PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Software para la creación, almacenamiento y búsqueda en índices virtuales de discos

JOSÉ FRANCISCO MONTOYA MARCHANT

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA.

DICIEMBRE 2007

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Software para la creación, almacenamiento y búsqueda en índices virtuales de discos

JOSÉ FRANCISCO MONTOYA MARCHANT

Profesor Guía: **Iván Mercado Bermúdez**

Profesor Co-referente: **Cristian Alexandru Rusu**

Carrera: **Ingeniería de Ejecución en Informática**

DICIEMBRE 2007

Agradezco a mi familia y amigos, por todo el apoyo, comprensión y confianza que me mostraron durante esta importante etapa de mi vida.

José

RESUMEN

En esta memoria de título, se presenta el desarrollo de un sistema de gestión de archivos, enfocado en el almacenamiento, específicamente el manejo de discos ópticos (CD, DVD).

La finalidad del presente, es el estudio y desarrollo de un software capaz de indexar los principales datos contenidos en los discos de almacenamiento de información digital, para su posterior recuperación “off-line”.

Este documento, considera el estudio realizado sobre catalogación, elecciones y definiciones sobre la metodología, paradigma, tecnología y herramientas, el análisis y diseño del proyecto y finalmente la construcción de la aplicación, tomando en cuenta la elaboración e implementación del mismo.

ABSTRACT

In this memory grade, presents the development of a system for file management, focused on the storage, specifically the handling of optical discs (CDs, DVDs).

The purpose of this is the study and development of software capable of indexing key data contained on the discs storage of digital information for subsequent retrieval off-line.

This paper considers the study of cataloguing, elections on the methodology and definitions, paradigm, technology and tools, analysis and design of the project and finally building the application, taking into account the development and implementation of the same.

Índice

Índice de Ilustraciones

1.- Introducción

En la actualidad gracias a los menores costos de las unidades de grabación y los dispositivos de almacenamiento y así como también a la masificación del acceso a internet*, los usuarios tienen la capacidad de buscar, bajar y almacenar gran cantidad de archivos digitales, creando así sus propios backups o respaldos de archivos para una futura utilización.

***Acceso a internet masificándose**, según la Internet World Stats desde el 2000 al 2007 en Latinoamérica y el Caribe éste ha aumentado en más de un 400%* [1].

Por dispositivos de almacenamiento se entiende, ya sea medios ópticos como CD-ROMs, DVDs, etc. o electrónicos como PenDrives, Mp3s Player, etc., que se usan para guardar y conservar archivos computacionales.

A raíz de esto se presentan nuevas necesidades con respeto al manejo de estos archivos digitales [2], o mejor dicho a los discos en los cuales se encuentran, este problema es muy común para la nueva generación de usuarios de computadores personales. Esta situación los lleva a una constante búsqueda de métodos, ayuda memorias, trucos, etiquetas y su consecuente pérdida de tiempo implementada en cada búsqueda que desean hacer para ubicar un archivo y luego en que disco se encuentran dichos archivos, además en muchas ocasiones se da por perdido un disco cuando en realidad había sido prestado a otra persona, esto produce que se vuelva buscar y guardar los archivos siendo que ya se tiene una copia, lo que aumenta los costos innecesariamente.

Debido a los inconvenientes descritos anteriormente se piensa que es necesario estudiar la situación y posteriormente crear una solución integral para estos problemas, la cual abarque principalmente la indexación de los discos, la búsqueda de archivos en estos, la ubicación física representada virtualmente por el software y la administración de listas e historiales de préstamos de los discos. Proporcionando así una mejora en los tiempos de respuesta cuando se desea ubicar un archivo, ahorrando tiempo y dinero por no tener que volver a bajarlo y grabarlo.

A continuación se muestran los objetivos y requerimientos para así entender que es lo que se busca y espera de este proyecto.

