL se sientan las bases para los sistemas de BBDD que usamos en la actualidad.

2.1.2 Explicación del modelo Entidad – Relación

El modelo Entidad – Relación es un método para diseñar estructuras de BBDD que más tarde implementaremos con el gestor. Se representa a través de diagramas y está formado por varios elementos:

- Entidad: Representan cosas u objetos que se diferencian entre sí. Cada entidad corresponderá a una tabla. En el modelo se representa como un rectángulo con su nombre dentro.

- Ejemplares: Son cada uno de los registros que contiene una tabla o entidad.
- Atributo: Es cada una de las características que la entidad que necesita conocer y almacenar. Cada entidad puede tener distintos tipos de atributos y éstos podrán ser de distintos tipos como numéricos, texto, fecha, binario, etc. Los atributos serán los elementos de las tablas. Se representan como círculos con su nombre fuera y unidos a la entidad por una línea.
- Relación: Vínculo que define una dependencia entre una o más entidades a través de un atributo en concreto que todas ellas deben contener. Uno de los datos que definen una relación es la cardinalidad que nos indica cuántos ejemplares de una entidad pueden tener el mismo valor para el atributo que marca la relación con cuántos registros de la otra. Se distinguen 3 tipos de cardinalidad:
    - 1 - 1: Relación que implica que un ejemplar de la Entidad 1 esté relacionado con sólo un ejemplar de la Entidad 2. Por ejemplo una matrícula sólo puede pertenecer a un coche y un coche sólo puede tener una matrícula.
    - 1 - N: Relación que indica que un ejemplar de la Entidad 1 puede estar relacionado con varios ejemplares de la Entidad 2 pero un ejemplar de la Entidad 2 sólo podrá estar relacionado con uno de la Entidad 1. Por ejemplo un libro estará formado por varios capítulos, pero un capítulo sólo podrá pertenecer a un libro.
    - M - N: Es la relación menos restrictiva, indica que un ejemplar de la Entidad 1 puede estar relacionado con varios ejemplares de la Entidad 2 y viceversa.

  Las relaciones se representan con un rombo unido a las dos (o más) entidades implicadas con líneas y con un nombre o expresión que explique dicha relación dentro del rombo. En el lado de cada entidad se indica el grado de cardinalidad.
- Clave: Es un atributo, o conjunto de atributos, de la entidad al que aplicamos una restricción que lo distingue de los demás registros. Existen claves de distintos tipos:
    - Primaria: Identifica unívocamente un ejemplar de la entidad no permitiendo que esta clave se repita. Se indica en el diagrama subrayando el atributo que es la clave.

- Clave externa o foránea: Se produce cuando una entidad tiene como atributo uno que en otra entidad es clave primaria. Se utiliza para que esa entidad haga alusión a otra entidad sin tener que contener toda la información de esta otra. Se indica poniendo FK ("*Foreign Key*") en el atributo que la forma.
- Superclave: Es la unión de varios atributos de una entidad para aplicarles la misma restricción para asegurarse que no se repiten en su conjunto.

La figura 2.1 es un ejemplo de lo que podría ser un diagrama de Entidad – Relación:

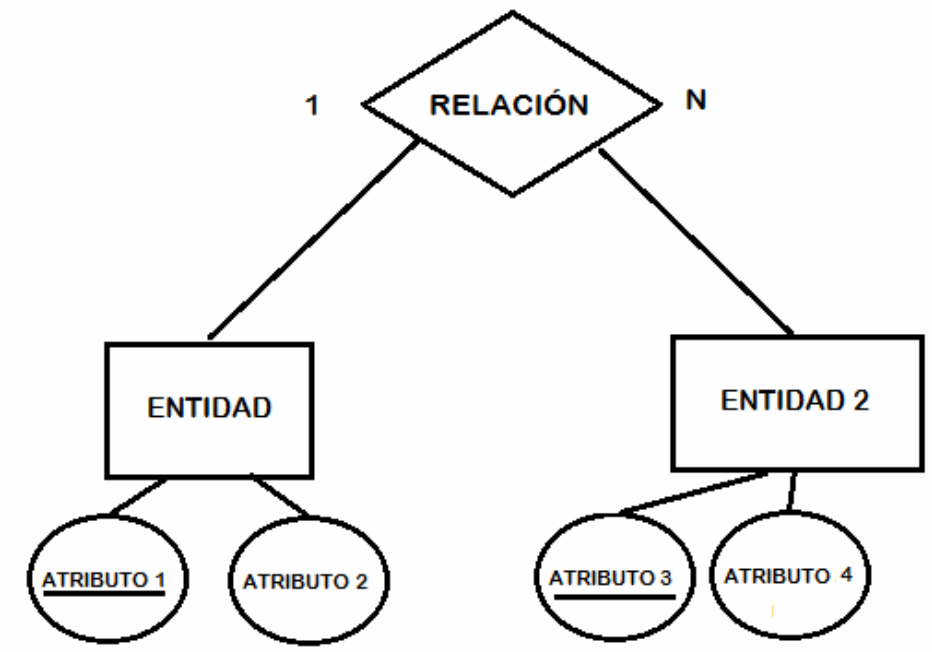


Figura 2.1 Ejemplo diagrama Entidad - Relacion

Se observan las entidades con sus atributos, cada una con su clave primaria. Entidad 2 tiene una relación de 1 – N con Entidad, eso significa que puede haber varios ejemplares de Entidad 2 que tengan relación con un ejemplar de Entidad y además atributo 4, en Entidad 2, es la clave que hace alusión a la clave primaria en Entidad.

Una vez que se tiene el diagrama Entidad – Relación diseñado se traza el diseño de la BBDD a partir de él:

- Las entidades se traducen en tablas
- Los atributos son los campos de la tabla. Los atributos que son claves primarias se declaran como claves primarias de la tabla.
- Las relaciones se traducen a la BBDD de una forma u otra dependiendo del nivel de cardinalidad que tengan.
    - Una relación 1 a 1 entre unas supuestas Entidad 1 y Entidad 2 se traduce en que Entidad 1 tenga su clave primaria y Entidad 2 tenga también su clave primaria y además una clave foránea con Entidad 1 declarada como UNIQUE, así no podrá repetirse en esa tabla.

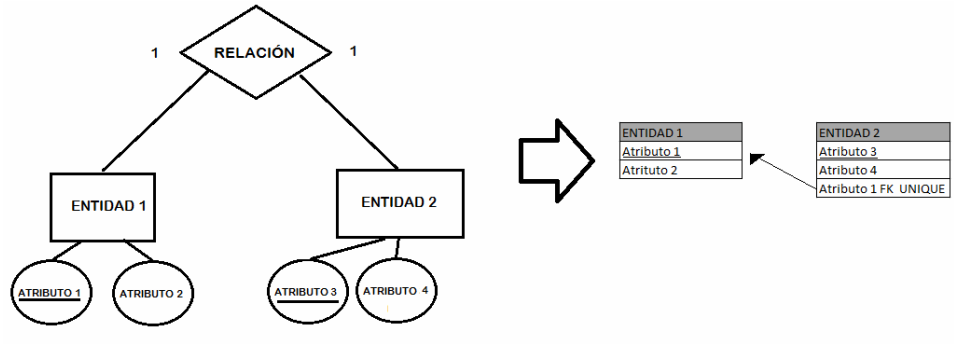


Figura 2.2 Ejemplo de implementación de una relación 1 a 1

- Una relación 1 a N entre unas supuestas Entidad 1 y Entidad 2 se traduce en dos tablas, una para cada entidad donde Entidad 1 guarda la información de algo que se desea que aparezca aludido en Entidad 2, pero no hay ningún problema en que varios registros de Entidad 2 hagan alusión al mismo registro de Entidad 1, con lo cual la clave foránea de Entidad 2 no se declara como UNIQUE.

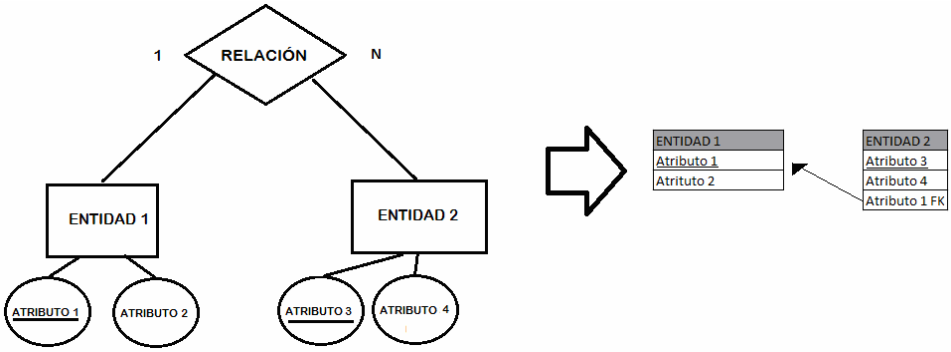


Figura 2.3 Ejemplo de implementación de una relación 1 a N

- Para traducir una relación M a N a tablas y campos hay que tener en cuenta que es una relación en la que varios registros de la Entidad 1 pueden hacer alusión a varios registros de la Entidad 2 y viceversa, con lo cual no se puede implementar añadiendo varias claves foráneas porque no se suele saber cuántos elementos pueden aludirse. La solución es crear otra tabla que relacione la clave primaria de Entidad 1 y la clave primaria de Entidad 2 cuya clave primaria sean sus dos campos y así para cada alusión de una tabla a otra se creará un registro.

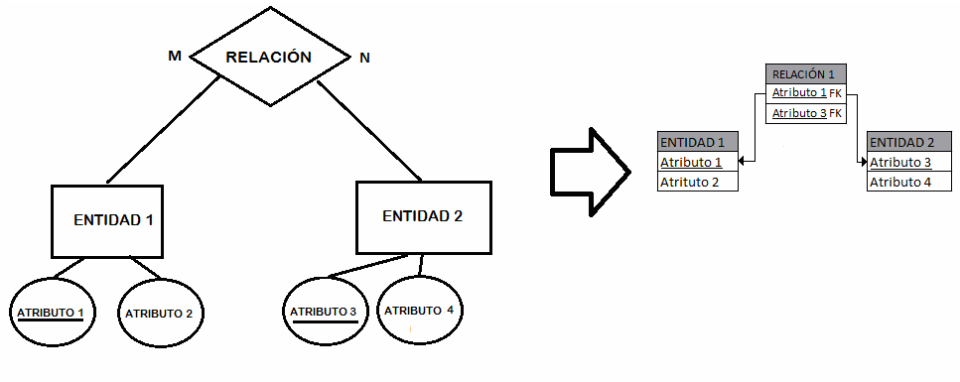


Figura 2.3 Ejemplo de implementación de una relación M a N

Después de estos pasos se tendría completamente definida la base de datos una vez se contara con el diagrama de relaciones entre tablas que puede hacerse de forma completa o separada según convenga para su entendimiento.

2.2 Conceptos de la programación orientada a objetos (POO)

2.2.1 Introducción histórica

La Programación Orientada a Objetos (en adelante POO) surgió en 1967 de la mano de Ole-Johan Dahl y Kristen Nygaard cuando trabajaban en el Centro de Cómputo Noruego en Oslo. Este combo creó el lenguaje Simula 67 para usarlo en la simulación de naves. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos. Esta nueva forma de programar usaba el objeto como elemento fundamental de la construcción de una solución y surgió de forma natural a partir de los principios de *"encapsulamiento"* (el hecho de ocultar la información de

cada objeto) y *"reutilización de código"* (Poder usar distintas funciones pertenecientes a otro objeto a partir de otro objeto que va un paso más del primero incluyendo todas las características de éste.). Debido al hecho de que la POO daba solución a necesidades surgida en la programación estructurada tradicional fue aceptada y en los años 80 surgieron muchos lenguajes que usaban objetos como el caso de C++, del que surgió el lenguaje que se va a usar en este PFC, el C#.

2.2.2 Conceptos del Diseño Orientado a Objetos

El Diseño Orientado a Objetos (DOO a partir de ahora) se basa en 4 pilares fundamentales:

- Abstracción: Es un proceso mental en que se separan las características importantes de algo de lo superfluo.
- Encapsulación: Proceso por el cual se ocultan los detalles de cómo se manipulan las características de una abstracción. Esto hace que desde fuera no se sepa cómo se hacen las cosas, que se usen las funciones pero no se intervenga en el proceso.
- Modularización: Proceso de diseño de un sistema en un conjunto de módulos con poco acoplamiento entre ellos, reduciendo al máximo las dependencias entre ellos; y siento cohesivos, que se expliquen por sí mismos sin depender de otros.
- Jerarquización: Estructurar algo por niveles con cierto grado de dependencia entre ellos.

El diseño deberá ser modular, cada entidad de la BBDD o agrupando las que sean de estructura similar, representará un módulo. Habrá de agruparse las características de cada módulo y pensar en las operaciones que podrán realizar cada uno. Seguidamente se diseñarán relaciones entre los módulos que guarden una jerarquía. También un módulo podrá incluir cierta parte del código que se encargue de alguna operación concreta y que para ello cuente con otros módulos dentro.

2.2.3 Conceptos de la Programación orientada a objetos

Una vez se tiene el diseño habrá que pasar a los objetos. Se trabaja con la clase como elemento elemental. Una clase es el elemento donde se definen las características que va a tener un objeto. Un objeto será una variable del tipo de esa clase. Las clases cuentan con distintos elementos:

- Atributos: Son variables de distintos tipos que conformarán las características del objeto.
- Métodos: Son funciones que la clase dispone para realizar las distintas operaciones. Cuentan con las mismas características de privacidad que las propiedades.
- Estado: Los atributos de las clases suelen ser privados, no puede accederse desde el exterior a ellos. Si el usuario de una clase necesita conocer el valor que tiene en ese momento un atributo o para definirle un valor usará una operación que informe o modifique ese estado del atributo.
- Mensajes: Como pasa con los atributos, los métodos de una clase suelen ser privados para que no pueda conocerse el proceso desde el exterior. Cuando se necesita que el usuario de una clase le pida a ésta realizar una operación, la clase define mensajes que el usuario podrá usar para tal menester.
- Constructores: Cuando se instancia (se reserva memoria para un objeto) un objeto de una clase, dicha clase pone a disposición del usuario una operación para fijar el estado del objeto. Esta operación se conoce como constructor. La clase al menos tiene que contar con el constructor de "por defecto" y pueden definirse todos los que se necesiten.
- Destructor: Cuando se destruye el objeto (se libera la memoria que el objeto ocupa) puede definirse una función que borre el estado del objeto y lo deje listo para su destrucción llamado destructor.

Una vez definidos los elementos que conforman una clase hay que hablar de los dos conceptos: la herencia y el polimorfismo

- La herencia: es la transmisión de código entre unas clases y otras. Existen dos tipos de clases desde el punto de vista de la herencia: las clases padres y

las clases hijas. La herencia se basa en el hecho de que las clases padres transmiten su “conocimiento” a las clases que se definen como sus herederas o hijas, así, si se tiene una clase que ya realiza determinadas operaciones y se necesita otra clase que realice esas mismas o algunas operaciones y además otras propias suyas se le hace clase hija. Esta es una forma de reutilizar código. Como se dijo antes, los atributos y los métodos pueden ser de distintos tipos dependiendo de su privacidad. Cuando se definen como públicos, puede accederse a ellos desde el exterior, cuando se definen como privados es imposible el acceso a ellos y finalmente si están definidos como protegidos sólo pueden acceder a ellos las clases que hereden de ellos, para el resto serán privados.

- El polimorfismo: es una característica de la POO que sirve para no tener que preocuparse sobre los objetos o tipos con los que se trabaja, y abstraernos para definir un código que sea compatible con objetos de varios tipos. Definida una jerarquía de clases, las clases definidas que heredan de una clase base (ya sean hijas o nietas) y uno o más métodos que todas ellas implementan será posible definir un solo código que sea capaz de saber a cuál de los métodos se hace referencia y ejecutarle. Como se ve en la siguiente figura, se define una clase base y luego tres clases más que heredan de la base, cada una de ellas con los mismos métodos A y B. Después durante la ejecución del programa se definirá una variable del tipo base que más tarde será instanciada como cualquiera de las tres clases y a la hora de invocar a cualquiera de los métodos A o B el programa sabrá cuál de todos ellos tiene que ejecutar.

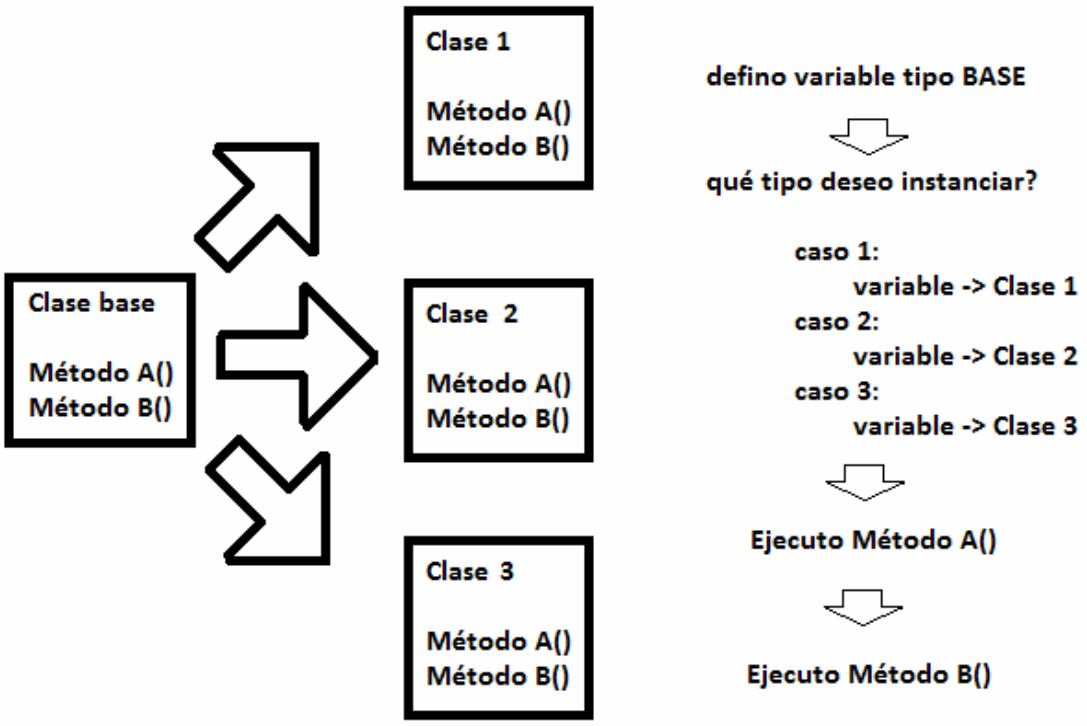


Figura 2.4 Breve esquema explicativo del polimorfismo.

Esto permite abstraerse a la hora de diseñar de con qué tipo se trabaja y concentrarse en el proceso en sí o definir un software capaz de trabajar con distintos equipos cuyo tipo se define al principio.

2.3 DLLs. Librerías de Enlace Dinámico

Una DLL o Librería de Enlace Dinámico es módulo que contiene funciones y datos (clases, tipos) que pueden ser usados por una o varias aplicaciones. En las DLLs pueden definirse dos tipos de funciones: las que se exportan, que pueden usarse también desde dentro de la DLL y las de uso interno que se ejecutan sólo dentro de la DLL. Se genera como un archivo con extensión “.dll”

Las DLLs ayudan a modularizar aplicaciones y ayudan a ahorrar espacio en memoria ya que pueden ser usadas por distintos programas a la vez y aunque cada aplicación reciba su propia copia de la DLL, comparten el código.

Existen dos formas de usar una función de una DLL de forma dinámica:

- Enlazando dinámicamente en tiempo de carga: Esto permite llamar a las funciones de la DLL como si fueran funciones locales. Para ello hay que enlazar la aplicación con la DLL con el archivo de librería .LIB.
- Enlazando dinámicamente en tiempo de ejecución: Durante la ejecución del programa que usa la DLL se llama carga la DLL usando la función *"LoadLibray"* y por cada función de la DLL que se desea usar hay que llamar a la función GetProcAddress que proporciona la dirección donde está alojada la función en la DLL.

2.4 El algoritmo PBKDRF2

El algoritmo PBKDF2 (*"Password-Based Key Derivation Function 2"*) ha sido desarrollado por los Laboratorios RSA y forma parte de sus estándares públicos de encriptación de contraseñas (PKCS), en concreto el PKCS #5 v2.0. publicado además por el Grupo de Trabajo de Ingeniería de Internet (*"IETF"*)[1] como RFC 2898.

El algoritmo usa una Función Pseudo Aleatoria que genera, a partir de la contraseña y una palabra generada aleatoriamente llamada *"salt"*, una palabra que es la encriptación y aplica el proceso a la palabra que va obteniendo tantas veces como se deseé. Por último se realiza la OR – Exclusiva del resultado de cada módulo PRF para producir lo que se conoce como clave derivada. Puede elegirse la longitud de la clave derivada en bits.

La función se define de la siguiente forma:

DK = PBKDF2(PRF, contr, salt, c, dkLen)

Donde:

- DK es la clave derivada.

[1] Internet Engineering Task Force (IETF) (en español, Grupo de Trabajo de Ingeniería de Internet) es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Se creó en los Estados Unidos, en 1986. Es mundialmente conocido porque se trata de la entidad que regula las propuestas y los estándares de Internet, conocidos como RFC.

- PRF es la Función Pseudo - Aleatoria elegida, se define hLen como la longitud de su cadena de salida.
- contr es la contraseña a encriptar, una cadena de bytes.
- salt es la secuencia de bits generados de forma aleatoria. El estándar recomienta una longitud para la salt de 64 bytes
- c es el número de veces que se repite la encriptación
- dkLen es la longitud deseada de la clave derivada en bytes. Como máximo podrá ser de $(2^{32} - 1)\ hLen$

El proceso cuenta con los siguientes pasos:

1. Si dkLen > $(2^{32} - 1)\ hLen$ el proceso para y se devuelve el error "*longitud de clave derivada excesiva*" y se termina el proceso.
2. Se definen:
    a. $m = CEIL(dklen / hLen)$. Nº de módulos que harán falta para conseguir la longitud dkLen de la clave derivada a razón de hLen bytes por módulo. Se coge el valor entero superior inmediato.
    b. $r = dkLen - (m - 1)\ hLen$. Que indica cuántos bits tiene la palabra generada con el último módulo.
3. En cada uno de los m módulos en que se ha dividido la operación se realiza la siguiente operación
    a. se hace una OR de la salt con una codificación en entero de 4 bytes del ordinal de cada módulo (módulo 1 el 1, módulo 2 el 2, ...)
    b. poniendo el valor de contr común a todos los módulos PRF se realiza la función PRF con el valor del OR de la salt y el ordinal del módulo.
    c. El resultado del primer PRF se usa como entrada del siguiente y se hace el PRF de ese resultado y el valor de contr repitiéndose esto hasta completar todos los módulos
    d. Se hace la operación OR – Exclusiva de todos los valores devueltos por las operaciones PRF

La siguiente figura ilustra el proceso de cada módulo:

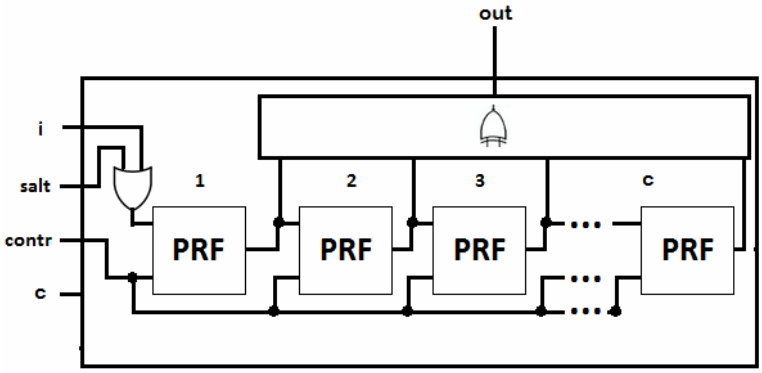


Figura 2.5 Esquema interno del funcionamiento de cada módulo PBKDRF2

4. Se concatenan todos los resultados de cada uno de los m módulos para construir la clave derivada. La siguiente figura ilustra el proceso completo:

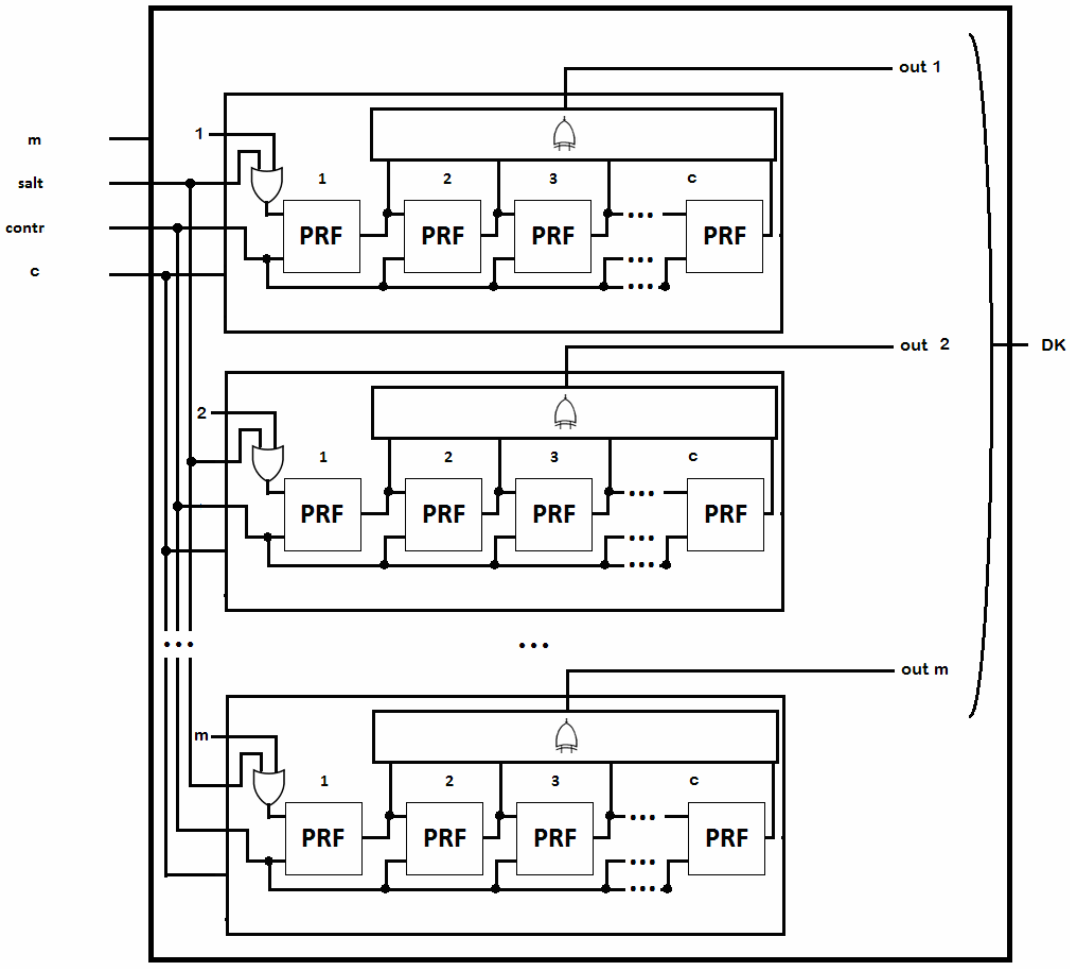


Figura 2.6: Esquema del proceso completo del algoritmo PBKDRF2

Una vez obtenida la DK es imposible recuperar la contraseña a partir de la salt.

2.5 Formato RIS para almacenar bibliografía.

El RIS es un formato para archivos de texto basado en el uso de etiquetas y que se usa para almacenar referencias bibliográficas y así exportarlas o importarlas. Es un formato al que exportan la mayoría de gestores bibliográficos como Proquest o Scopus. El formato surge del software *"Reference Manager"* de "Thomson Reuters" que fue donde apareció por primera vez.

El formato se basa en almacenar todas las características importantes de la referencia identificándolas con un código de dos letras y poniendo una en cada línea.

Las normas del formato son:

- Se pone una etiqueta, se añaden 2 espacios, un guion y otro espacio. Después de esto se pone la información correspondiente a la etiqueta.
- La forma de delimitar cuándo empieza y acaba una referencia es que comience con la etiqueta **TY** (tipo de referencia) y que acabe con la etiqueta **ER** (fin de referencia). El resto de etiquetas pueden ponerse en cualquier posición.
- Cuando se ponen los autores de una referencia hay que usar una línea diferente con la etiqueta **AU** para cada uno de ellos.

Puede verse seguidamente un ejemplo de referencia en formato RIS donde hay dos documentos:

```
TY  - JOUR
AU  - Shannon,Claude E.
PY  - 1948/07//
TI  - A Mathematical Theory of Communication
JO  - Bell System Technical Journal
SP  - 379
EP  - 423
VL  - 27
ER  -
TY  - JOUR
T1  - On computable numbers, with an application to the
Entscheidungsproblem
A1  - Turing, Alan Mathison
JO  - Proc. of London Mathematical Society
VL  - 47
IS  - 1
SP  - 230
```

```
EP  - 265
Y1  - 1937
ER  -
```

La siguiente tabla contiene las distintas etiquetas disponibles que hay en el estándar. Además de las estándar se han encontrado algunas bibliotecas que usan otros códigos distintos que se han marcado con un '*' porque van a tenerse en cuenta para la importación de archivos:

Etiqueta	Significado
TY	Tipo de referencia. Siempre el primero.
A2	Autor secundario. Cada uno de ellos en una línea distinta, precedidos de la misma etiqueta.
A3	Autor terciario. Cada uno de ellos en una línea distinta, precedidos de la misma etiqueta.
A4	Autor Subsidiario. Cada uno de ellos en una línea distinta, precedidos de la misma etiqueta.
AB	Resumen de la referencia.
AD	Dirección del Autor
AN	Nº de acceso
AU	Autor. Cada uno de ellos en una línea distinta, precedidos de la misma etiqueta.
C1	Personalizables: Son etiquetas que contienen información que el gestor de la base de datos piense que debería guardarse.
C2	
C3	
C4	
C5	
C6	
C7	
C8	
CA	Subtítulo
CN	Teléfono
CY	Lugar de publicación
DA	Fecha

DB	Nombre de la Base de datos donde se exportó la referencia.
DO	DOI. Identificador digital del objeto
DP	Motor de BBDD
EP	Página final
VL	Volumen
ET	Edición
IS	Número
JF	Revista, nombre entero
JO	Revista, nombre entero *
JA	Revista, nombre abreviado
J2	Revista en nombre abreviado o libro de capitulo *
KW	Keywords (por cada keyword se añade una línea con su etiqueta)
L1	Archivo adjunto. Es un enlace local al archivo, no una URL
L4	Figura. También es un enlace local, no un archivo.
LA	Idioma
LB	Etiqueta
M1	Número
M3	Tipo de trabajo
N1	Notas
NV	Número de volumen
OP	Publicación original
PB	Editor
PY	Año de publicación
RI	Elemento reseñado
RN	Notas de investigación
RP	Edición de reimpresión
SE	Sección
SN	ISBN/ISSN
SP	Página inicial de un artículo dentro de una revista o un capítulo dentro de un libro.
ST	Título corto
T1	Título *
T2	Título secundario

T3	Título terciario
TA	Traductor
TI	Título
TT	Título de la traducción
UR	URL
Y2	Fecha del acceso a BBDD
ER	Fin de referencia. Siempre al final.

Tabla 2.1 Tipos de etiquetas posibles para el formato RIS

En la siguiente tabla se incluyen los tipos de referencia contemplados por el estándar.

Código	**Significado**
ABST	Sumario
ADVS	Material audiovisual
AGGR	Base de datos agregada
ANCIENT	Texto viejo
ART	Art Work
BILL	Factura
BLOG	Blog
BOOK	Libro
CASE	Caso
CHAP	Capítulo de libro
CHART	Tabla o lista que clasifica publicaciones.
CLSWK	Trabajo sobre cultura clásica.
COMP	Programa informático
CONF	Documento de conferencia.
CPAPER	Documento de conferencia
CTLG	Catálogo
DATA	Archivo informático
DBASE	Base de datos online
DICT	Diccionario
EBOOK	Libro electrónico

ECHAP	Capítulo de libro electrónico
EDBOOK	Libro publicado
EJOUR	Articulo electrónico
ELEC	Página web
ENCYC	Enciclopedia
EQUA	Ecuación
FIGURE	Figura
GEN	Genérico
GOVDOC	Documento público
GRANT	Beca.
HEAR	Audiencia
ICOMM	Documento publicado en cualquiera de las posibilidades que da Internet: Web, chat, correo electrónico, foro, etc.
INPR	Artículo escrito y aprobado pero pendiente de publicar
JFULL	Artículo completo
JOUR	Artículo
LEGAL	Texto legal
MANSCPT	Manuscrito
MAP	Mapa
MGZN	Artículo de revista
MPCT	Imagen en movimiento
MULTI	Archivo multimedia online
MUSIC	Archivo musical
NEWS	Periódico
PAMP	Panfleto
PAT	Patente
PCOMM	Comunicación personal.
RPRT	Informe
SER	Publicación por entregas
SLIDE	Diapositivas, transparencias.
SOUND	Grabación de audio.
STAND	Normativa estándar
STAT	Estatuto

THES	Tesis
UNPB	Trabajo no publicado
VIDEO	Grabación en video.

Tabla 2.2 Tabla con las distintos tipos de documento posibles en el campo TY

2.6 Formato CIW para almacenar bibliografía

El paquete de software EndNote de la empresa *"Clarivate Analytics"* (anteriormente Thomson Reuters) que maneja BBDD bibliográficas tiene una opción de exportar una búsqueda bibliográfica a archivos con extensión CIW.

Durante la fase de investigación de este formato fue imposible encontrar ningún tipo de documentación sobre él. Debido a lo cual la investigación se basó en generar distintos archivos .ciw con diversas búsquedas y proceder a deducir el significado de cada etiqueta.

El formato de estos archivos es semejante al formato RIS pero con algunas cualidades propias.

- Cada campo se identifica con una etiqueta de dos caracteres, se añade un espacio y se escribe la información referente a la etiqueta.
- Es posible poner el texto correspondiente a una etiqueta en varias líneas, para lo cual se pondrá dos espacios en blanco donde iría una etiqueta, se añade el otro espacio en blanco (3 en total) y se continúa con
- La forma de delimitar cuándo empieza y acaba una referencia es que comience con la etiqueta **PT** (tipo de referencia) y que acabe con la etiqueta **ER** (fin de referencia). El resto de etiquetas pueden ponerse en cualquier posición.
- Cuando se ponen los autores de una referencia hay que usar una línea diferente para cada uno de ellos, pero se pone sólo una etiqueta, para los sucesivos autores se ponen dos espacios en blanco donde iría la etiqueta.
- Cuando acaba el archivo se escribe la etiqueta EF

Puede verse seguidamente un ejemplo de referencia en formato RIS donde hay dos documentos:

```
PT B
AN INSPEC:16508921
DT Book Chapter
TI Offline data synchronization with occasionally connected
   databases using smart-IPMS
AU Jagadish, R.M.
   Jyothi, L.S.
   Patil, R.
ED Attele, K.R.
   Kumar, A.
   Sankar, V.
   Rao, N.V.
   Hitendra Sarma, T.
SO Emerging Trends in Electrical, Communications and Information
   Technologies: LNEE 394
PY 2017
PD 2017
BP 53
EP 62
PS 53-62
LA English
IP G06F7/00 Methods or arrangements for processing data by
   operating upon the order or content of the data handled;
   G06F17/30 Information retrieval
C1 Jagadish, R.M.; Dept. of Comput. Sci., BITM, Bellary, India.
   Jyothi, L.S.; RRCE, Bangalore, India.
   Patil, R.; BITM, Bellary, India.
EA Attele, K.R.; Math. & Comput. Sci., Chicago State Univ.,
   Chicago, IL,
   USA.; Kumar, A.; India Council, BioAxis DNA Res. Centre, IEEE,
   Hyderabad, India.; Sankar, V.; Coll. of Eng., Dept. of Electr.
   Eng., JNTU, Ananthapur, India.; Rao, N.V.; Dept. of Comput. Sci.,
   CVR Coll. Of Eng., Hyderabad, India.; Hitendra Sarma, T.; Dept.
   of Comput. Sci. & Eng., Srinivasa Ramanujan Inst. of Technol.,
   Ananthapur, India.
U1 0
U2 0
PU Springer
PV Singapore, Singapore
SC Computer Science (provided by Thomson Reuters)
NR 10
BN 978-981-10-1538-0
DI 10.1007/978-981-10-1540-3_6
UT INSPEC:16508921
ER


PT J
AU Sierra Caballero, Francisco
   Gravante, Tommaso
```

```
TI Ciudadanía digital y acción colectiva en América Latina: Crítica
   de la mediación y apropiación social por los nuevos movimientos
   sociales
SO La trama de la comunicación
VL 20
IS 1
BP 163
EP 175
PD 2016-06
PY 2016
ZB 0
Z8 0
ZR 0
ZS 0
TC 1
Z9 1
SN 1668-5628
UT SCIELO:S1668-56282016000100009
ER

EF
```

La siguiente tabla contiene las distintas etiquetas disponibles que hay en el formato CIW:

Etiqueta	**Significado**
PT	Tipo de referencia. Siempre el primero.
AB	Resumen de la referencia
AN	Nº de acceso
AF	Autor o autores de la publicación en formato completo <Apellido/s, Nombre>. Cada uno de ellos en una línea
AU	Autor o autores de la publicación en formato abreviado <Apellido/s, N.>. Cada uno de ellos en una línea.
BE	Editor o editores de la publicación en formato abreviado <Apellido/s, N.>. Cada uno de ellos en una línea.
BN	ISBN
BP	Página inicial de un artículo o capítulo dentro de una revista o libro.
C1	Customs: Son etiquetas que contienen información que el gestor de la base de datos piense que debería guardarse.
C2	
C3	
C4	

C5	
C6	
C7	
C8	
CL	En caso de conferencia, lugar donde se celebró
CT	En caso de conferencia, su denominación
CY	En caso de conferencia, fecha de celebración
DE	Palabras clave del autor
DI	DOI. Identificador digital del objeto
DT	Descripción del tipo de documento
ED	Editor o editores de la publicación en formato abreviado <Apellido/s, N.>. Cada uno de ellos en una línea.
EI	ISSN electrónico
EM	Dirección de correo electrónico
EP	Página final de un artículo o capítulo dentro de una revista o libro.
ET	Edición
FN	Nombre de la Base de datos donde se exportó la referencia. En cabecera
FU	Entidad que ha financiado el estudio, conferencia, etc.
FX	Texto completo de la financiación
GA	Número IDS de la conferencia
HO	Lugar donde se celebra la conferencia
IP	Keywords, separadas por ";"
IS	Número
LA	Idioma de la publicación
NR	Nº de referencias que se conocen de este documento.
OP	Publicación original
PA	Editorial: Departamento de la entidad que realiza la editorial
PD	Fecha o mes del año en que se ha publicado el documento.
PG	Número de páginas que tiene el documento.
PI	Editorial: Ciudad donde se encuentra el departamento
PS	Rango de páginas donde se encuentra la publicación
PU	Editorial: Universidad o centro que realiza la editorial.
PV	Ciudad de la editorial que realiza la publicación

PY	Año de publicación
RP	Dirección para petición de copias
SC	Áreas de investigación. Separadas por ";"
SE	Revista que publica el artículo
SN	Número Internacional Normalizado de Publicaciones Seriadas ISSN
SO	Nombre del libro donde está publicado el capítulo
SP	Patrocinador /es (separados por ";") de la conferencia
TC	Nº de veces citado en Web of Science
TI	Título
UT	Nº de acceso
VL	Volumen
VR	Versión de la BBDD (suele ir de cabecera del documento)
WC	Categorías dentro de Web Of Science
ER	Fin de referencia. Siempre al final.

Tabla 2.3 Tipos de etiquetas posibles para el formato CIW

En la siguiente tabla se incluyen algunos de los tipos de referencia contemplados por el formato.

Código	**Significado**
B	Libro
J	Artículo o Acta de congreso
S	Capítulo de libro

Tabla 2.4 Tabla con las distintos tipos de documento posibles en el campo PT

3. DISEÑOS E IMPLEMENTACIONES

En este capítulo se procederá a explicar, después de exponer los conceptos teóricos utilizados en la realización del proyecto, cómo se ha realizado el diseño de las distintas partes de la aplicación y su implementación.

3.1 Diseño del modelo de datos.

Para diseñar el modelo de datos primero se obtendrá su modelo Entidad – Relación y a partir de ahí se empezará a diseñar la base de datos siguiendo las pautas del diseño de BBDD relacionales.

3.1.1 Diseño del modelo Entidad - Relación

Después de las reuniones con los profesores para ver qué tipos de documentos necesitaban poder referenciar desde su aplicación y guardar en su base de datos.

Los documentos son los pilares de la base de datos, las tablas más importantes. Los tipos elegidos son cuatro:

- **Libros:** De ellos se necesitará almacenar la siguiente información:
  - Autor/es
  - Título
  - Año de publicación
  - Editorial que lo ha publicado

  - Ciudad de la editorial
  - Edición
  - Keywords: palabras clave que ayudarán a agruparlos y a saber de qué tema son.
- **Capítulos de libros:** Es posible que se haga referencia a un solo capítulo de un libro e incluso que ese capítulo tenga unos autores distintos a los que figuran como autores del libro en sí. Se necesita almacenar la siguiente información:
  - Autor/es
  - Título del capítulo
  - Información del libro al que pertenece
  - Keywords
- **Artículos:** Es la referencia que tendrá más registros, la más usada por los profesores. La información que necesitará almacenarse es:
  - Autor/es
  - Título
  - Revista donde ha sido publicado.
  - Página de la revista donde se inicia el artículo
  - Página de la revista donde acaba el artículo
  - Año de publicación
  - Número de la publicación
  - Volumen dentro del número en caso de que haya varios.
  - Keywords
- **Actas de congresos o *"white paper"*:** La información que se desea almacenar es la siguiente:
  - Autor/es
  - Título
  - Año de publicación
  - Revista donde se publica
  - Código identificativo
  - Keywords

Los distintos documentos no son iguales entre sí, tienen campos comunes, pero no coinciden en todos los campos, con lo cual hay que crear una entidad de cada una de ellas.

Cada documento necesita almacenar información de sus autores y cada uno de los documentos puede tener uno o varios autores y además una misma persona puede ser autor de varios documentos incluso de distinto tipo. Según los fundamentos del modelo Entidad – Relación, para no guardar información redundante sobre una misma persona habría que crear una entidad Autores con el nombre y los apellidos. Cada documento tendría una relación de M a N con la tabla autores.

Los artículos y las actas de congreso hacen alusión a la revista donde son publicados. Por ello se crea una entidad Revista con la siguiente información:

- Nombre corto
- Nombre completo
- Año de primera publicación
- País de publicación
- Web
- Keywords

Una revista puede publicar varios artículos, pero un artículo sólo puede estar publicado por una sola revista, eso significa que la revista tiene relación 1 a N con estos documentos que la aluden.

La revista hace alusión al país de publicación. Así que se debe crear una entidad país. Para crear la entidad se ha consultado el estándar ISO 3166-1 que incluye todos los países del mundo tabulados con los siguientes datos:

- Número identificativo
- Nombre corto
- Nombre largo
- Código de dos letras
- Código de tres letras

La información del estándar es perfecta para crear una entidad y tener todos países identificados, así que se decidió proceder así. Una revista podrá pertenecer sólo a un país y un país podrá albergar muchas revistas: la relación de la entidad país con la entidad revista será de 1 a N.

La entidad capítulo necesita incluir información del libro al que pertenece y siguiendo las normas del Entidad – Relación en lo referente a no generar información redundante se necesita que haya una relación entre la entidad libro y la entidad capítulo. Puede haber varios capítulos que pertenezcan a un solo libro, así que la relación de la entidad libro será de 1 a N con la entidad capítulo.

Los libros y, por extensión, los capítulos de los libros hacen alusión a la editorial que los ha publicado. Debe crearse otra entidad llamada editorial que almacene el nombre de la editorial y la ciudad donde está situada. La ciudad podría ser también convertida en una entidad, pero como no se considera un dato tan importante se ha decidido no hacerla entidad y poner sólo el nombre de la ciudad y la provincia o estado si es necesario. Un libro de una determinada edición podrá pertenecer a una sola editorial mientras que una editorial podrá publicar muchos libros: la relación entre la editorial y el libro es de 1 a N.

Con estos datos ya se puede realizar el “Diagrama Entidad – Relación” que permitirá diseñar después la Base de datos. Para que pueda entenderse mejor y plasmarse mejor en esta memoria se ha decidido realizarlo a trozos. Las relaciones 1 a N aparecerán como 0 a N porque la base de datos permitirá dar de alta registros que no tengan entidades con quien se relacionen para poder darlas de alta más tarde y establecer también más tarde la relación.