2.- Objetivos y Requerimientos

Para todo proyecto informático se deben establecer ciertos puntos y metas que son importantes tanto para los usuarios como para los desarrolladores. Es así como surgen los objetivos generales y específicos del proyecto.

2.1.- Generales

- Desarrollar un software capaz de indexar los principales datos contenidos en los discos de almacenamiento de información digital, para su posterior recuperación "off-line".

2.2.- Específicos

- Llevar a cabo un estudio sobre la Catalogación, entendiendo claramente sus principios, modelos existentes, terminologías, clasificaciones, etc. de manera de ser capaces de llevar la Catalogación al área de los discos: CD-R, DVD-R, etc. y permitir de esta forma el correcto modelado del problema para desarrollar una aplicación que satisfaga los requerimientos.
- Estudiar formatos de Meta-Información sobre archivos de uso común (Multimedia, Documentos, etc.), de compresión (Zip, Rar, Cab, etc.), para la búsqueda y extracción de datos relevantes de cada archivo, mediante librerías de terceros.
- Definir las características principales que tendrá el software, como se maneja la información, como se almacena usando todas las ventajas de XML, como se busca gracias a la tecnología de XML Query y XPath.
- Implementar un prototipo con las principales funciones del sistema, empezando por el modulo de registro de discos, despliegue de información, búsqueda de archivos y terminando con administración de contactos y administración de la ubicación.

2.3.- Requerimientos Funcionales del Software

El Software debe cumplir 3 objetivos principales (Discos, Préstamos, Ubicación), ahora desde el punto de vista del cliente se pasa a enumerar las características funcionales que debería tener el Software para cumplir con los requerimientos.

- Este sistema debe ser capaz de leer un Disco, catalogarlo y guardar las propiedades de los archivos en el, como propiedades me refiero a: Nombre, Directorio, Atributos, Tamaño, Fecha, Descripción del usuario y Etiquetas ya sean generadas por el usuario o por el propio Sistema.

- También debe guardar la información con respecto al disco, Serial, Sistema de Archivos, Etiqueta de Volumen, Número de Archivos, Tamaño total del disco, Descripción y Etiquetas ya sean generadas por el usuario o por el propio Sistema.

- El usuario podrá buscar un archivo en particular en la base o si prefiere puede navegar a través de todos los discos de forma manual, sin ni siquiera insertar el Disco.

- El usuario será capaz de representar virtualmente donde se almacenan los discos, gracias a una interfaz en la cual podrá definir las características y capacidades de almacenamiento de los contenedores de discos de manera jerárquica, con lo cual podrá ubicar de manera exacta la posición de estos, mediante coordenadas del tipo: Estante 2, Fila 3, Disco 25 de izquierda a derecha.

- Se podrá crear y administrar una lista de contactos los cuales se asocian a los discos cuando estos son prestados, de los cuales se mantendrá un registro de Nombres, Apellidos, Teléfonos, Mails, Dirección e Historial de préstamos.

- El usuario podrá exportar las Bases de discos a varios formatos conocidos por ejemplo: texto plano, CVS, XML, MS Excel, PDF, etc., pudiendo explicitar que discos y que atributos exportar, y luego imprimir estos listados.

- El explorador de discos tendrá una apariencia muy cercana al de MS Windows, en el cual se podrán visualizar los archivos tal cual como si el disco estuviera conectado al PC, respetando los iconos de cada formato e incluso conservando los iconos de los ejecutables.

- Tendrá una Interfaz de Usuario intuitiva y fácil de usar.

- Sera capaz de leer las etiquetas ID3v1 e ID3v2 que contienen algunos archivos multimedia.

- Sera capaz de leer el contenido de archivos comprimidos, por ejemplo: ZIP y RAR.

- Las bases serán guardadas en un formato comprimido, ahorrando espacio, también tendrá la posibilidad de protegerlas con contraseña.

- Se podrá agrupar los discos por categorías al momento de almacenarlos, por ejemplo: Juegos, Música, Películas, Programas, etc., las categorías serán creadas por el usuario.