Primero se muestran las relaciones que hay entre los libros, los capítulos, las editoriales y los autores:

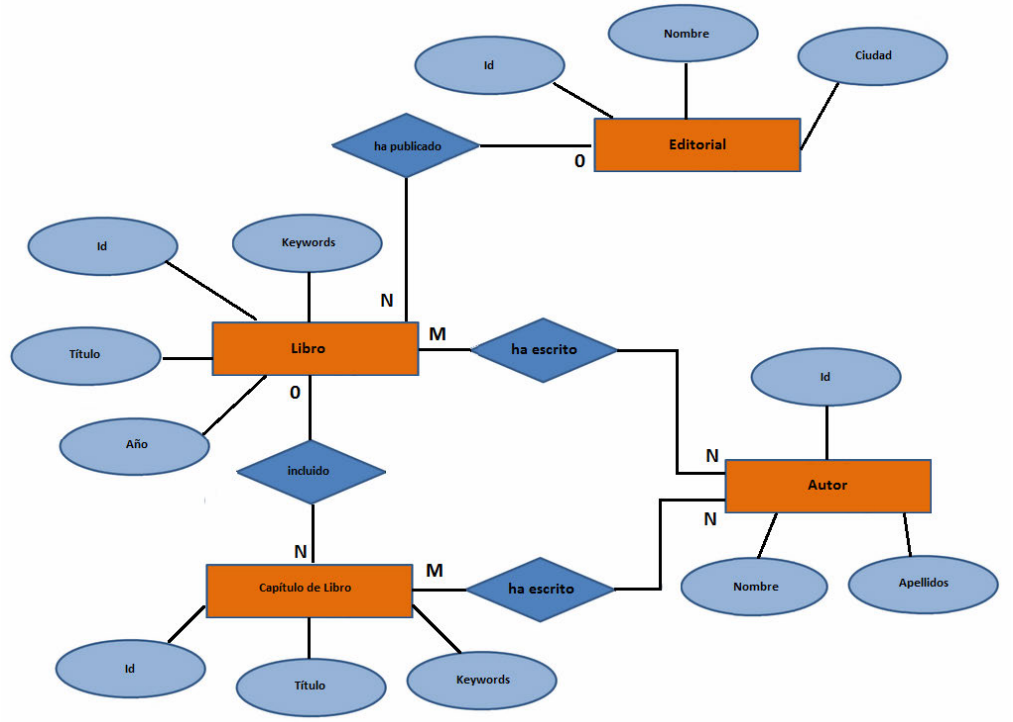


Figura 3.1 Diagrama Entidad – Relación entre los libros, capítulos, editoriales y autores

Seguidamente aparecen las relaciones entre artículos, actas, autores, revistas y países.

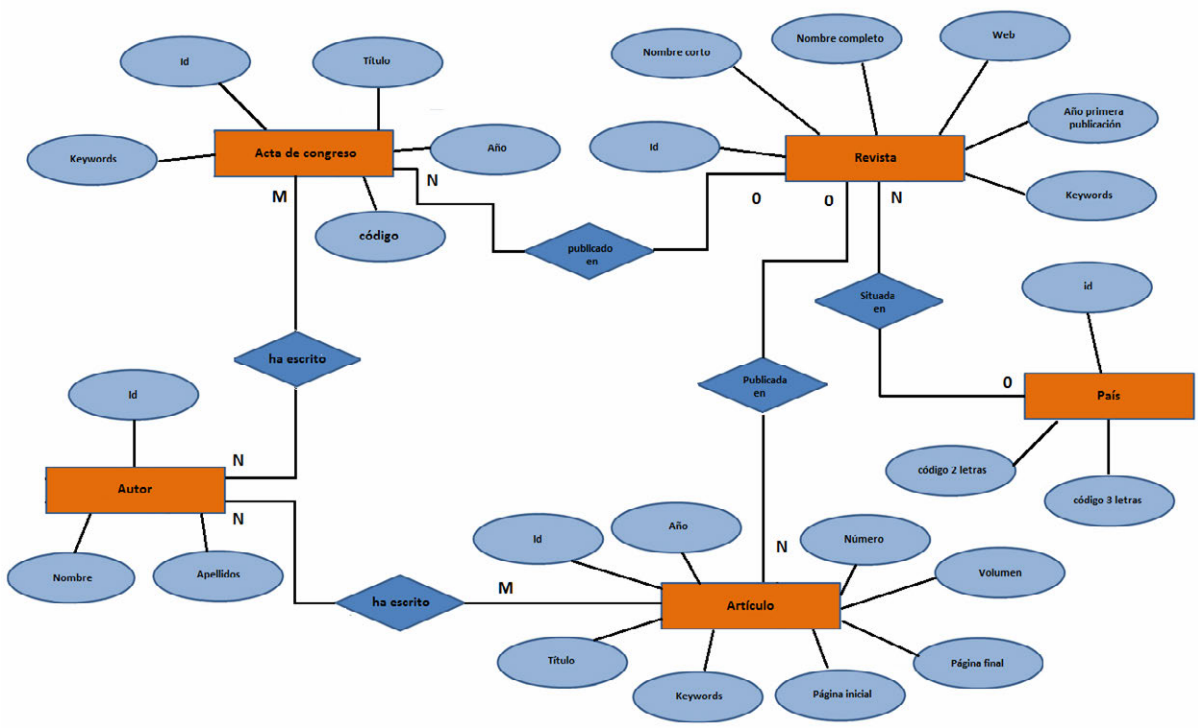


Figura 3.2 Diagrama Entidad – Relación de artículos, actas, revistas, países y autores

Cada usuario de la aplicación puede dar de alta “proyectos” que son asignaturas, cursos o cualquiera de las distintas actividades que realizan en su día a día como profesores y de los que necesitan referenciar su bibliografía. Se necesita almacenar los siguientes campos de los proyectos:

- Descripción
- Año
- Observaciones: Explicar en qué consiste el proyecto.

Cada proyecto puede referenciar a uno o varios de los distintos documentos que pueden guardarse en la BBDD, además un documento puede ser referenciado por varios proyectos, con lo cual la relación entre la entidad proyecto y cada uno de los documentos es de M a N.

La aplicación necesita tener un control de usuario para entrar en ella y además los proyectos necesitan poder asociarse a los usuarios que tienen permitido examinarlos. El usuario tiene que tener, como se explicó anteriormente, un nivel de usuario que le permitirá acceder a unas partes u otras de la aplicación. Esto demuestra que es necesaria otra entidad “Usuario” cuya información a almacenar sería:

- Nombre
- Fecha de alta en el sistema
- Fecha de baja en el sistema
- Un campo que informe si el usuario está bloqueado
- Los campos necesarios para el control de contraseña (que se diseñarán más adelante)

Además como el usuario necesita hacer referencia a un nivel de usuario, es necesaria la entidad “Nivel de Usuario” para que la entidad “Usuario” haga alusión a ella y no almacenar reiteradamente información sobre el nivel en cada usuario. La entidad “Nivel Usuario” necesita esta información

- Identificador numérico que además de identificar unívocamente al nivel, identifique lo alto o bajo que es el nivel.
- Descripción del nivel

Un usuario puede tener cualquiera de los niveles que se definen así que la relación entre la entidad “Usuario” y la entidad “Nivel Usuario” es de 1 a N. Un proyecto puede estar asignado a muchos usuarios y un usuario puede ser asignado a muchos proyectos, con lo cual la relación entre la entidad “Proyecto” y la entidad “Usuario” es de M a N. Con esta información puede diseñarse el resto del Diagrama Entidad – Relación:

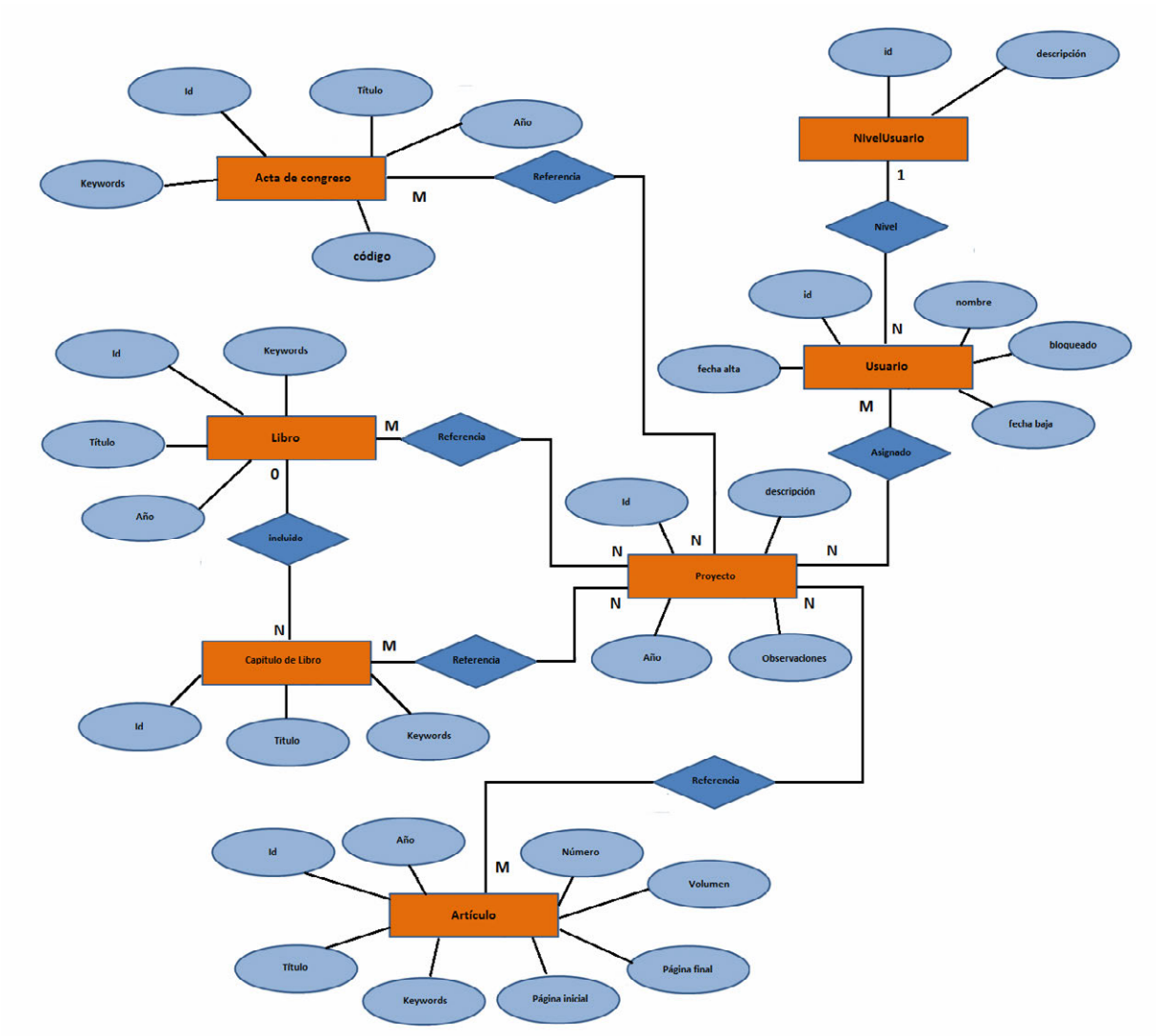


Figura 3.3 Diagrama Entidad – Relación entre las entidades proyecto, libro, capítulo, artículo, acta, usuario y nivel de usuario.

3.1.2 Diseño de la BBDD a partir del diagrama Entidad – Relación

Una vez realizado el diagrama Entidad – Relación se procede a diseñar el modelo de datos con las tablas que será necesario crear en la BBDD de la aplicación con la forma anteriormente comentada:

1º Todas las entidades son tablas y sus características son campos de la tablas.

Según esta norma se crean las primeras tablas:

Revista

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	tituloCorto	Texto
	tituloLargo	Texto
	anyoPrimeraPublicacion	Número
	web	Texto
	keywords	Texto

Pais

	Nombre del campo	Tipo de datos
🔑	id	Número
	nombre	Texto
	nombreLargo	Texto
	codigoISO2char	Texto
	codigoISO3char	Texto

Libro

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	titulo	Texto
	anyo	Número
	edicion	Número
	keywords	Texto

Autor

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	nombre	Texto
	apellidos	Texto

CapituloLibro

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	titulo	Texto
	keywords	Texto

ActaCongreso

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	titulo	Texto
	anyo	Número
	codigo	Texto
	keywords	Texto

Artículo

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	titulo	Texto
	anyo	Número
	numero	Número
	volumen	Número
	paginaInicial	Número
	paginaFinal	Número
	keywords	Texto

Editorial

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	nombre	Texto
	ciudad	Texto

Usuario

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	nombre	Texto
	hashClave	Objeto OLE
	sal	Objeto OLE
	bloqueado	Sí/No
	fechaAlta	Fecha/Hora
	fechaBaja	Fecha/Hora

NivelUsuario

	Nombre del campo	Tipo de datos
🔑	id	Número
	descripcion	Texto

Proyecto

	Nombre del campo	Tipo de datos
🔑	id	Autonumérico
	descripcion	Texto
	anyo	Número
	observaciones	Texto

Figura 3.4 Paso de entidades a tablas

Los campos “hashClave” y “sal” de la tabla Usuario se explicarán más adelante.

2º Las relaciones 1 a N se traducen en añadir un campo en la entidad donde está colocada la N que sea clave foránea (FK) de la entidad que tiene el 1.

Se observan las figuras de los Diagramas Entidad - Relación y se buscan las relaciones 0 a N:

- La entidad Libro tiene una relación 0 a N con la entidad CapituloLibro, con lo cual a la entidad artículo se le añade un campo más llamado IdLibro que ese clave foránea de la entidad Libro.
- La entidad Editorial tiene una relación 0 a N con la entidad Libro. A la tabla Libro se le añade un campo más llamado IdEditorial que es clave foránea de la tabla Editorial.
- La entidad Revista tiene una relación 0 a N con las tablas Articulo y ActaCongreso, lo que se traduce en añadir a sendas últimas tablas un campo IdRevista que hará alusión a la clave Id de dicha tabla Revista.
- La entidad Pais tiene una relación 0 a N con la tabla Revista. Eso se traduce con un campo IdPais a la tabla Revista.
- La entidad NivelUsuario tiene una relación 1 a N con la tabla Usuario, lo que se traduce añadiendo un campo IdNivel a la tabla Usuario.

Después de traducir las relaciones 1 a N las tablas quedan así:

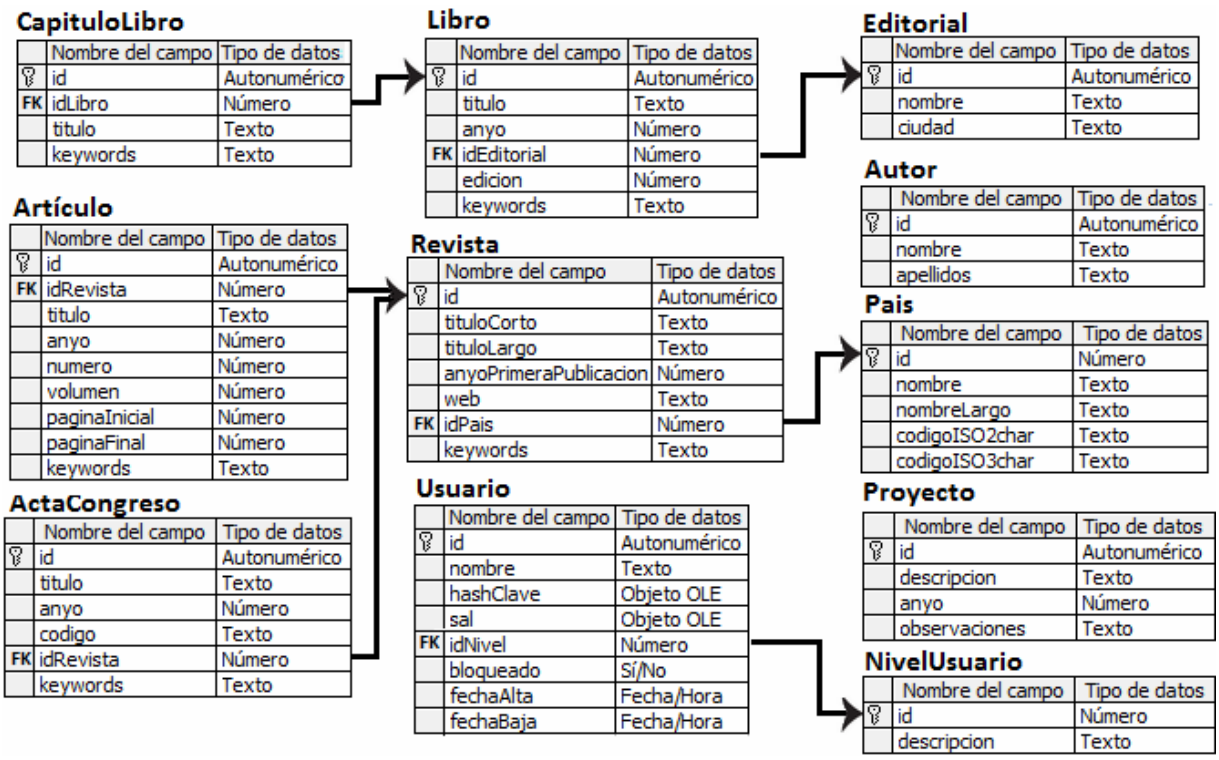


Figura 3.5 Traducción de las relaciones 1 a N a campos añadidos.

3º Las relaciones M a N se traducen en una tabla con dos campos numéricos. Cada uno de ellos hace alusión respectivamente a la clave de una de las tablas relacionadas. Esto implica que:

- Todos las entidades de documentos tienen una relación M a N con la tabla autor, así que se crea una tabla para relacionar cada documento con la tabla autor con dos campos IdAutor y el id del documento que corresponda (idLibro, idCapitulo, idActa, idArticulo) e IdAutor.
- La entidad Proyecto tiene una relación M a N con cada uno de los documentos lo cual se traduce en una tabla por cada documento con dos campos cada una: IdProyecto e Id del documento correspondiente.
- La entidad Proyecto tiene además una relación M a N con la tabla Usuario, relación que se traduce con la creación de la tabla UsuarioProyecto que cuenta con dos campos: IdProyecto e IdUsuario.

Finalmente el diseño del modelo de datos se muestra en la figura 3.6 en la página siguiente.

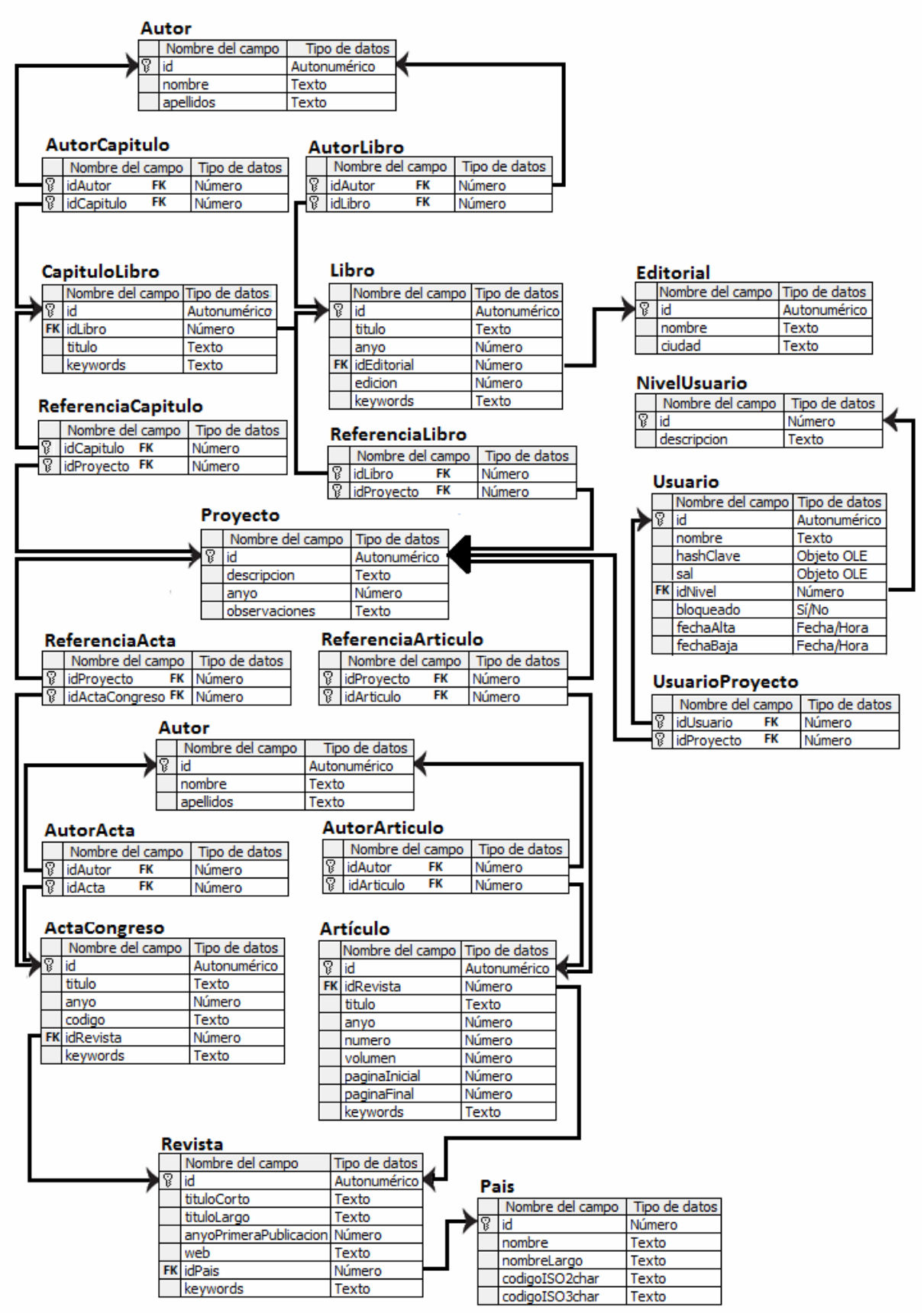


Figura 3.6 Diseño final del modelo de datos de la aplicación.

3.2 Diseño en POO de la aplicación

Una vez implementado el modelo de datos de la aplicación se pasará a diseñar la aplicación en C# con la metodología de la POO. Para poder separar la parte de presentación del resto de funcionalidades se separará la aplicación en 3 capas:

- **Capa de presentación:** Es la parte de la aplicación que está en contacto con el usuario, donde están incluidos todos los formularios, ventanas y funcionalidades relacionadas con ellos. Es la parte del programa que representará al usuario la información que requiera y donde el usuario introducirá la que quiera añadir.
- **Capa de datos:** Parte del código que se encargará de atacar a la BBDD y realizar operaciones de consulta, inserción, modificación y borrado, así como transacciones (serie de operaciones que es necesario realizarlas todas o volver atrás), traducir a los objetos la información obtenida en la base de datos.
- **Capa de negocio:** Parte del código que sirve de enlace entre la capa de datos y la capa de presentación. Se encarga de proporcionar las funciones que necesita la capa de presentación para obtener, guardar o modificar la información y se encarga de llamar a las funciones que oferta la capa de datos.

Las capas de datos, negocio y tipos se incluyen dentro de una DLL llamada “ReferenciasLib.dll”. El propósito de incluir todo esto dentro de una DLL es dejar la aplicación preparada para en caso de querer cambiar la capa de presentación por otra, ésta sea independiente y sea todo fácil.

La siguiente figura muestra la separación de las capas, cómo se relacionan entre ellas y dentro de la parte de la aplicación en la que están.

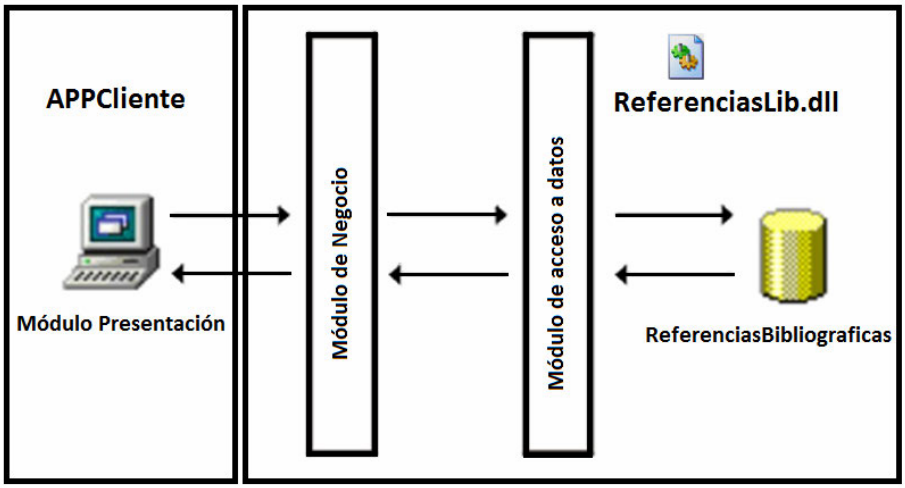


Figura 3.7 Distintas capas de la aplicación dentro de la aplicación

3.2.1 Diseño de las tipos que se usan en toda la aplicación

En los tipos de datos que va a usar la aplicación existirán 3 niveles de herencia:

- **Clases independientes y bases:** Son objetos que no heredan de ninguna clase, ya sea porque son la base de donde heredan el resto de clases o porque son clases independientes que no heredan ni heredan de ellas. Hay tres clases de este tipo:
    - Cada una de las entidades de la BBDD tiene en común que tienen un ID para identificarlos unívocamente y un nombre o descripción así que se creará una clase base de la que heredarán todas las clases que se creen, se llamará CBase que incluirá dos propiedades una de tipo "*int*" llamada id que servirá para guardar el identificador y otra de tipo "*string*" llamada nombre que guardará el nombre o denominación.

CBase
int Id string Nombre
CBase() CBase(int id, string nombre) ~CBase()

- Para contener todos los documentos asociados a un proyecto se crea una clase llamada CDocumentos que contiene 4 listas una para cada tipo de documento que puede contener un proyecto. Las clases a las que pertenecen las propiedades de la clase se definirán más adelante

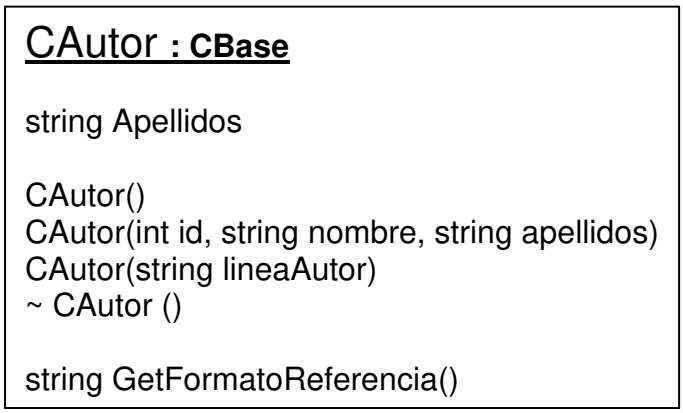
CDocumentos
CActasCongreso Actas CArticulos Articulos CLibros Libros CCapitulosLibro Capitulos CDocumentos() ~CDocumentos()

- Debido a que en algunos casos se necesita hacer una lista de todos los documentos sin separar por tipo y que estén mezclados, sin que haga falta contener todos sus datos, sino los datos comunes de todos los documentos se crea también una clase llamada CDocumentoGenerico que contiene un identificador de tipo de documento y una instancia de la clase CDocumentoBase (que se definirá más adelante).

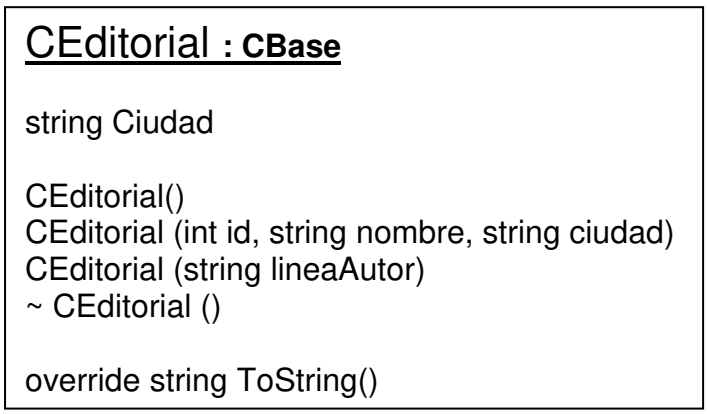
CDocumentoGenerico
CDocumentoBase Documento CTipoDocumento TipoDocumento CDocumentoGenerico(CDocumentoBase documento, CTipoDocumento tipoDocumento) ~CDocumentoGenerico()

- **Clases que heredan de CBase:** Las clases tipo que son equivalentes a las entidades de la BBDD heredan prácticamente todas de la clase CBase. Estas clases son:
  - **CAutor:** Esta clase, además de las propiedades que tiene por herencia, incluye una propiedad tipo "*string"* llamada "apellidos" para guardar los apellidos del autor. Además de los constructores pertinentes incluye un

constructor extra que obtiene la información de un autor a través de una línea con formato “apellidos, nombre”. En los métodos incluye uno de tipo “*string”* llamado “*GetFormatoReferencia()*” que devuelve el nombre y apellidos del autor en formato “apellidos, nombre”, que será útil para rellenar el documento de la bibliografía.



- **CEditorial:** Esta clase, además de las propiedades que tiene por herencia, incluye la propiedad de tipo “*string”* llamada “ciudad” que guardará la ciudad de la editorial. En los métodos, sobrescribe el método “*ToString()*” para devolver la editorial en formato “editorial, ciudad”.



- **CNivelUsuario**: Esta clase, además de las propiedades que tiene por herencia, incluye la propiedad de tipo “*string”* llamada “Descripción” que devuelve la propiedad “nombre” de la clase base.

CNivelUsuario : **CBase**

CNivelUsuario()
CNivelUsuario(long id, string descripcion)
~CNivelUsuario ()

override string ToString()

- **CPais**: Esta clase, además de las propiedades que tiene por herencia, incluye las propiedades de tipo "*string"* llamadas "nombreLargo" para el nombre completo del país, "codigoISO2char" y "codigoISO3char" que almacenan sus códigos de 2 y 3 caracteres respectivamente.

CPais : **CBase**

string NombreLargo
string CodigoISO2
string CodigoISO3

CPais()
CPais(long id, string nombreCorto,
string nombreLargo, string codigoISO2,
string codigoISO3)
~CPais()

- **CRevista**: Esta clase, además de las propiedades que tiene por herencia, incluye las propiedad tipo "*string"* "nombreCorto", "web", "pais" y "keywords" y de tipo "*int*" anyoPrimeraPublicacion" que muestran el campo del mismo nombre de la BBDD revista.

CRevista : **CBase**

string NombreCorto
int AnyoPrimeraPublicacion
string WEB
CPais Pais
string KeyWords

CRevista()
CRevista(long id, string nombreCorto,
string nombre, int anyoPrimeraPublicacion,
string web, CPais pais, string keywords)
~CRevista()

void RellenaAmbosTitulos()

- **CUsuario**: Esta clase, además de las propiedades que tiene por herencia, incluye la propiedad de tipo "*CNivelUsuario*" Nivel, una tipo "*bool*" llamada bloqueado y dos de tipo "*DateTime*" llamadas "*alta*" y "*baja*" y un método de tipo "*bool*" llamado "*Disponible()*" que informa si el usuario está disponible para la aplicación.

CUsuario : **CBase**

CNivelUsuario Nivel
bool Bloqueado
DateTime FechaAlta
DateTime FechaBaja

CUsuario()
CUsuario(long id, string nombre)
CUsuario(long id, string nombre,
CNivelUsuario nivel, bool bloqueado,
DateTime fechaAlta, DateTime fechaBaja)
~ CUsuario()

public bool Disponible()

- **CProyecto**: Esta clase es la que guardará toda la información de un proyecto, además de las propiedades que tiene por herencia, incluye dentro de sus propiedades un conjunto de listas de cada uno de los documentos que más adelante se explicarán. Además de ello incluye la

propiedad de tipo "*int"* llamada "anyo", otra de tipo *"CUsuarios"* que incluye la lista de usuarios que están asignados a esta aplicación y la propiedad de tipo "*string"* "observaciones".

CProyecto : CBase

int Anyo
CUsuarios Usuarios
CDocumentos Documentos
string Observaciones

CProyecto()
CProyecto(long id, string descripcion,
int anyo, string observaciones)
CProyecto(long id, string descripcion,
int anyo, string observaciones,
CUsuarios usuarios)
CProyecto(long id, string descripcion,
int anyo, string observaciones,
CUsuarios usuarios, CDocumentos documentos)
~CProyecto()

- **CDocumentoBase:** Existe además una clase que hereda también de la clase "*CBase"* y que se ha diseñado con todos los elementos que tienen en común los 4 tipos de documentos que soporta la aplicación. Esta clase incluye las siguientes propiedades: una de tipo CAutores (lista de CAutor) llamada autores que incluirá los autores de un documento, otra de tipo "*int*" llamada anyo que almacena el año de publicación del documento, y finalmente una de tipo "*string"* llamada keywords que incluye las claves por las que puede ser definido el documento.

CDocumentoBase : **CBase**
string Titulo CAutores Autores int Anyo string KeyWords CDocumentoBase() CDocumentoBase(long id, string titulo, int anyo, string keywords) CDocumentoBase(long id, string titulo, int anyo, CAutores autores, string keywords) ~CDocumentoBase()

- **Clases que heredan de CDocumentoBase:** Las clases correspondientes a las entidades que guardan los datos de los cuatro documentos soportados por la aplicación son las clases que heredan de esta clase base de segundo nivel. Cada una con sus características propias:
    - **CLibro:** Esta clase cuenta con la propiedad de tipo "*int" "*edicion" que informa del nº de edición a la que pertenece esa copia y otra del tipo CEditorial "editorial" que contiene los datos de la editorial que ha publicado el libro.

CLibro : **CDocumentoBase**
CEditorial Editorial int Edicion CLibro() CLibro(long id, string titulo, int anyo, CEditorial editorial, int edicion, string keywords) CLibro(long id, string titulo, CAutores autores, int anyo, CEditorial editorial, int edicion, string keywords) ~CLibro()

    - **CCapituloLibro:** Esta clase tiene la propiedad de tipo CLibro "libro" que almacena la información del libro al que pertenece, de donde se

sacará cierta información. Tiene varios constructores para adaptarse a las necesidades de la aplicación.

CCapituloLibro : **CDocumentoBase**
CLibro Libro CCapituloLibro() CCapituloLibro(long id, string titulo, int anyo, string keywords) CCapituloLibro(long id, string titulo, int anyo, string keywords, CAutores autores) CCapituloLibro(long id, string titulo, int anyo, string keywords, CLibro libro) CCapituloLibro(long id, string titulo, int anyo, string keywords, CAutores autores, CLibro libro) ~CCapituloLibro()

- **CArticulo:** Esta clase tiene una propiedad tipo "*CRevista"* con la información general de la revista a la que pertenece y cuatro propiedades de tipo "*int*" correspondientes al número de la revista, el volumen de ese número, la página donde comienza el artículo y la página donde finaliza. Tiene varios constructores para adaptarse a las necesidades de la aplicación

CArticulo : CDocumentoBase

```
CRevista Revista
int Numero
int Volumen
int PaginaInicial
int PaginaFinal

CArticulo()
CArticulo(long id, string titulo, int anyo,
          CRevista revista, int numero, int volumen,
          int paginaInicial, int paginaFinal,
          string keywords)
CArticulo(long id, string titulo, CAutores autores,
          int anyo, CRevista revista, int numero,
          int volumen, int paginaInicial, int paginaFinal,
          string keywords)
~CArticulo()
```

- **CActaCongreso:** Esta última clase tiene la propiedad CRevista con la información general de la revista donde ha sido publicado. Además tiene otra propiedad tipo "*string*" con el código identificativo que le asigna al acta quien lo publica. Tiene varios constructores para adaptarse a las necesidades de la aplicación

CActaCongreso : CDocumentoBase

```
CRevista Revista
string Codigo

CActaCongreso()
CActaCongreso(long id, string titulo, int anyo,
          CRevista revista, string codigo,
          string keywords)
CActaCongreso(long id, string titulo, CAutores autores,
          int anyo, CRevista revista, string codigo,
          string keywords)
~CActaCongreso()
```

Finalmente el diagrama de clases y herencias de los tipos de datos quedaría así:

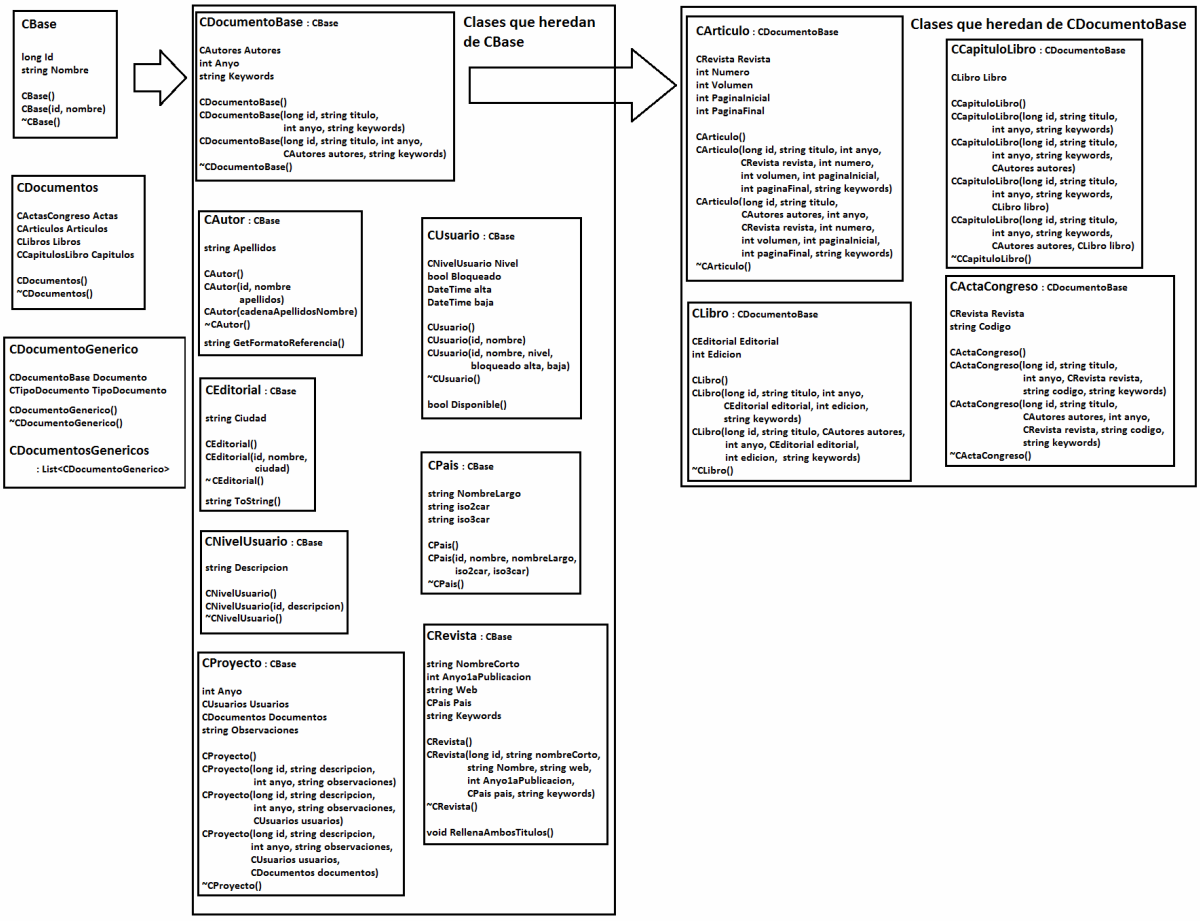


Figura 3.8 Diagrama de clases y herencias.

3.2.2 Diseño de la capa de Datos.

La aplicación ReferenciasBibliográficas necesita realizar muchas operaciones contra la BBDD. Desde la inclusión de bibliografía nueva, la edición de los datos, la creación de proyectos nuevos y el posterior relación de bibliografía con él hasta el mantenimiento de tablas maestras, todo pasa por una operación de base de datos. Existen cuatro operaciones básicas: Consulta, inserción, edición, borrado. Todo ello se realizará en esta parte del código. Para incluir todos los archivos que formarán la capa de datos se define el *"Namespace"* DAL.

La clase CDAL es la clase que se encarga de todo lo enunciado en el párrafo anterior. Para poder funcionar debe incluir los espacios de nombre System.Data y System.Data.OleDb. Esta clase contará con las siguientes propiedades:

- **m_conn_str:** Propiedad privadas de tipo string que guardará la ruta donde está la BBDD. Esta aplicación usa una base de datos de MS Access, con lo cual ahí se guardará un texto de formato `"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=<ruta de la BBDD>"`.
- **m_transaccion:** Propiedad protegida de tipo OleDbTransaction que se ocupa del caso en que se necesite ejecutar operaciones de BBDD de forma transaccional. Estos casos son en los que para realizar una operación se necesita que todas las suboperaciones que conlleva se ejecuten todas o ninguna, y si alguna falla se anulen todas ellas. Esta propiedad nos proporciona lo necesario para conseguirlo.
- **m_connection:** Propiedad protegida de tipo OleDbConnection que contiene una conexión a BBDD. A esta conexión habrá que indicarle la "*ConnectionString"* definida anteriormente, m_conn_str. Esta clase es a través de la que se realizan todas las operaciones de BBDD.

La clase CDAL incluye los siguientes métodos:

- **LeerConfiguracionBBDD:** Es un método privado que va a leer el archivo de configuración "*config.xml"*. Este archivo contiene la ruta y el tipo de motor de BBDD. Dentro de la etiqueta <app> se encuentra otra etiqueta denominada config_bbdd que incluye dos parámetros: provider y dataSource. Este método es capaz de obtener la cadena de conexión a partir del archivo xml y devolverla en un "*string"*.
- **ComenzarTransaccion:** Es un método público que la clase ofrece para indicar que va a empezar una transacción.
- **HacerCommit:** Es un método público que la clase ofrece para indicar que la transacción ha finalizado y que todo ha salido bien.
- **HacerRollback:** Es un método público que la clase ofrece para indicar que alguna de las operaciones de la transacción ha salido mal y hay que echar para atrás todo el proceso.
- **ProbarConexion:** Es un método público que la clase ofrece para informar a quien la usa el estado de la conexión a BBDD. Si la conexión está activa devolverá un "*true*", en caso contrario devolverá "*false*".

La clase también ofrece métodos protegidos para que las clases que hereden de ella las usen para ejecutar alguna *"query"* en la BBDD, construir alguna *"query"* o interpretar un valor:

- **EjecutarSelect:** Método que ejecuta una *"query"* que contiene una select y devuelve en un DataReader el resultado. Es necesario que está configurada la conexión.
- **EjecutarSelectReader:** Método que ejecuta una *"query"* que contiene una select y que espera un resultado muy grande: un archivo, un binario….
- **EjecutarEscalar:** Método que ejecuta una *"query"* en que se espera sea devuelto sólo un número.
- **Insertar:** Método que ejecuta una *"query"* que contiene una operación de inserción en BBDD. La *"query"* se pasa como parámetro. Si el parámetro "booleano" devolverIDNuevo está a true la función devolverá el ID de la nueva inserción que se ha generado automáticamente; si está a false la función devolverá el nº de registros a los que ha afectado la operación. El método da la opción de poder pasarle valores como *"OleDbParameter"* por si es necesario, como en el caso de pasar valores que son binarios o muy grandes.
- **EjecutarQueryGeneral:** Es un método que sirve para ejecutar cualquier otra *"query"* que no sea una inserción o una consulta: modificación o borrado. También da la opción de pasar parámetros como *"OleDbParameter"*.

Una vez desarrollada la clase CDAL, para cada tipo que existe, que a su vez está basado en las entidades de la BBDD existirá una clase que se llamará igual que el tipo seguido de la partícula DAL. Todas estas clases heredan de la clase CDAL e incluyen todas las operaciones de BBDD sobre ese tipo que la aplicación requiera. Aunque cada tipo de datos tiene alguna operación propia que sólo se realiza para él todas las clases constarán de ciertos métodos que se repetirán para todos.

- **CargaTodos:** Un método que hace una select en que pide todos los registros disponibles. Se devolverán en una List de ese tipo.
- **CargaPorID:** Un método que devolverá un solo registro que habrá sido identificado por su ID.

- **Insertar:** Un método que recibe un objeto de la clase con que se está trabajando y tiene que insertarlo en BBDD.

- **Borrar:** Un método que a partir de un ID borra de la base de datos el registro correspondiente.
- **Modificar:** Un método que modifica un registro con información nueva.
- **CargaObjetoDeDataRow:** Se ha diseñado un método que rellene el objeto con la información contenida en el *"DataRow"* que devuelve el método que realiza la *"Select"*. Esto obliga a poner un "alias" a todos los datos que se extraen de la BBDD y que sean siempre igual.

3.2.3 Diseño de la capa de Negocio

La capa de negocio es una parte del código que hace de mediadora entre la capa de aplicación y la capa de datos. Es la que hace posible que la capa de aplicación se abstraiga de la forma en que se obtienen los datos que ella necesita, esto hace posible que si hay que cambiar la base de datos, sólo habría que cambiar la capa de Datos y el resto se podría aprovechar; al igual que en el caso en que se decidiera migrar este proyecto a una aplicación WEB sólo habría que cambiar la capa de aplicación.

Se crea un "*Namespace"* Negocio donde se irán metiendo todos los archivos que compondrán la Capa de Negocio. Por cada uno de los archivos que se crearon para las operaciones con las entidades de la BBDD en la Capa de Datos se creará uno con el mismo nombre pero comenzando con la partícula "Negocio." seguido del nombre del archivo (por ejemplo "Negocio.Articulo.cs". La clase se denomina <nombre_tipo>Negocio (por ejemplo ArticuloNegocio) y cada una de ellas incluirá una instancia de la clase correspondiente en el "*Namespace*" DAL, a través de la cual conectará con la BBDD y realizará las operaciones. Por lo general las funciones declaradas en la Capa de Negocio serán meras copias de sus homónimas en la Capa de Datos y devolverán una llamada a dicha función, salvo en casos en que la operación demandada conlleve varias operaciones como sería el caso de la inserción de uno de los cuatro documentos soportados:

Primero, en caso que los autores del documento no existan en BBDD habrá que darlos de alta, también en el caso de la revista o la editorial que publique el documento, finalmente se realizará la inserción del documento a través de la capa de datos.