- La interfaz de usuario del sistema tendrá mínimo dos lenguajes, español e inglés.

- Mostrara el progreso de lectura del disco de manera amigable, mediante varios métodos (Pre-contar Archivos, Por Tamaño de Disco y Archivos, Hibrido) lo que permitirá el correcto funcionamiento con discos multi-sesion.

2.4.- Requerimientos no Funcionales

- Mantener la integridad de la Información.

- La interfaz del usuario deberá ser tan familiar como sea posible a los usuarios que han usado otras aplicaciones de escritorio en Windows. Por ejemplo, se seguirán las guías de la UI para nombrar los menús, botones y las cajas de diálogo siempre que sea posible.

- Responder de manera rápida y precisa.

- Mantener la seguridad de la información que se maneja en el sistema, el acceso a las bases de datos será controlado por contraseñas.

En lo concreto, lo que se pretende es hacer del sistema un repositorio de información sobre el contenido de los discos y datos asociados a éstos que el propio usuario genera, pudiendo consultar dicha información según los intereses y preferencias que satisfagan las necesidades del usuario.

Para esto es muy importante el ingreso de los datos antes mencionados, con el fin de ir configurando con esta información el sistema, para luego hacer uso de dicha información de manera rápida y clara mediante búsquedas que sean amigables para los usuarios del sistema.

3.- Estado del Arte

3.1.- Origen del Problema

El problema de la ubicación y fácil recuperación de archivos digitales se origina a partir de la experiencia del autor en el uso de este tipo de tecnologías, la memoria del ser humano no es capaz de relacionar la cantidad de archivos que pueden llegar a ser almacenados en un disco, mucho menos en cientos de estos, lo que sí se puede recordar es que alguna vez bajamos y almacenamos a dicho archivo en un disco, el concepto de que tenemos ese archivo.

Una manera de aproximarse a su ubicación es saber en cual disco se encuentra dicho archivo, la manera natural de mantener un orden consiste en cada vez que almacenamos archivos tratamos de darle un código al disco y luego ir incrementando ese código a medida que almacenamos mas archivos, en seguida para la fácil ubicación a primera vista sin ningún tipo de ayuda computacional, se suele agrupar de alguna forma los discos según el contenido de estos, ya sean Programas, Películas, Música, etc. Y así también se pueden empezar a sub clasificar en por ejemplo: Películas de Terror, de Acción, Románticas, etc...

Al momento de necesitar un archivo se piensa primero en que clasificación podría caer, se comienza una búsqueda secuencial por los discos, si el disco fuera una sola unidad en cuanto a su contenido por ejemplo: “una película en DVD”, sería fácil su ubicación leyendo algún tipo de etiqueta en el disco mismo, pero cuando los discos son un conjunto de archivos de cualquier tipo asociados solo por el hecho de estar en el mismo disco se vuelve muy problemática su ubicación, los discos no dan la oportunidad de etiquetarlos con todo el contenido a simple vista, en este caso la opción que queda es tomar el disco e insertarlo en la unidad lectora de nuestra computadora, y comenzar a buscar el archivo entre los demás, por lo general no lo encontramos con el primer disco insertado, lo que se vuelve un trabajo muy tedioso el de poner y revisar varios discos para buscar un solo archivo.

3.2.- La Catalogación

Sobre el problema no se encuentran estudios específicos en cuanto a la ubicación de archivos dentro de los discos, pero si en cuanto a la catalogación de contenidos los cuales serán base para la creación y adaptación de la solución a implementar.

De los métodos de catalogación existentes hay dos que se destacan sobre los demás por ser los más usados en sus ámbitos. El Dublín Core o la Iniciativa de Metadatos Dublín Core (DCMI) es el esquema de meta-información más utilizado a nivel mundial según la SEDIC [3]. El MARC, el formato más usado en el mundo de las bibliotecas según la DND [4] y apoyado y mantenido por la Library of Congress in Washington, D.C. USA. [5]

A continuación se explicara en qué consiste MARC y Dublín Core, para luego en las conclusiones definir como ayuda esto al proyecto.