3.2.4 Diseño de la capa de Aplicación

Hasta aquí se ha visto el diseño de los objetos y módulos incluidos en la DLL ReferenciasLib.dll. Todo esto se exportará para ser usado por la capa de aplicación. La capa de aplicación contiene todos los formularios que necesita la aplicación para funcionar y funcionalidades no incluidas en la DLL.

- **Namespace Utilidades:** Se ha generado el archivo Utilidades.cs que incluye el "*Namespace*" Utilidades para ir guardando todas las funcionalidades de uso general que pueden ser usadas por cualquier formulario. Se dividen en 4 clases:
    - **Utilidades:** Clase que contiene métodos para manejar elementos de formularios tales como los "*DataGridViews*" que son tablas que se insertan en los formularios para representar información contenida en tablas de BBDD o resultados. Contiene los siguientes métodos:
        - **AnyadirColumnaDataGridView:** Método que genera una columna nueva para un "*DataGridView*". Existe una sobrecarga de esta función y hay dos versiones, una en que se puede definir el nombre de la columna, el texto que aparece en la cabecera y otro en que además de todo lo anterior, puede definirse el ancho de columna.
        - **CeroEsNada:** Método que se usa cuando se está rellenando el valor de una celda de un "*DataGridView*" y no interesa que cuando el valor es 0 salga representado, este método lo detecta y cambia el valor.
    - **UtilidadesXML:** En este PFC, la aplicación usa un fichero XML para saber dónde está la base de datos que va a usar. Para ello hay que usar una pequeña clase que permite usar estos ficheros. Para ello se incluye el "*Namespace"* System.Xml donde están definidas algunas

clases tales como XmlTextReader, XmlDocument y XmlNode. Al constructor habrá que pasarle la ruta del archivo al instanciar la clase. Se han implementado los siguientes métodos:

- **GetAtributoNodo:** Método al cual se le pasa el nombre del nodo del XML y el atributo que requerido y devuelve el valor en formato "*string*". Al buscar en el archivo XML dónde se sitúa la BBDD habrá que usar esté método.
- **SetAtributoNodo:** Método que escribe el valor de un atributo de un nodo. Toda esta información se pasa por parámetros. La aplicación ofrece la posibilidad de que el usuario cambie la localización de la BBDD, al realizar este cambio la aplicación cambia el archivo XML y necesita de este método.

- **UtilidadesWord:** Esta aplicación tiene como objetivo generar un archivo de MS Word con lo cual debe contar con utilidades que nos permitan pasar la información que sacamos de la base de datos al papel. Es necesario añadir el "*Namespace*" Microsoft.Office.Interop.Word donde están las clases necesarias para manejar un archivo Word. Hay que definir un objeto de la clase Word.Application que es una clase que maneja la aplicación MS Word y un objeto de la clase Word.Document que se añadirá a la aplicación y se comenzará a rellenar. En el constructor de la clase se abre la aplicación y se le añade un documento y se configura todo para poder comenzar. Los métodos que la clase UtilidadesWord ofrece son:
  - **VisibilidadWord:** Método que hace que la instancia de MS Word sea visible para el usuario. Se usa para que el usuario no vea cómo se genera el documento y poder visualizarlo cuando está acabado.
  - **AgregarTrozoTexto:** Método que agrega un fragmento de texto en un párrafo donde puede elegirse el formato que se va a aplicar: el tamaño de letra, si se quiere negrita y si se quiere que sea cursiva. Este método sirve sólo para trozos de texto, no párrafos enteros, al usarlo se entiende que el párrafo no acaba ahí. Es útil cuando en un párrafo hay fragmentos con distinto formato.

- **AgregarTrozoTextoConFin:** Método que nos permite agregar un fragmento de texto a un párrafo pero después del texto se da por terminado. Así que se podrá elegir el mismo formato que en el método anterior y además de ello se podrá configurar el espacio que se deja después del párrafo y el nº de saltos de línea que se dan.
- **SaltoLinea:** Método que permite realizar saltos de línea en el documento. Habrá que elegir cuántos saltos y el espacio entre cada salto.

- **UtilidadesEncriptacion:** Como se dijo anteriormente, para almacenar las contraseñas de los usuarios se va a usar una encriptación con algoritmo PBKDF2 para lo cual se genera una sal que es una serie de bytes aleatorios y luego esa sal se usa como parámetro para la función de encriptado de la contraseña que el usuario ha elegido. En la tabla se guarda la sal y la encriptación. Para comprobar que la contraseña que, al hacer login, el usuario introduce es correcta se obtiene la sal del usuario y se vuelve a hacer la encriptación y se comprueba que es la misma que la contraseña encriptada que hay almacenada en BBDD. Para realizar estas operaciones se necesita implementar una clase UtilidadesEncriptacion que permita a la aplicación realizar estas operaciones. Para ello se necesita usar el "*Namespace*" System.Security.Cryptography que incluye dos clases que permiten realizar las dos operaciones necesarias:
  - Para generar la sal del usuario se crea el método **GenerarSalUsuario** que se usa la clase RNGCryptoServiceProvider cuyo método GetBytes(byte[]) genera tantos bytes aleatorios como longitud tenga el array que reciba. El estándar de RFC 2898 recomienda que la sal sea de al menos de 8 bytes, para esta aplicación se usará de 16 bytes.
  - Para generar la encriptación a partir de la contraseña y la sal del usuario, llamada "*hash*", usando el algoritmo PBKDF2 se usa la clase Rfc2898DeriveBytes en cuyo constructor habrá que pasarle la sal y la contraseña y su método GetBytes(int numBytes). En el parámetro numBytes se le indicará que genere

un hash de 64 bytes. Este método se usará tanto para generar como para comprobar el usuario.

- **Formularios de la aplicación:** La clase Form es la clase base de la que heredan todos los formularios que se crean para la aplicación. En el "*namespace*" System.Windows.Forms reside la clase Form. Dentro de cada uno de estos formularios puede añadirse cualquier elemento contenido como clase dentro del mismo "*namespace*". La aplicación cuenta con una serie de formularios que sirven para que el usuario interactúe con ella diseñados conforme a las especificaciones de la aplicación que se explican a continuación. Cada vez que un formulario contenga un elemento o clase que no se haya visto aún se procederá a explicar, en caso contrario, se omitirá la explicación para no hacer este documento reiterativo:
    - **Formulario de inicio:** Es el formulario principal de la aplicación, el que se carga al iniciarla y el que llama al resto de funcionalidades y que también servirá para cerrar la aplicación. La apariencia de este formulario depende en el estado en que se encuentre, se distinguen 2 estados "*logado*", "no *logado*". Al iniciarse la aplicación estará en modo "no *logado*" y mostrará sólo las opciones del menú e iconos que se permitan realizar sin iniciar sesión. Una vez realizado el inicio de sesión entrará en el estado "*logado*" y mostrará las opciones de menú correspondientes en función del nivel que tenga el usuario. A este formulario se le ha activado la propiedad "*IsMDIContainer*" lo que significa que e s contenedor MDI, es decir, el formulario que contendrá las ventanas que se generen durante la ejecución de la aplicación y éstas no podrán salir de él.

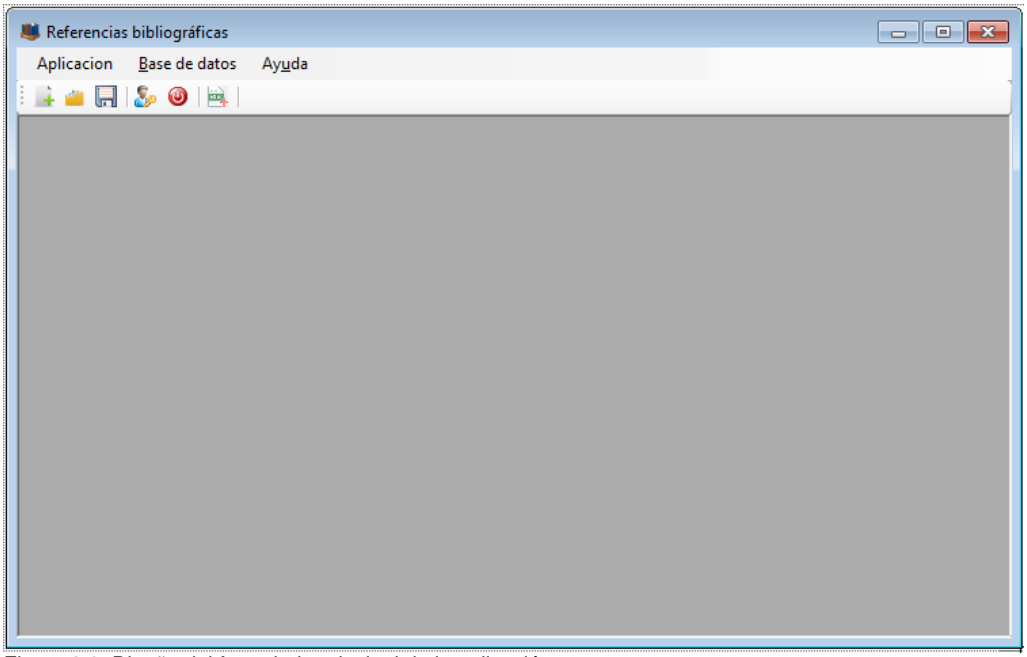


Figura 3.9 Diseño del formulario principal de la aplicación

El formulario cuenta con dos elementos:

- Un elemento "*MenuStrip*". Este elemento es el menú de la aplicación y que contiene objetos de la clase "*ToolStripMenuItem*" en forma de submenús y paneles con opciones. Desde ese menú se puede acceder a todas las funcionalidades de la aplicación. Se le han añadido tres submenús cada uno de ellos con distintas opciones cuyo evento "*click"* se traduce en ejecución de funcionalidad:

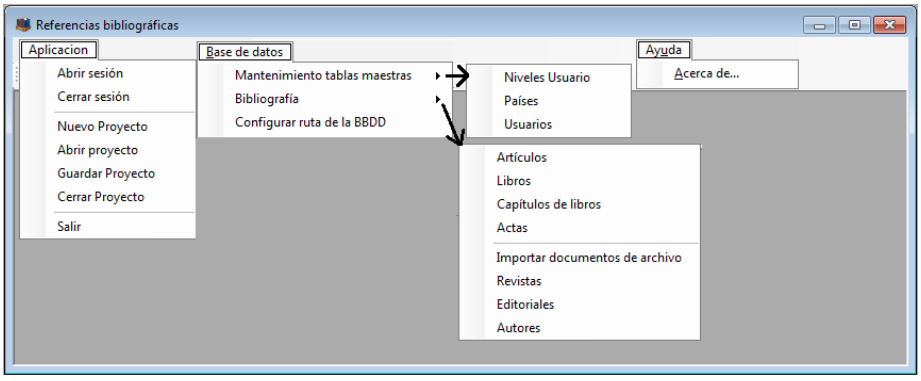


Figura 3.10 Contenido de todos los menús de la aplicación

- Submenú "Aplicación": Contiene las opciones correspondientes a iniciar y cerrar sesión en la que el programa queda configurado dependiendo del nivel de

usuario con el que se inicie el usuario. También contiene las opciones relativas a la creación y edición de proyectos y la operación de cerrar la aplicación.

- Submenú “Base de Datos”: Contiene las opciones relativas al mantenimiento de las tablas de la BBDD de la aplicación, su conexión con ella y la creación, edición y borrado de los distintos documentos que componen la bibliografía de la aplicación, así como la opción de exportado a partir de fichero RIS. Dependiendo del nivel del usuario “*logado*” se mostrarán unas u otras.
- Submenú “Ayuda”: Este submenú sólo contiene la opción “Acerca de …” que muestra la versión de la aplicación, quién la ha desarrollado y para quién.

▪ El formulario también cuenta con un objeto de la clase “*ToolStrip*” una clase que es una barra donde pueden ponerse los distintos iconos, objetos de la clase “*ToolStripButton*” que atajan el acceso a las tareas más importantes. Pueden añadirse también separadores para clasificar los iconos por utilidad:

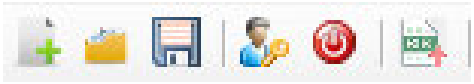


Figura 3.11 Diseño de la barra de iconos del formulario principal

- Grupo de iconos de login: Da opción al usuario a realizar el inicio y cierre de sesión, cuando un icono está activo no lo estará el otro dependiendo de si el usuario ha abierto sesión o no.
- Grupo de iconos de la aplicación: Ofrece un acceso rápido al hecho de crear, abrir o guardar un proyecto
- Icono de atajo a la pantalla de “Importar de archivo RIS”.

○ **Formulario de login:** Formulario que permite al usuario de la aplicación iniciar la sesión con su usuario y contraseña. Comprueba que los datos introducidos por el usuario son correctos e informa de ello al formulario de Inicio.

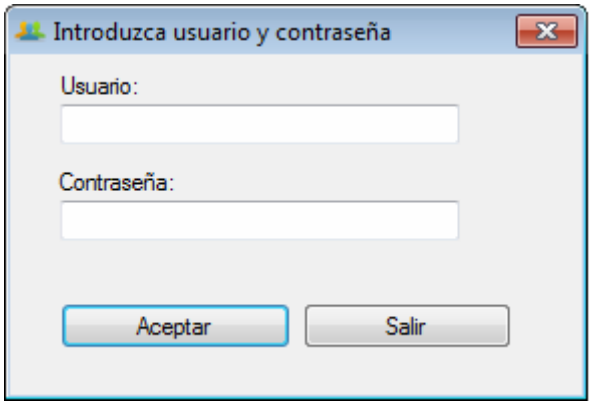


Figura 3.12 Diseño del formulario de "login"

El formulario contiene dos tipos de elementos:

- Clase "*TextBox*". Es uno de los elementos más comunes en los formularios, sirve para que el usuario tecleé texto y poder hacérselo llegar a la aplicación.

  Este formulario tiene dos "*TextBoxes*", uno para introducir el usuario y otro la contraseña. Las propiedades de la clase "*TextBox*" más usadas en esta aplicación son:

  - "*Text*": Acceso al texto que contiene el control. Puede ser escrito por el usuario o puede ser escrito por la aplicación y visto por el usuario.
  - "*UseSystemPasswordChar*": Esta propiedad sirve para cuando no se desea que el texto que el usuario introduce sea mostrado. En su lugar se escribirán asteriscos, pudiéndose recuperar el texto introducido por la aplicación. En este caso, en el "*textbox*" que se usa para la contraseña de usuario se activará esta propiedad.

- Clase "*Button*": Es otra de las clases más usadas en los formularios, son los botones que el usuario puede pulsar para seleccionar acciones tales como aceptar y enviar la información de un formulario, para cancelar, o cualquier otra funcionalidad que contenga. El botón puede tener un texto con la descripción de la acción a realizar como "Aceptar" o "Cancelar" o puede

contener una imagen que defina dicha funcionalidad. Se suele asociar el evento “click” a la funcionalidad que se desea que realice.

Este formulario contiene dos botones el de “Aceptar” y el de “Cancelar”. Cuando se acepta se comprueba que la información usuario/contraseña es correcta y se informa a la aplicación.

- Clase “*Label*”: Un elemento que se usa para poner textos fijos que no suelen cambiar que definen para qué son otros controles sobre los que se colocan. También tienen una propiedad “*Text*” donde en diseño se escribe el texto. En este formulario hay dos elementos de esta clase que definen cada “*TextBox*”.

- **Formulario de apertura de proyecto:** Formulario que se usa para buscar el proyecto que se desea abrir. Aparecerá cuando el usuario pulse la opción del menú “Aplicación / Abrir proyecto” o el icono de la barra. Si el usuario es administrador tendrá acceso a todos los proyectos, pudiendo filtrar por usuario, en caso contrario sólo tendrá acceso a los suyos.

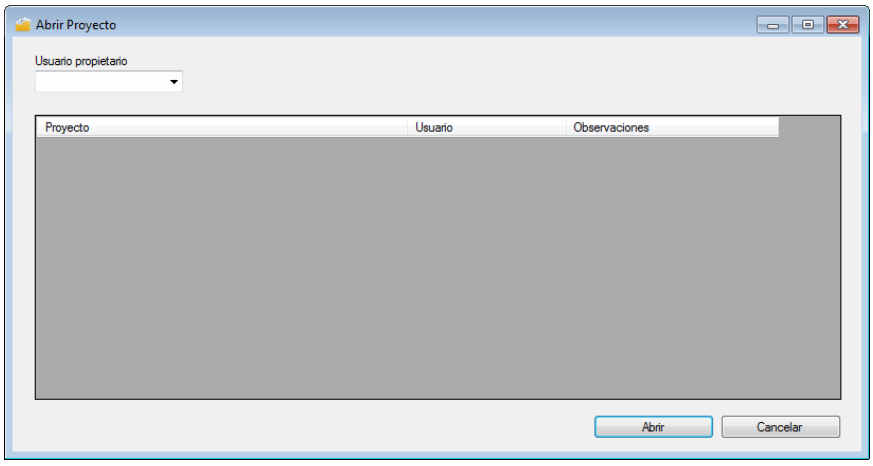


Figura 3.13 Diseño del formulario de apertura de proyecto

En el formulario se encuentran varios tipos de elementos:

- Clase “*ComboBox*”. Esta clase es un elemento de forma parecida al “*TextBox*” pero que al pinchar con el ratón sobre él despliega un panel con varias opciones entras las que podemos

elegir una. En este caso se usa para que el usuario de nivel Administrador puede filtrar por usuario los proyectos que se muestran.

- Clase "*DataGridView*": Este elemento es una tabla donde pueden mostrarse los resultados, pueden configurarse las columnas y luego rellenarse por código las filas. Las filas son objetos de la clase "*DataGridViewRow*". Este formulario y todos los formularios de esta aplicación de mantenimiento de alguna tabla o que necesite un listado con distintas columnas de algo contendrá un objeto de esta clase, en este caso es para mostrar los distintos proyectos disponibles que pueden abrirse.
- El formulario cuenta con dos objetos de la clase "*Button*" para abrir el proyecto seleccionado y para cancelar y cerrar la ventana.

- **Formulario para crear o editar un proyecto:** La aplicación usa este formulario tanto para crear un proyecto nuevo, cuando el usuario llama a la opción del menú "Aplicación / Nuevo proyecto" o en su icono de la barra; como para editar un proyecto que el usuario ha seleccionado del formulario de apertura de proyectos.

  Para guardar el proyecto que contiene el formulario habrá que hacerlo haciendo uso de la opción "Aplicación / Guardar" del menú o con el icono de la barra de tareas. El formulario principal que es el que contiene los proyectos está preparado para trabajar con varias ventanas de este formulario a la vez.

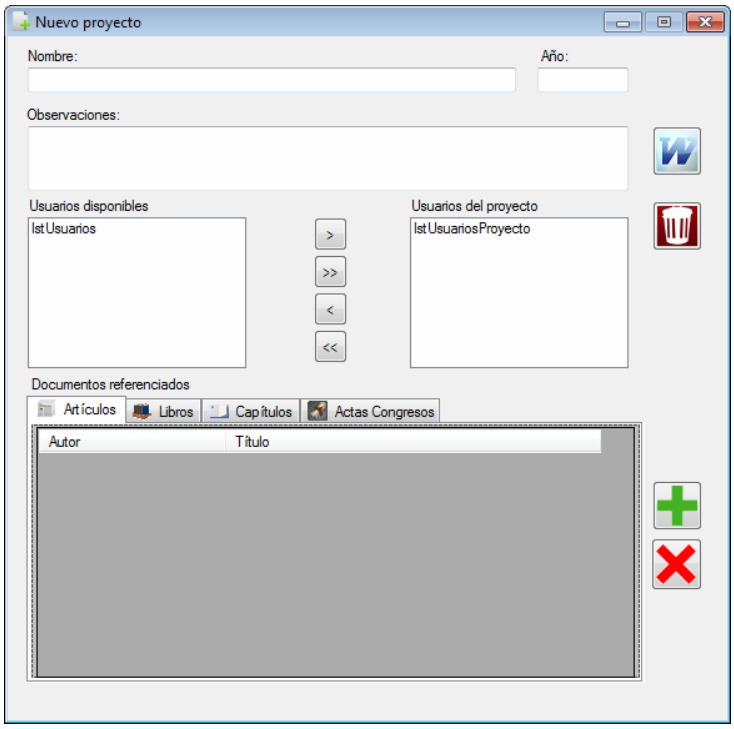


Figura 3.14 Diseño del formulario de edición y creación de proyectos

El formulario cuenta con 4 partes diferenciables:

- Zona de los datos básicos del proyecto: Formado por varios objetos de la clase "*TextBox*". Éstos serán visibles tanto en modo "nuevo" como en modo "edición" el usuario dará nombre al proyecto, guardará el año en que se creó y dará una explicación breve en las observaciones.
- Paneles de asignación de usuarios: Visibles para cualquiera de los dos modos del formulario. Compuesto por dos objetos de la clase "*ListBox*" cuyo elemento representa un panel donde pueden introducirse distintas opciones, cada una en una línea y el usuario puede seleccionar una o varias de esas opciones. Uno de los "ListBoxes" contendrá los usuarios disponibles en BBDD y otro contendrá los usuarios asignados al proyecto. En el momento en que un usuario de BBDD sea asignado al proyecto pasará al panel de asignados y desaparecerá del panel de disponibles y viceversa.

  También en esa parte del formulario se han introducido cuatro

elementos de la clase "*Button*" para realizar las operaciones de "Añadir 1", "Eliminar 1", "Añadir todos", "Eliminar todos".

- Contenedor de pestañas con los documentos relacionados con el proyecto como bibliografía: Esta zona está formada por un objeto de la clase "*TabControl*" donde se definen una serie de pestañas, elementos de la clase "*TabPage*" accesibles pinchando sobre la pestaña. Estos elementos heredan de la clase "*Form*" y forman un trozo de formulario auxiliar donde pueden introducirse elementos del "*namespace*". En este caso cada uno de las cuatro páginas que se han definido contienen un "*DataGridView*" donde muestran los documentos asociados al proyecto. Esta parte es visible también en los dos modos. Guardan los documentos que el usuario asocia al proyecto y es posible añadir y eliminar con los "*Buttons*"que hay a la derecha de los paneles.
- Botones de "Exportar a Word" y Eliminar proyecto: Estos dos elementos de la clase "*Button*" sólo están disponibles para el modo "edición" del formulario. El botón del símbolo de MS Word realiza la exportación de los datos del proyecto a Word y el botón de "Eliminar" borra de la base de datos el proyecto y todas las relaciones incluidas en él.

- **Formulario de selección de documentos:** Es un formulario que se utiliza con dos modos de funcionamiento y relativo a varios tipos de elementos: Mantenimiento de cualquiera de las tablas de documentos y de la de revistas y como ventana de selección de cualquier tipo de elemento citado anteriormente cuando es necesario la elección de uno o varios de ellos desde otra ventana para realizar cualquier tipo de operación con ellos tales como asociarlos a un proyecto o asociarlos a un documento. Es un ejemplo de la reutilización de código para realizar dos cosas distintas o realizar la misma operación sobre varios tipos de elementos.

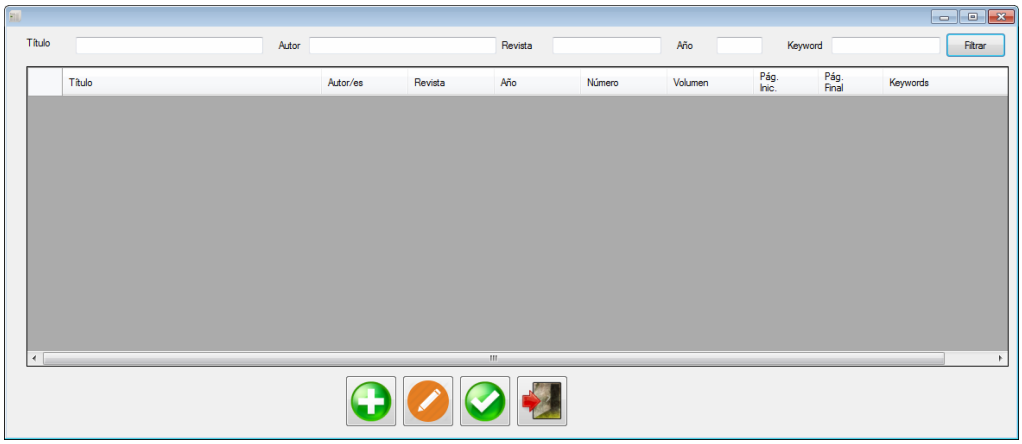


Figura 3.15 Diseño del formulario de selección de documentos

El formulario contiene un "*DataGridView*", una vez que el formulario recibe la información de qué elementos debe mostrar y en qué modo debe funcionar rellena la tabla buscando en BBDD los elementos que le mandan y configurándose en el modo indicado. Si se trata de un mantenimiento ofrecerá funcionalidad para añadir, editar (en estos casos abrirá el correspondiente formulario para añadir o editar el elemento) y borrar elementos nuevos a la tabla y si se trata de una mera selección de elementos ofrecerá funcionalidad para seleccionar los elementos (uno o varios) y devolver qué elementos han sido seleccionados al acabar.

Se ha definido también en la parte superior del formulario funcionalidad para filtrar (con objetos de las clases "*TextBox*", "*Label*" y "*Button*"), dependiendo del tipo de elemento, por algunos campos y a la hora de contar con una BBDD extensa, poder encontrar fácilmente el elemento que se está buscando.

- **Formulario para añadir o editar un documento:** Siguiendo con la reutilización de código, este formulario permite a la aplicación, cuando se está manteniendo cualquiera de las cuatro tablas que guardan documentos poder añadir o editar uno de ellos. Para ello tiene diferentes objetos de la clase "*Textbox*" donde guardar información tal como nombre, año de publicación, número, edición, volumen, página inicio. Tiene también un elemento de la clase "*ListBox*" donde se

agregan o eliminan los autores del documento. En caso de añadir se abrirá una ventana de selección para poder seleccionar uno o varios autores. En caso de artículo o acta de congreso habrá que añadir la revista donde ha venido publicado el documento y en caso de libro habrá que tener en cuenta la editorial donde ha sido publicado dicho libro; esto se seleccionará también de un elemento de selección. Por último hay elemento de la clase "*ListBox*" para ir introduciendo los "keywords" que son palabras clave por las que pueden buscarse o clasificarse los distintos documentos o revistas.

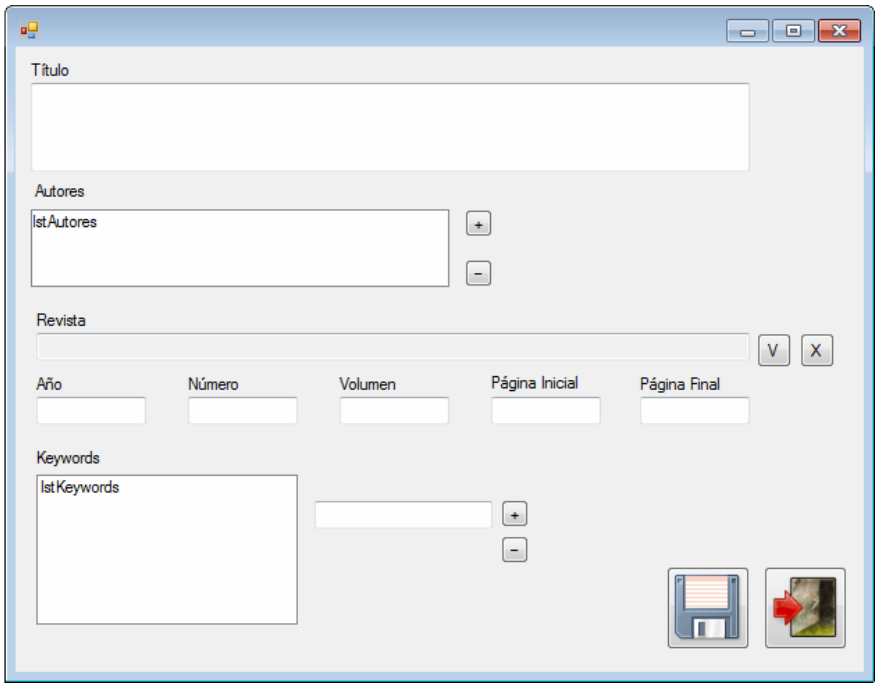


Figura 3.16 Diseño del formulario de añadir o editar documentos

- **Formulario de selección de elemento simple:** Este es otro formulario que sirve de selección de elementos a través de un objeto de la clase "*ListBox*", que requieren información menos detallada para mostrar que el otro formulario de selección, donde hay que mostrar más campos de la BBDD para que queden identificados. Los elementos que se muestran y se seleccionan en este formulario son elementos tales como el autor, editorial, revista, país. El modo de funcionamiento es igual que el anterior, se le configura con el tipo de documento que se necesita y se recogen los "*IDs*" de los elementos que se han seleccionado. Este formulario tiene un botón que facilite el añadido del

elemento que se esté buscando en caso de no encontrarse en la BBDD, de esta forma el usuario no tendrá que cerrar la ventana de selección ni la de edición del documento, dependiendo de qué elemento se esté seleccionando se le abrirá una ventana u otra para agregar. También se ha añadido una funcionalidad para filtrar en caso de tener muchos elementos donde seleccionar.

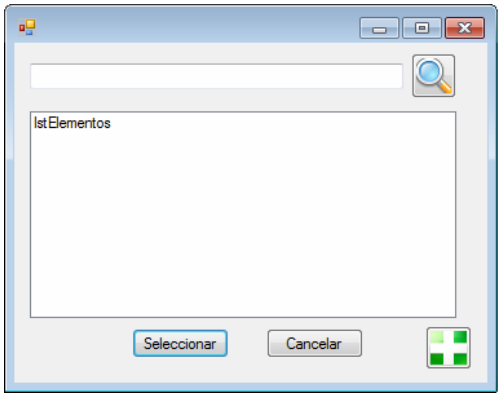


Figura 3.17 Diseño del formulario de selección de elementos simple.

- **Formulario para añadir o editar revistas:** Este es un formulario que sirve para dar de alta o para editar revistas. Contiene elementos de la clase "*Textbox*" que contienen la información que se desea guardar o recuperar, un control que selecciona, usando el formulario de selección simple, el país donde se publica la revista. También la funcionalidad vista anteriormente para definir las "keywords".

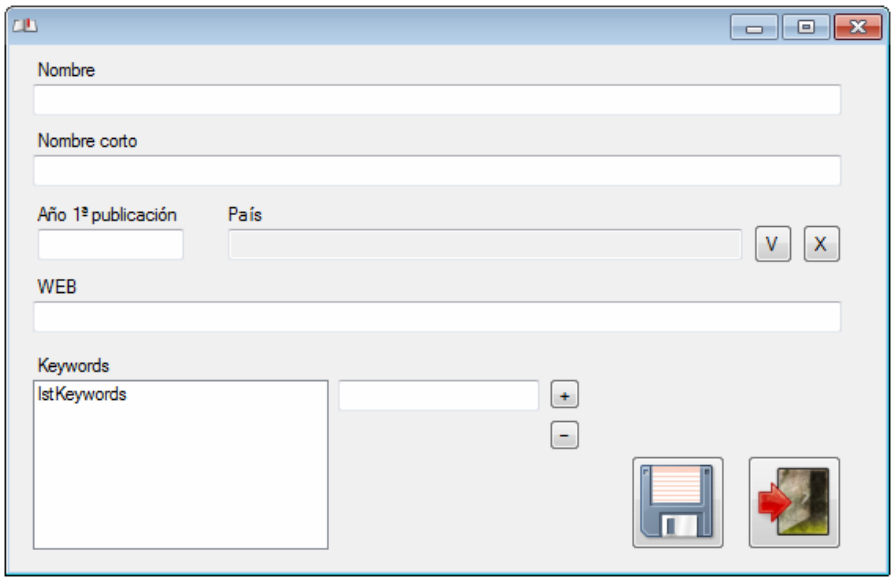


Figura 3.18 Diseño del formulario para añadir o editar revistas

- **Formulario de mantenimiento de autores o editoriales:** Este formulario se utiliza para cuando se quiere hacer el mantenimiento de las tablas de autores o de editoriales ya que los dos elementos tienen pocos campos y la información que hay que mostrar es de tamaño semejante. Al igual que otros formularios de mantenimiento tiene un "*DataGridView*" donde se cargan los elementos de la tabla para poder elegir si se quiere añadir, editar o eliminar. En caso de querer añadir o editar un elemento se llamará a otro formulario más específico para la tarea.

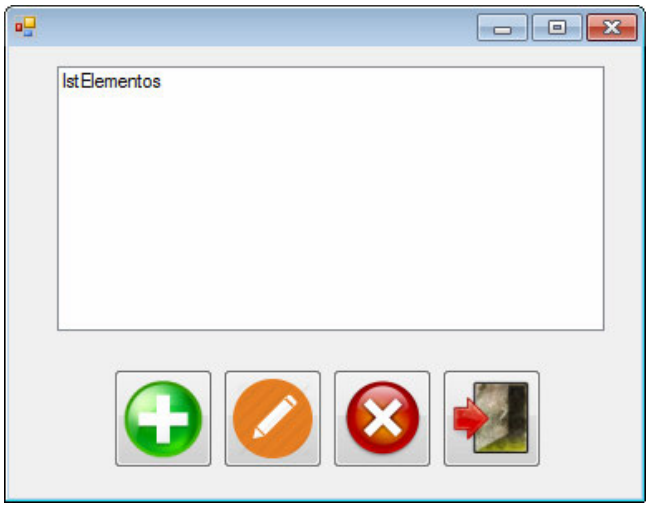


Figura 3.19 Diseño del formulario de mantenimiento de tabla de autores o editoriales.

- **Formulario de mantenimiento de países:** Este formulario se usa para elegir el país que se desea cambiar, añadir o borrar. Esta tabla se ha rellenado usando la información citada anteriormente del ISO 3166-1 y no se recomienda cambiar, pero podría darse el caso de que un país cambiara de código, nombre o desapareciera y hubiera que tocar la tabla.

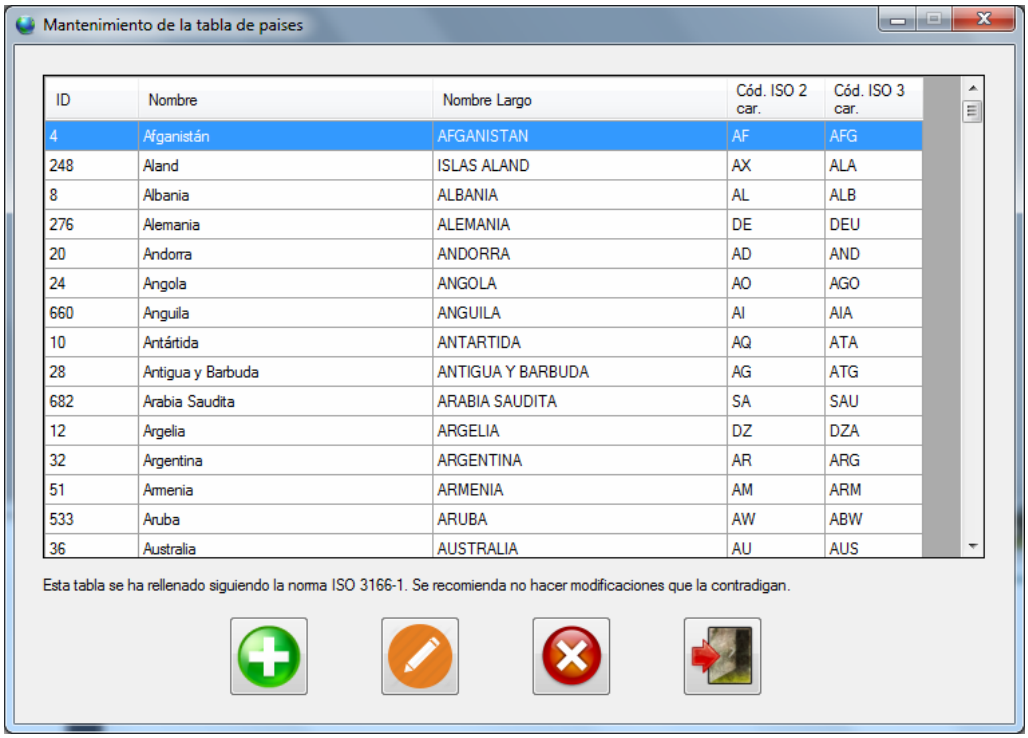


Figura 3.20 Formulario de mantenimiento de la tabla países

- **Formulario de mantenimiento de Usuarios:** Es el formulario que se usa para el mantenimiento de la tabla de usuarios. Es uno de los mantenimientos que más se usará porque es el que se usa para dar de alta y baja a usuarios y para cambiar el nivel que pueda tener un usuario. También tiene la opción de bloquear o dar de baja a un usuario sin borrarle de la base de datos.

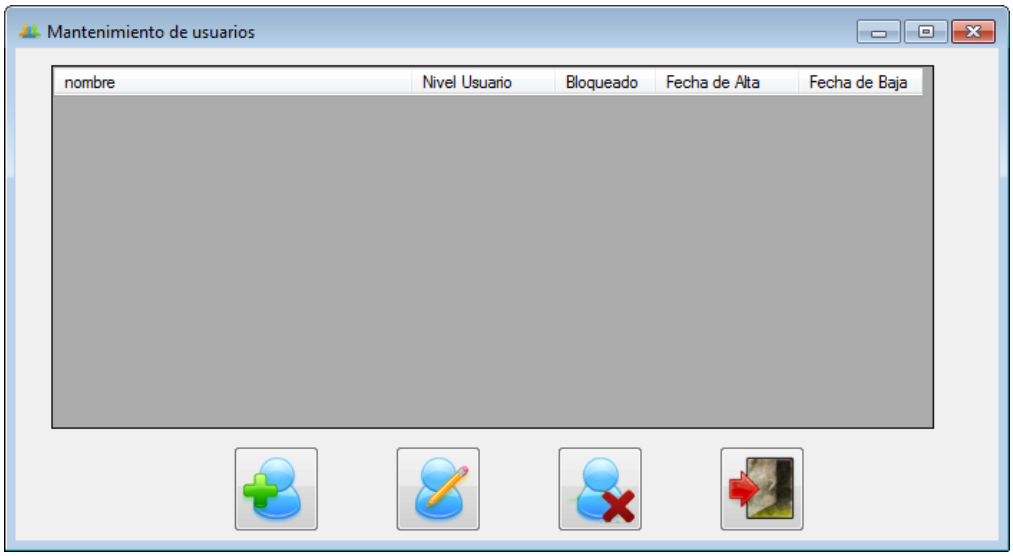


Figura 3.21 Diseño del formulario de mantenimiento de la tabla usuarios.

- **Formulario de creación y edición de autores o editoriales:** Este formulario se usa para crear o editar autores o editoriales, se usa el mismo debido a que los dos elementos cuentan sólo con dos campos y puede reutilizarse el formulario. El funcionamiento de estos formularios compartidos es similar: En caso de creación se abre en blanco y en caso de edición se le pasa el identificador de elemento y se informa del tipo de elemento con que se va a trabajar y se busca la información en BBDD. Al pulsar el botón de “Aceptar” se edita el elemento con el valor que tengan los campos.

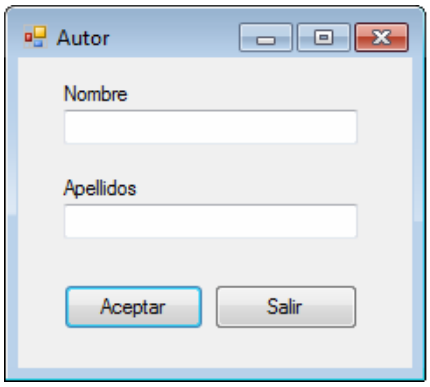


Figura 3.21 Diseño del formulario de creación y edición de autores o editoriales.

- **Formulario de creación y edición de país:** En este formulario, realizado a base objetos de la clase “*TextBox*”, “*Label*” y “*Button*” se introduce o modifica la información de un país y se envía para su almacenamiento en BBDD. Esta ventana será llamada por la de mantenimiento de la tabla país.

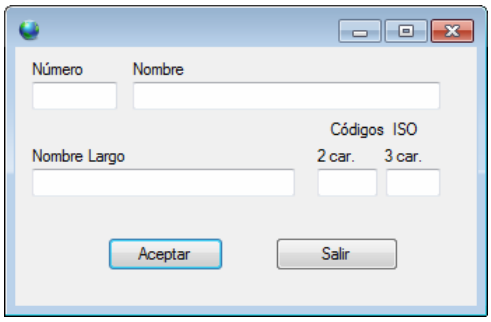


Figura 3.22 Diseño del formulario de creación y edición de países.

- **Formulario de creación y edición de usuario:** Este formulario se utiliza para crear o editar usuarios que hayan sido elegidos del formulario de mantenimiento de usuarios.

  Además de los elementos vistos ya "*Label*", "*TextBox*", "*Button*" y "*ComboBox*" desde los que se recoge parte de la información, este formulario usa un elemento de la clase "*CheckBox*" que sirve para que el usuario realice elecciones que sean "sí" o "no". Al activar el control se le pintará un "check" y al desactivarlo se le borrará. Esta operación activará o desactivará la propiedad "*Checked*" del control. Esto ayuda a la aplicación a saber si se quiere o no activar algo. En este caso es la opción de bloquear o desbloquear a un usuario.

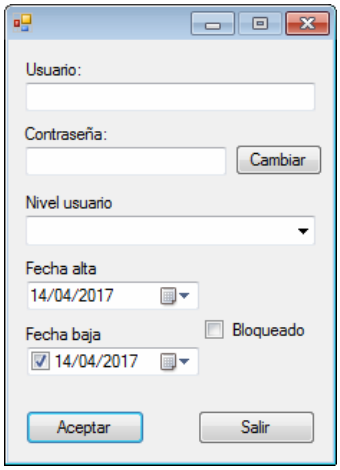


Figura 3.23 Diseño del formulario de creación y edición de usuarios.

El siguiente elemento usado en el formulario es el "DateTimePicker" un control que sirve para que el usuario elija una fecha. Muy útil a la hora de casos donde se necesite seleccionar la fecha de inicio, fin, o elaboración de algo y no haya problemas con el formato porque el control despliega un calendario donde el usuario selecciona la fecha y la aplicación la recibe como objeto de la clase "*DateTime*" y no es necesaria la conversión de formatos a la entrada del dato. En este caso hay dos controles de este tipo uno en que se guarda la fecha en que el usuario es dado de alta en BBDD y otro en el que se elige el momento en que el usuario se da de baja. Este último de los dos controles tiene activada la propiedad "*ShowCheckBox*" para tener la oportunidad de no

tener en cuenta ese control si está desactivado ya que lo normal no es que un usuario esté dado de baja.

- **Formulario de configuración de BBDD:** Este formulario le da la opción al usuario de asociar a la aplicación otro archivo de BBDD distinto al que estaba usando en ese momento o a indicarle a la aplicación cuál es la ruta correcta en caso de que por alguna razón se pierda. Este formulario contiene un objeto de la clase "*OpenFileDialog*" que forma un diálogo para ayudar al usuario a buscar un archivo. Una vez añadido al formulario, este elemento mostrará una ventana de tipo "Explorador de archivos" donde el usuario podrá seleccionar el archivo que busca. Es posible configurar el objeto para que sólo busque por un tipo o varios tipos de archivos y no muestre otros usando su propiedad "*Filter*".

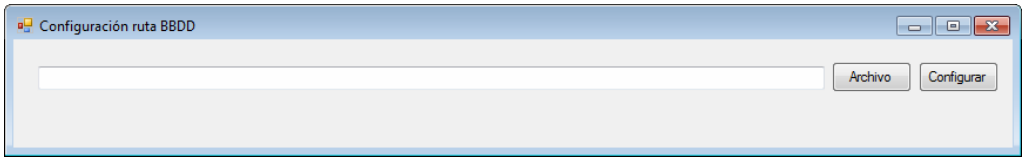


Figura 3.24 Diseño del formulario de configuración de BBDD.

Además de lo anterior, el formulario contiene un "*TextBox*" donde, una vez seleccionado el archivo a través del "*OpenFileDialog*" se escribirá la ruta donde se encuentra dicho fichero. Tiene además dos "*Buttons*", uno para llamar abrir el "*OpenFileDialog*" y otro para, una vez seleccionada la ruta llamar a la funcionalidad que configurará el fichero XML donde se indica la ruta de la BBDD a la aplicación.

- **Formulario para importar bibliografía de un archivo bibliográfico:** Este formulario es parte fundamental de la aplicación y se encarga de realizar la operación de importar bibliografía a través de un fichero de texto ya sea en formato RIS ó CIW.

El formulario contiene un elemento de la clase "*TextBox*" que junto con el "*Button*" y el "*OpenFileDialog*" se encargan de seleccionar el archivo

de disco, que podrá ser RIS o CIW, y al pulsar al "*Button*" de "Exportar" comienza el proceso. El formulario también contiene un objeto de la clase "*ProgressBar*" que implementa una barra de progreso que va marcando lo que lleva de proceso con el fin de que el usuario sea consciente de ello. Este elemento tiene una propiedad llamada "*Value*" que admite valores entre 0 y 100. Cuando se está haciendo un proceso hay que calcular cada cuánto hay que aumentar en 1 el valor de "*Value*" para que coincida con el proceso.

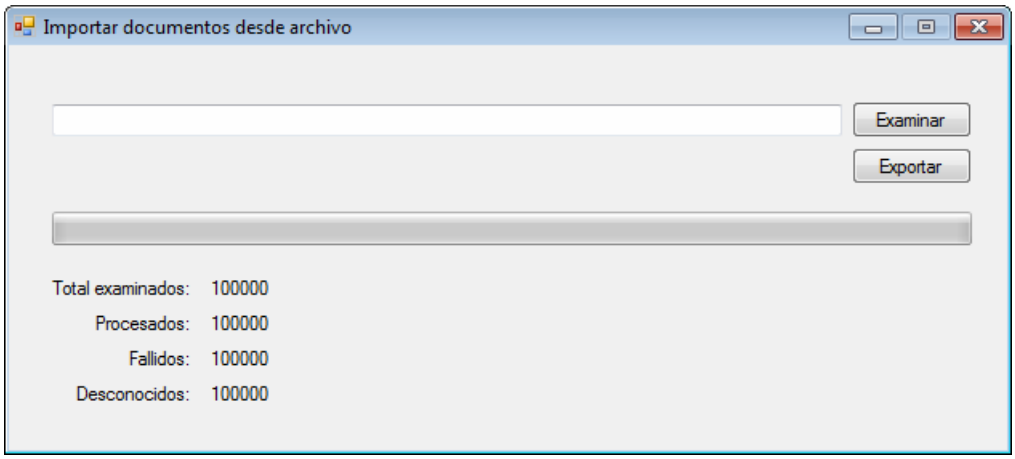


Figura 3.25 Diseño del formulario de importar documentos de archivo

El formulario lleva también unos "*Labels*" donde se muestran los elementos examinados en el archivo, los procesados bien, los fallidos y los desconocidos.

- **Formulario de progreso de exportación a MS WORD:** En el momento en que se está realizando la hoja de bibliografía de un proyecto desde el formulario de edición de un proyecto, dicha operación ha se realiza en segundo plano y no es visible para el usuario hasta que la aplicación no muestra el resultado final. Para evitar que el usuario piense que el ordenador se ha quedado "colgado" mientras realiza el proceso se abre este formulario de progreso que informa de la operación y muestra el proceso mediante un elemento "*ProgressBar*". Al acabar la operación el formulario desaparece solo.

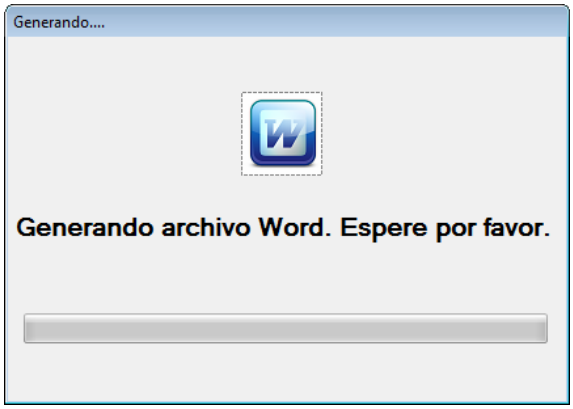


Figura 3.26 Diseño del formulario de progreso de exportación a WORD

- **Formulario "Acerca de …":** Con la intención de informar al usuario de la versión de la aplicación y de quién la ha desarrollado y para quién se crea este formulario que aparecerá cuando el usuario pinche la opción del menú "Ayuda / Acerca de …". El número de versión se saca por código del valor que guarda la variable Application.ProductVersion

Figura 3.27 Diseño del formulario Acerca de…

4. MANUAL DE USUARIO

4.1 Instalación.

La aplicación de este PFC es portable, es decir, no requiere una instalación y puede ejecutarse desde cualquier ruta, incluso desde un "*pendrive"*. Lo único que se requiere para empezar es tener todo el contenido de la carpeta "instalación" en la ruta que se deseé y ejecutarla.

El contenido de la carpeta se muestra en la siguiente tabla

Nombre del archivo	**Descripción**
AppCliente.exe	Es el ejecutable de la aplicación.
Config.xml	Es un fichero xml que permite a la aplicación construir la cadena *"conexionString"* que permite conectar con la BBDD
ReferenciasLib.dll	Es la dll que contiene los tipos de datos y la capa de negocio y acceso a datos de la aplicación.
ReferenciasBibliográficas.mdb	Es la BBDD de la aplicación. Un archivo MS Access.

Tabla 4.1 Contenido de la carpeta Instalación del DVD

4.2 Ejecución de la aplicación

Para entrar en la aplicación se deberá ejecutar el archivo AppCliente.exe.

4.2.1 Creación del usuario Administrador.

Si la aplicación detecta que la BBDD no tiene ningún usuario que pueda administrarla, ésta ofrece la posibilidad de crear un usuario administrador.

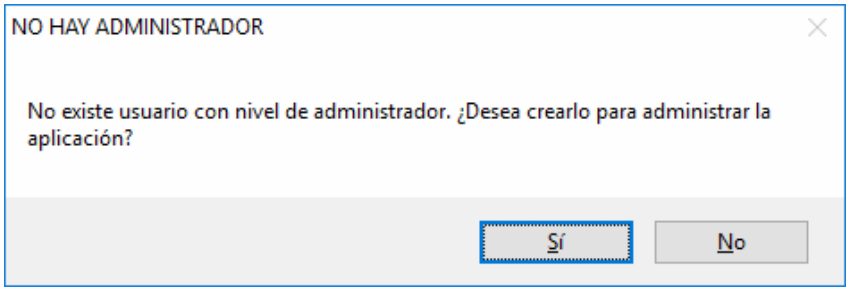


Figura 4.1 Mensaje que informa de la inexistencia de un usuario administrador

Si se elige no, la aplicación se cerrará.

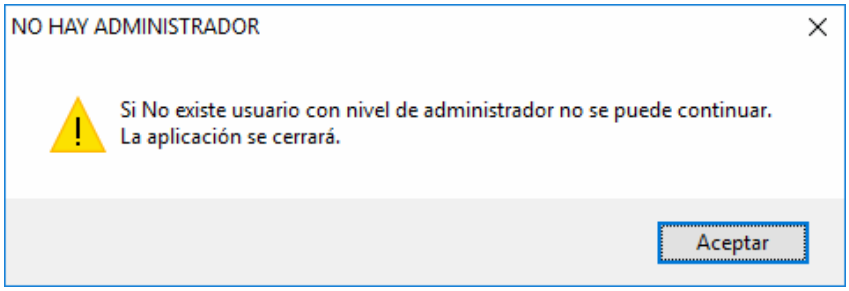


Figura 4.2 Mensaje que informa que la aplicación se cerrará si no se crea usuario administrador.

Si se elige crear el usuario administrador, la aplicación pedirá usuario y contraseña para crearlo y la aplicación seguirá su curso.

4.2.2 Cambio de la ubicación de la BBDD

Cuando aún no se ha realizado el inicio de sesión en la aplicación, el menú “Base de Datos” proporciona la posibilidad de cambiar la ubicación de la BBDD de la aplicación y configurar así el archivo config.xml. Esta opción puede ser útil si se necesita tener la BBDD guardada en un servidor y que todos los que usan la aplicación en su PC puedan enlazar con la misma base de datos y usarla en común.

Para realizar esto hay que pinchar sobre el menú “Base de datos” y seleccionar la opción “Configurar la ruta de la BBDD”. Sale una ventana que contiene la ubicación

actual de la BBDD y dos botones: El botón Archivo que abre un explorador de archivos que permite buscar el archivo de BBDD que queremos enlazar

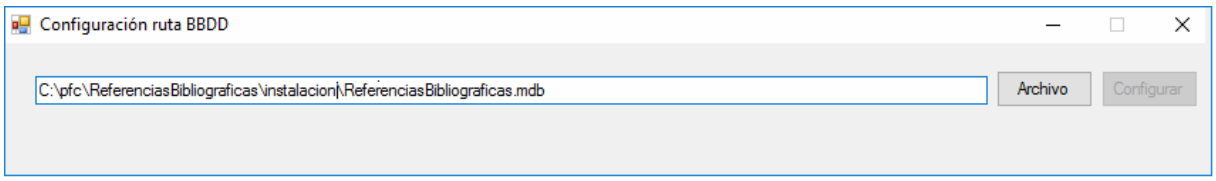


Figura 4.3 Ventana de configuración de la ruta de la BBDD

Cuando esté seleccionado el archivo se activará el botón “Configurar”. Al pulsarle se realizarán los cambios en el archivo config.xml y enlazará la aplicación con la nueva BBDD.

4.2.3 Inicio y fin de sesión

Para poder comenzar a trabajar en la aplicación se requerirá que el usuario inicie sesión contra la BBDD del sistema. Para acceder al formulario de Inicio de sesión se podrá hacer de dos formas: En el Menú Aplicación -> Abrir sesión o pinchando sobre el icono de acceso directo 

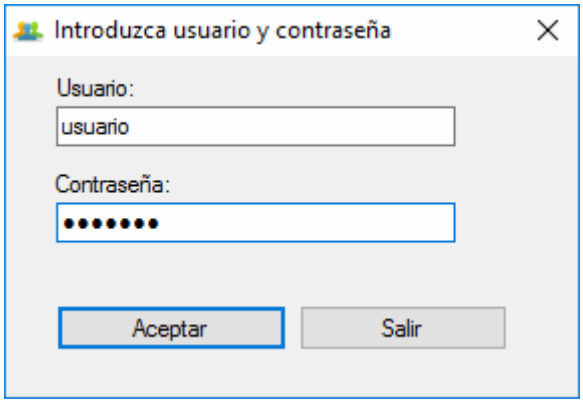


Figura 4.4 Formulario para iniciar sesión

Una vez que se realiza el inicio de sesión correctamente serán visibles las opciones permitidas para cada tipo de usuario.

Cerrar la sesión y volver al estado “*No logado*” puede realizarse de dos formas: En el menú Aplicación -> Cerrar sesión o pinchando sobre el icono  de cierre de sesión.

4.3 Opciones Menú Aplicación

El menú “Aplicación” es el menú que tiene las opciones de inicio y cierre de sesión y las relativas a la creación y edición de proyectos de los que tiene asignado un usuario. También contiene la opción de salir de la aplicación. Seguidamente se detallarán todas y cada una de las opciones del menú “Aplicación”. Se omitirán las opciones de abrir y cerrar sesión porque ya han sido explicadas.

4.3.1 Opción “Nuevo Proyecto”

Esta opción sólo se muestra a usuarios con perfil Administrador o Gestor y se encuentra en el menú Aplicación – Nuevo proyecto y en el icono de acceso directo de la barra de tareas  . Al pulsarse la aplicación abre un formulario de creación de proyecto. Se podrán crear tantos proyectos nuevos a la vez como se necesite y quedarán dentro del formulario.

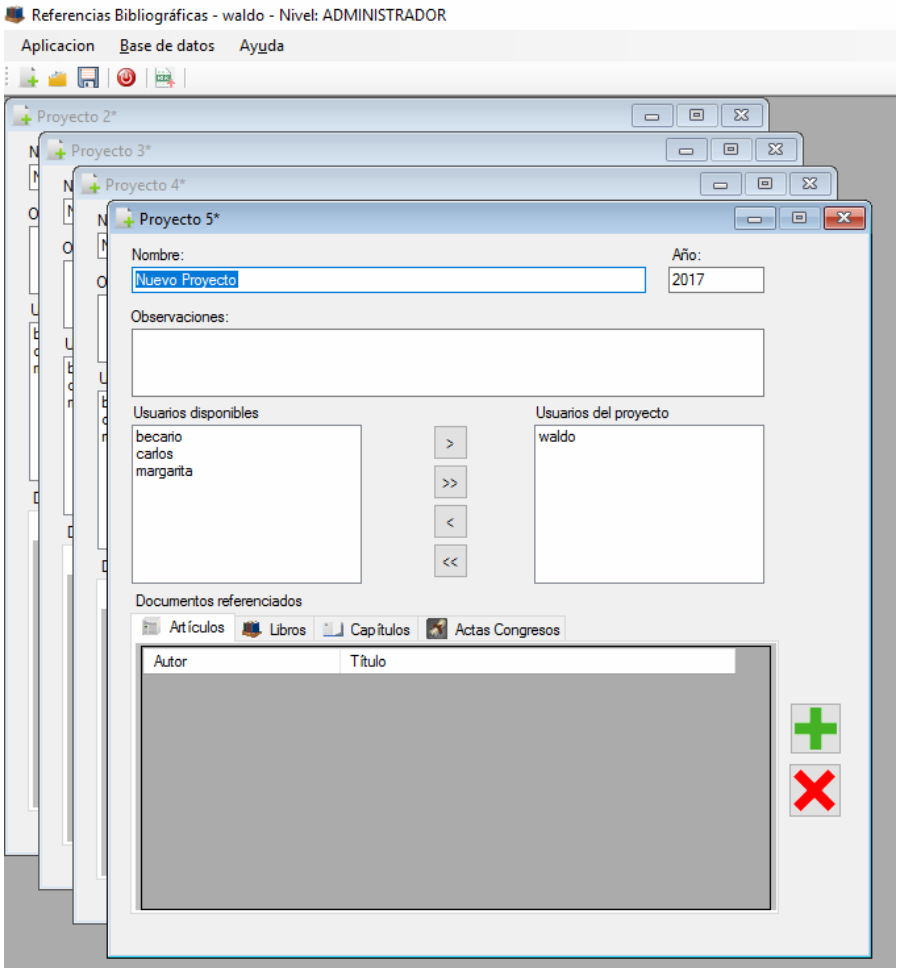


Figura 4.5 Aplicación con varios nuevos proyectos a la vez.

4.3.2 Opcion “Abrir Proyecto”

Esta opción se encuentra en el menú Aplicación – Abrir proyecto y en el icono de acceso directo de la barra de tareas  . Al pulsarse la aplicación abre un formulario que ayuda a elegir qué proyecto de la BBDD se desea abrir.

El formulario tendrá un comportamiento distinto para usuarios de perfil “Administrador”, a los que se permitirá abrir todos los proyectos disponibles y permitirá filtrar por usuario; y para los perfiles “Gestor” y “Usuario” para los que sólo cargará los proyectos que estén asignados para ellos.

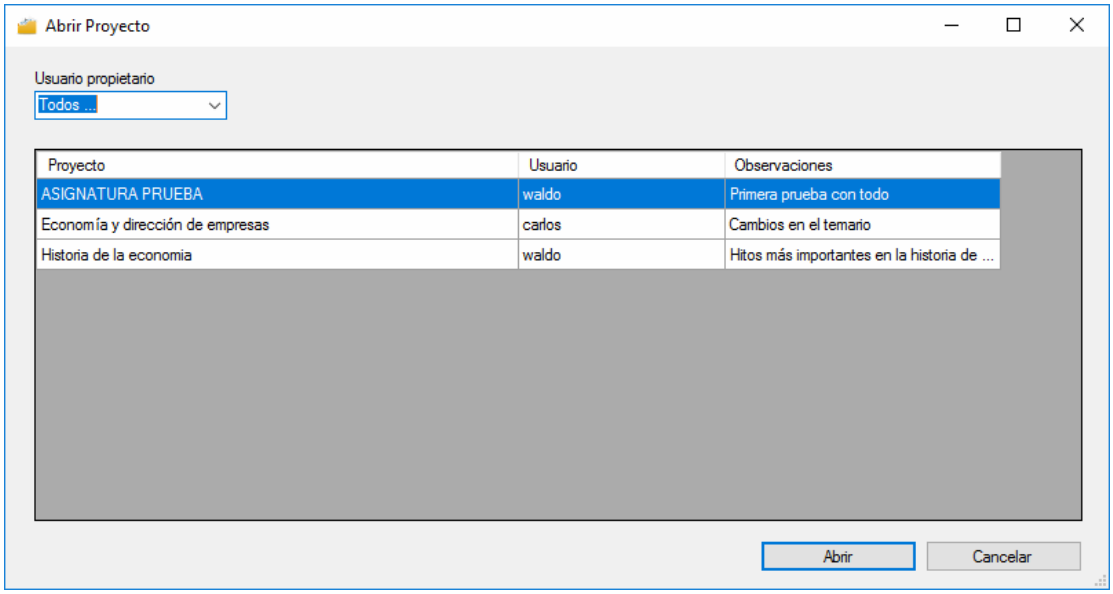


Figura 4.6 Formulario para abrir proyectos.

Cuando se encuentra el proyecto deseado se hace doble clic sobre él o se pincha en el botón “Abrir” y el proyecto se carga en el mismo formulario que cuando se crea nuevo con toda la información que hay sobre él en BBDD. Una vez abierto se podrá modificar y guardar de la misma forma que cuando se crea.

4.3.3 Inserción de la información del proyecto

Una vez que empezamos a crear o a editar un proyecto abierto se puede dividir el formulario en 3 partes:

- **Información general del proyecto:** Aquí se introduce el nombre del proyecto, el año al que pertenece (cuando el proyecto es nuevo aparecerá por defecto

el año en curso) y un resumen general que define en qué consiste dicho proyecto

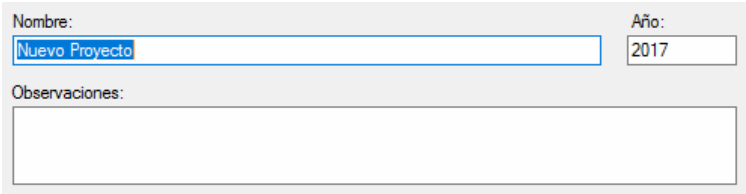


Figura 4.7 Parte de información general del proyecto

- **Usuarios autorizados a entrar en el proyecto:** Esta parte del formulario cuenta con 2 paneles, el de la izquierda son todos los usuarios disponibles en la BBDD y el de la derecha con los usuarios autorizados. Según se van seleccionando usuarios del panel de la izquierda con los botones con flechas los usuarios van desapareciendo del panel de disponibles y apareciendo en el de autorizados. De la misma forma es posible sacar usuarios del panel de autorizados y devolverlos al panel de disponibles.

Figura 4.8 Parte de usuarios autorizados del proyecto

- **Documentos asociados al proyecto:** Esta es la parte del formulario donde se asocian al proyecto documentos que se encuentran en las tablas de documentos de la base de datos. Hay una pestaña para cada tipo de documento requerido.

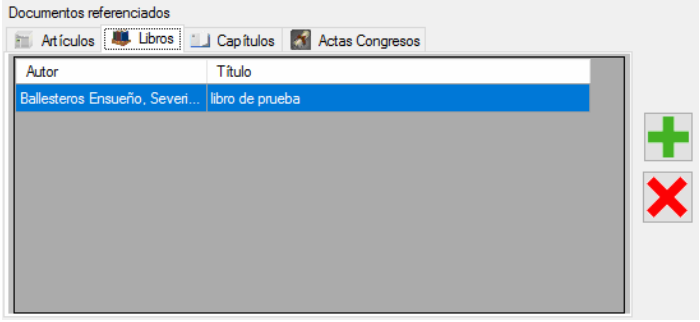


Figura 4.9 Parte de documentos referenciados

Para asignarle documentos hay que seleccionar la pestaña concreta del tipo de documento que se desea rellenar y usar usar el botón de “añadir” . Al pulsarse el botón aparece un formulario que muestra todos los documentos de ese tipo de la base de datos y que pueden filtrarse para encontrar el documento deseado.

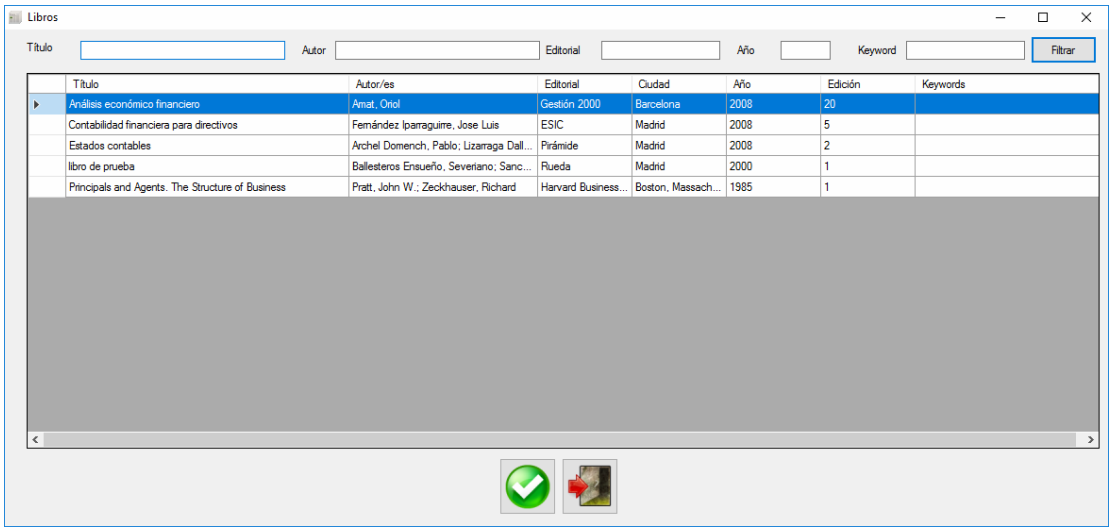


Figura 4.10 Formulario para asociar documentos al proyecto

para desasociar un / os documento/s al proyecto habrá que seleccionarle/os de la lista de la figura 4.9 y pulsar el botón de “borrar” .

En cualquier momento se podrá ir guardando la información del proyecto en Archivo -> Guardar Proyecto o pinchar en el icono de la barra de tareas y el proyecto se creará o se actualizará en la base de datos.

4.3.4 Impresión de la bibliografía y borrado del proyecto.

Una vez que el proyecto se encuentra completamente relleno, ya sea, recien creado y guardado o abierto, y con todas las referencias bibliográficas, a la derecha del formulario aparecerán los botones de “Exportar referencias a Word” y borrar proyecto .

Cuando se pulsa en el botón de exportar a Word se genera un documento en MS WORD con toda la bibliografía ordenadas por orden alfabético del primer autor

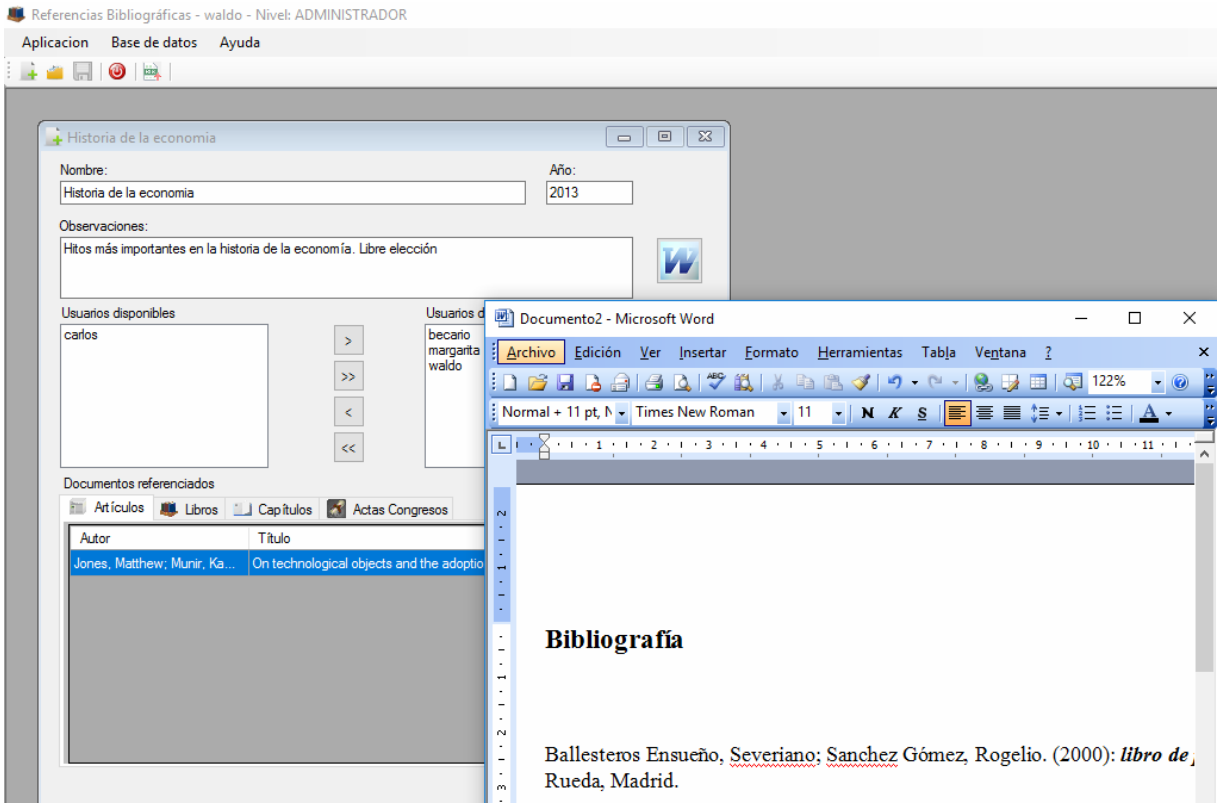


Figura 4.11 Documento en MS WORD generado por la aplicación

Si se pulsa el botón de borrar proyecto, después de las preguntas para asegurarse que el usuario quiere seguir adelante se procede a borrar el proyecto y toda su información de la BBDD. Este botón sólo aparecerá para usuarios con perfil “Administrador” o “Gestor”.

4.3.5 Salida de la aplicación

Al pulsar en la opción Archivo -> Salir se cerrarán todos los proyectos que haya abiertos, preguntando la aplicación al usuario qué hacer con los que estén modificados sin guardar: guardarlos, cerrarlos sin guardar o cancelar.

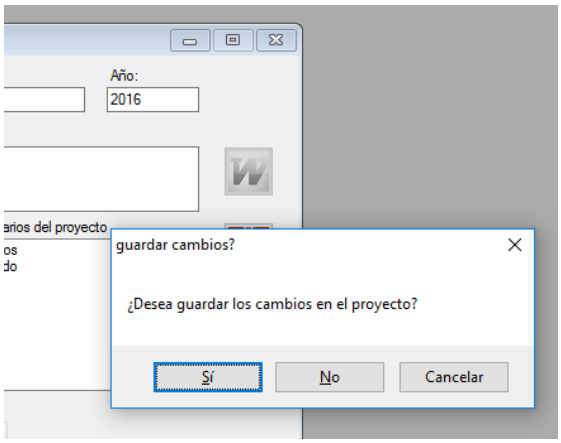


Figura 4.12 Cierre de la aplicación con proyectos abiertos y modificados.

En caso de elegir la opción de “Cancelar” la aplicación detendrá su cierre. Para el resto de casos, se procederá como el usuario estime oportuno y se cerrará el formulario, cerrándose la aplicación al final.

4.4 Menú Base de datos

Este menú contiene todas las opciones en que se modifican las tablas de la base de datos que no sean las referentes a proyectos. Se divide en dos partes, una que realiza el mantenimiento de alguna tabla de la BBDD y otra en la que se añade algún documento de bibliografía.

4.4.1 Submenú “Mantenimiento tablas maestras”

Este submenú está disponible sólo para usuarios con perfil “Administrador”. Existen 2 tablas de la que se puede hacer el mantenimiento desde la aplicación:

- Los países: una tabla que se usa para hacer referencia a los países de publicación de las revistas. Al hacer clic sobre la opción “Mantenimiento tablas maestras->Países” aparece un formulario con los registros de esa tabla con un mensaje que avisa que esta tabla se rellenó siguiendo el ISO 3116-1 que es una tabla con códigos normalizados para cada país.

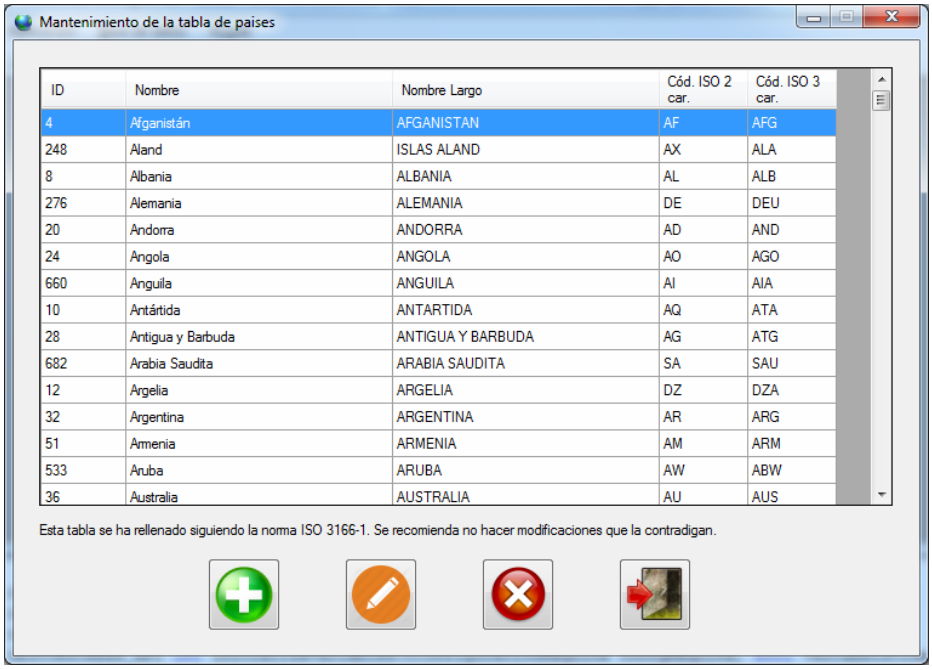


Figura 4.13 Vista del formulario de mantenimiento de países.

En caso de que por distintas causas, en el futuro un país pueda desaparecer, separarse en 2 ó más países o ser absorvido por otro, como ha ocurrido a lo largo de la historia, este formulario permitirá actualizar la tabla. Mientras no ocurra eso, no es recomendable cambiar datos de esta tabla.

- Usuarios de la aplicación: Esta opción permitirá al administrador de la aplicación añadir, borrar, bloquear o modificar usuarios. Al hacer clic sobre “Tablas maestras ->Usuarios” aparecerá un formulario que permite administrar la tabla.

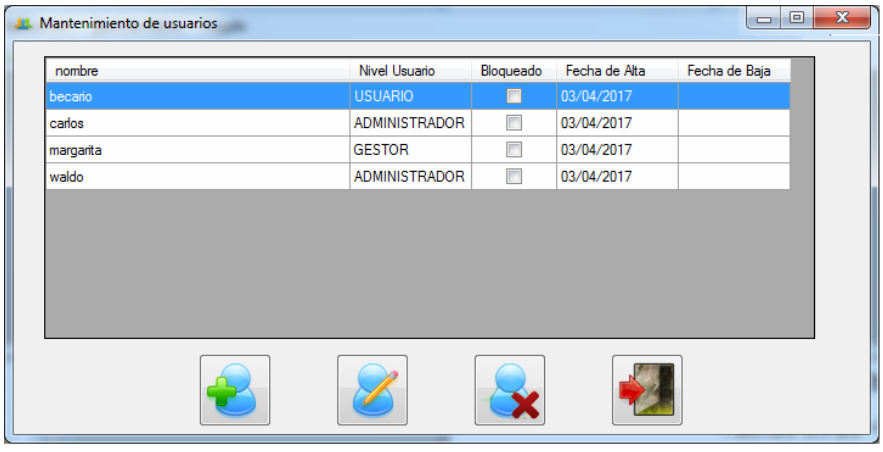


Figura 4.14 Vista del formulario de mantenimiento de usuarios.

El funcionamiento de los formularios es similar al de algún otro formulario ya visto. Hay un botón para crear un registro en el que aparece un formulario donde se introducen los datos. Si señalas varios registros podrás borrarlos con el botón de "Eliminar" y si señalas un solo registro, además de borrarle, se puede modificar, para lo que nos aparecerá el formulario con los datos que podrán ser modificados.

4.4.2 Submenú "Bibliografía"

Este submenú estará disponible sólo para los usuarios con perfil "Gestor" y "Administrador". Las opciones se pueden dividir en 3 partes:

- **Mantenimientos de tablas de documentos:** Sirven para poder incluir más bibliografía en la base de datos de forma manual. Se selecciona qué tabla quiere mantenerse y la aplicación abre un formulario adaptado al tipo de documento con el que se desea trabajar.

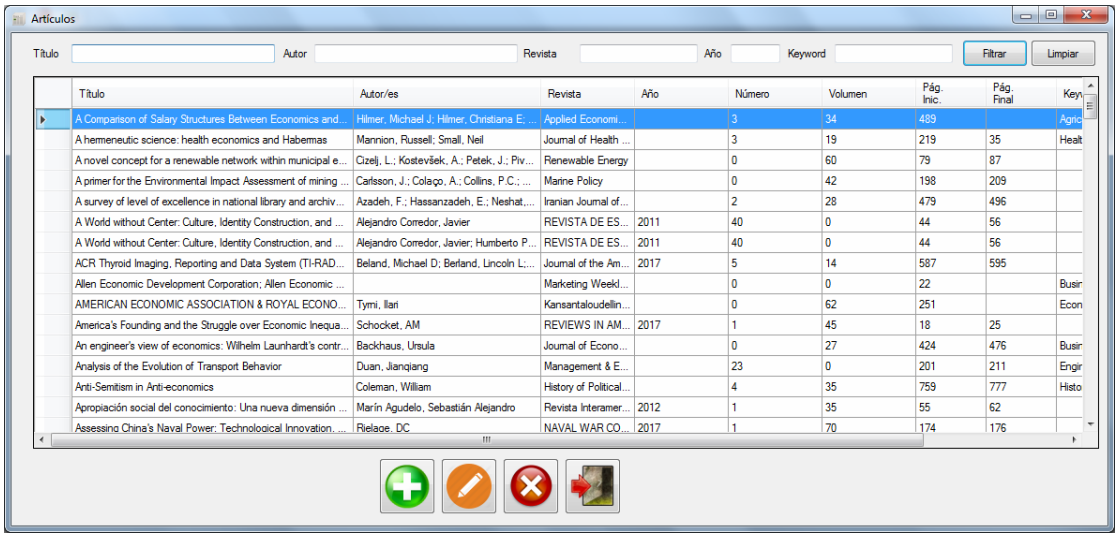


Figura 4.15 Ejemplo del formulario de la tabla de artículos

Puede observarse que cuenta con una tabla donde aparecen todos los documentos y podrán filtrarse, usando los cuadros de texto de la parte superior, por los campos más significativos del documento. En la parte inferior están los botones que permiten las acciones de mantenimiento: Crear, editar y borrar. Para editar o crear se usarán formularios específicos con los campos necesarios de cada documento.

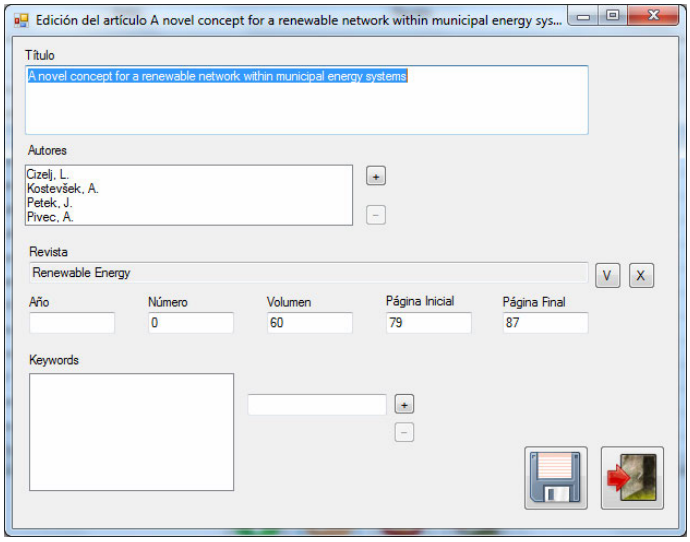


Figura 4.16 Ejemplo del formulario para crear o modificar artículos

- **Importar documentos desde archivo bibliográfico:** Esta es una de las opciones más útiles de la aplicación y a la vez de las más delicadas. Este formulario, que se abre cuando se selecciona la opción “Bibliografía->Importar documentos de archivo” o pulsando en el icono de la barra de tareas, permite al usuario introducir documentos que pertenezcan a cualquiera de los 4 tipos soportados desde un archivo que se ha exportado de los motores de bibliografía que existen en internet. En el formulario hay un botón que abre un explorador de archivos y busca cualquier archivo en formato RIS o formato CIW. Una vez seleccionado el archivo se pinchará sobre el botón “Exportar” y la aplicación abrirá el archivo y lo recorrerá añadiendo a la base de datos cada uno de los documentos que contenga el archivo, evitando así que el usuario deba realizar la aburrida tarea de ir introduciéndolos en la BBDD uno a uno. Lo delicado de esta funcionalidad reside en el poco control que tiene la aplicación de errores de formato del archivo. Deberá ser el propio usuario el que se asegure de cada documento añadido contenga todos los datos requeridos. De lo contrario deberá editarlo manualmente para añadir más información o corregirla.

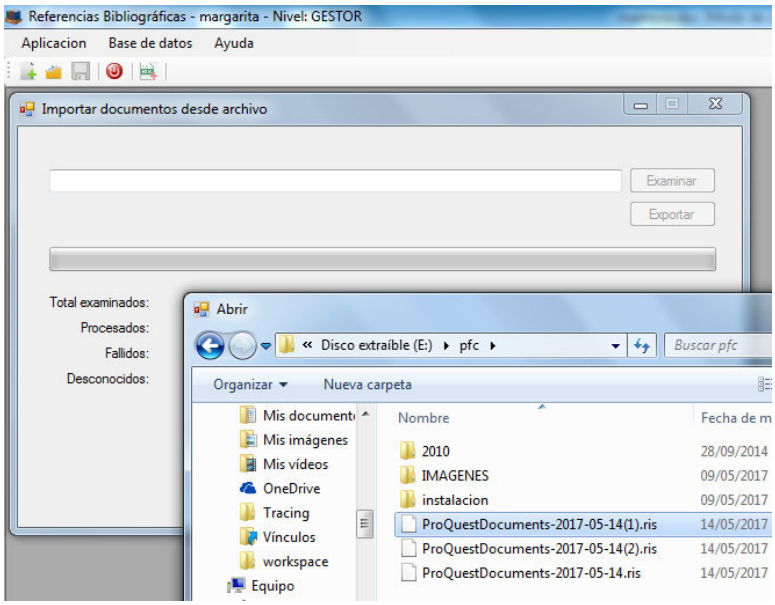


Figura 4.17 Formulario “Importar documentos desde archivo”

Finalmente el formulario informará del número de archivos examinados, procesados correctamente, fallidos (cuyo formato no sea correcto) y desconocidos.

- **Mantenimiento de otras tablas que afectan a los documentos:** En esta tercera parte se realiza el mantenimiento de algunas tablas que contienen información que necesitan los documentos como es el caso de las revistas, las editoriales y los autores. Hay una opción para cada una de ellas en el menú y conducirá a un formulario de mantenimiento de la tabla con su

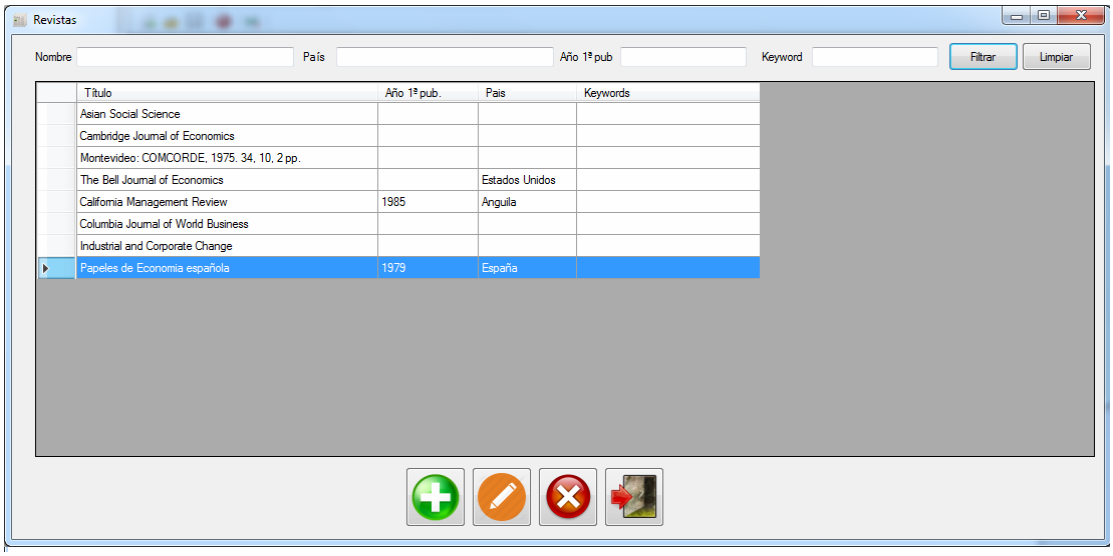


Figura 4.18 Formulario de mantenimiento de Revistas

respectivo filtro. Como en otras pantallas ya explicadas pueden realizarse las opciones de mantenimiento de crear, modifica o eliminar.

4.5 Menú Ayuda

En el menú ayuda hay una sola opción: “Acerca de …”. Esta opción despliega una ventana con la información de la versión de la aplicación y de quién la ha desarrollado. También explica que forma parte de un PFC

Figura 4.19 Ventana acerca de… del menú Ayuda

5. CONCLUSIONES Y MEJORAS

En esta sección final se van a exponer las conclusiones que a las se han llegado durante la realización y documentación del presente PFC y las mejoras que se proponen de cara a futuros trabajos relacionados.

- El uso de la *"framework"* .NET 2010 de Microsoft frente al módulo que incluye MS Access de *"Visual Basic For Applications"* (lenguaje en el que estaba hecho el PFC del que deriva este) supone un gran avance en el diseño de este tipo de aplicaciones, facilita mucho el diseño de ventanas poniendo a disposición del desarrollador más posibilidades a la hora de hacer un mejor diseño desde el punto de vista visual y por supuesto funcional.
- El lenguaje C# de Microsoft .NET es un lenguaje muy cómodo para estudiantes de esta escuela ya que en ella se aprende a programar usando C y posteriormente C++. El lenguaje C# es un paso más allá de C++ al conservar gran parte de la sintaxis de C y C++, combinándola además con la lógica de objetos similar a JAVA que incluye la "*framework*" .NET, lo que hace de C# un lenguaje muy potente y, por supuesto, con más posibilidades que el ya nombrado *"Visual Basic For Applications"*.
- La realización de este PFC ha permitido al autor de este proyecto introducirse en el mundo de las bibliotecas de internet y en los formatos en que estas plataformas exportan los datos que se consultan y las posibilidades que ofrecen. Desgraciadamente, al contrario que el formato RIS, del que se ha encontrado sobrada bibliografía, del formato CIW no ha sido posible encontrar nada y su estudio se ha basado en exportar consultas y averiguar el significado de cada etiqueta y otras características. El ampliar el conocimiento de esos formatos y a la vez de las posibilidades de la aplicación podría ser un frente a investigar para futuros proyectos.

- El algoritmo utilizado para encriptar las contraseñas de usuarios de la aplicación puede considerarse que es demasiado seguro para el nivel de seguridad que a priori va a necesitar una aplicación de uso interno en la universidad, pero ha servido al autor de este PFC para investigar conceptos que serán útiles en su presente y futura carrera profesional.

Seguidamente se van a enumerar una serie de mejoras propuestas para futuros PFCs:

- En las primeras reuniones para la toma de requerimientos los profesores se mostraron interesados en que la aplicación tuviera otra funcionalidad que les permitiera hacer un estudio previo de los posibles documentos que pudieran necesitar para un proyecto, llegando incluso a poder añadir a la BBDD documentos en formato PDF y conformando un mapa del estudio como si de un explorador de archivos se tratara. Finalmente se llegó a la conclusión de que no había tiempo para implementar esa parte. Así que es justo que esta sea la primera mejora propuesta.
- La aplicación de este PFC está diseñada de tal forma que sea fácilmente migrable a una aplicación WEB, más acorde con los tiempos que corren. Sólo habría que sustituir la capa de aplicación, correspondiente a una aplicación Windows por una capa que fuera una aplicación WEB y enlazar con la DLL ReferenciasLib.dll desde un servidor. Toda la capa de negocio y DAL se ha diseñado para que sea perfectamente reutilizable por otro tipo de aplicación.

6. BIBLIOGRAFÍA

A continuación se dan las referencias bibliográficas que se han utilizado en la elaboración del presente PFC:

- GIL PERALES, O. *Modificación de una Base de Datos Access de Referencias Bibliográficas.* Proyecto Fin de Carrera. Universidad Politécnica de Madrid. 2005

- LANA SERRANO, S. *Principios de Bases de datos,* Diatel. 1997.

- CEBALLOS SIERRA, F. J. *Enciclopedia del lenguaje C++.* Paracuellos del Jarama (Madrid): RA-MA editorial, 2009.

- SHARP, J. *Visual C# 2010 (Paso a paso)*, 1 ed. Madrid: Anaya – Multimedia, 2010.

- KALISKI, B. *PKCS #5: Password-Based Cryptography Specification Version 2.0,* Laboratorios RSA, Septiembre 2000.

- MORIARTI K., KALISKI, B., RUSCH, A. *PKCS #5: Password-Based Cryptography Specification Version 2.1,* Internet Engineering Task Force (IETF), Enero 2017.

- *RIS Format Documentation. Adding a "Direct Export" Button to Your Web Page or Web Applicatio.* [documentación]. Thomson Reuters ResearchSoft,

2008.

- BLAZQUEZ OCHANDO, M. *Modelo entidad-relación ER.* [entrada a blog] En: ccdoc-basededatos. Fundamentos y diseño de bases de datos, 20 Febrero. 2014. [acceso 07 Abril 2015]. < http://ccdoc-basesdedatos.blogspot.com.es/2013/02/modelo-entidad-relacion-er.html>.

- *Historia de las Bases de Datos.* [entrada a blog] En: Blog Historia de la Informática. Museo Informática. Universidad Politécnica de Valencia. 4 Enero 2011. [acceso 05 Febrero 2017]. < http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/>.

- ÁLVAREZ, M. A. *Qué es la programación orientada a objetos.* [entrada a blog] En desarrolloweb,24 Julio 2001. [acceso 10 Febrero 2017]. <https://desarrolloweb.com/articulos/499.php>.

- *¿Qué es un archivo DLL?* [documento de ayuda en web] En: Microsoft, última revisión 26 Abril 2017. [acceso en 30 Abril 2017] <https://support.microsoft.com/es-es/help/815065/what-is-a-dll>