3.2.1.- MARC

3.2.1.1.- ¿Qué es un registro MARC?

Un registro MARC es un registro catalográfico legible por máquina (MAchine- Readable Cataloging).

¿Y qué es un registro legible por máquina?

Legible por máquina: "Legible por máquina" significa que un tipo particular de máquina, una computadora, puede leer e interpretar los datos contenidos en un registro catalográfico. En las siguientes páginas se explicará porque es esto importante y cómo ha llegado a ser posible.

Registro catalográfico: Un registro catalográfico es un registro bibliográfico, o sea, la información que tradicionalmente se presenta en una ficha de catálogo de biblioteca. Un registro puede incluir (no necesariamente en este orden):

1) una descripción del ítem
2) el asiento principal y los asientos secundarios,
3) los encabezamientos de materia y
4) la clasificación o signatura topográfica.

Los registros MARC contienen con frecuencia mucha información adicional.

1) Descripción: Los bibliotecarios compilan la descripción bibliográfica de los materiales mediante la aplicación de las *Reglas de Catalogación Angloamericanas*, 2a. ed., revisión 2002. Esta "descripción" presenta las secciones (compuestas por párrafos) de cada ficha, incluyendo: el título, la mención de responsabilidad, la mención de edición, los detalles específicos del material, la información sobre la publicación, la descripción física, la serie, las notas y los números normalizados.

2) Asiento principal y asientos secundarios: Las RCAA2 contienen también reglas para determinar cuáles serán los "puntos de acceso" a la información del registro (a los cuales llamamos habitualmente "asientos principales" y "asientos secundarios"); y para establecer la forma que éstos adoptarán. Los puntos de acceso son los puntos de recuperación de datos en el catálogo de la biblioteca que los usuarios necesitarán buscar para localizar los materiales.

Dicho de otra manera, las reglas de las RCAA2 se utilizan para contestar preguntas tales como: ¿debe haber, en el caso de un libro en particular, más de un asiento de autor y más de un título?, ¿debe anotarse el título de la serie?, ¿Cómo debe escribirse el nombre del autor?, ¿debe un ítem (sin autor) asentarse bajo título?

3) Encabezamientos de materia (asientos secundarios temáticos): El bibliotecario usa la lista de Sears (*Sears List of Subject Headings*), la Lista de Encabezamientos de la Biblioteca del Congreso (*LCSH*) u otras listas normalizadas de encabezamientos de materia, para seleccionar los encabezamientos bajo los cuales se asienta cada ítem. La utilización de una lista normalizada es importante para asegurar la consistencia y para garantizar que todos los materiales que tratan sobre un tema se asienten bajo un encabezamiento y se encuentren en un mismo lugar en el catálogo.

Por ejemplo, si la lista indica que todos los libros sobre gatos deben asentarse bajo el encabezamiento GATOS; la aplicación de este encabezamiento autorizado elimina la posibilidad de que unos libros se asienten bajo GATOS y otros bajo FELINOS. Aún cuando un libro se titule Todo sobre felinos, el encabezamiento de materia será GATOS, de esa forma todos los libros sobre ese tema se encontrarán en un solo lugar para que el usuario los pueda localizar. El usuario no tendrá que imaginar todos los sinónimos posibles de la palabra que busca.

4) Signatura topográfica: El bibliotecario utiliza los esquemas de clasificación del Sistema Decimal de Dewey o de la Biblioteca del Congreso (LC) para seleccionar la signatura topográfica de un ítem. El propósito de dicha signatura es colocar juntos en los estantes los materiales sobre un mismo tema. La mayoría de los materiales se subarreglan en orden alfabético por autor. La segunda parte de la signatura topográfica, que representa generalmente el nombre del autor, sirve para facilitar dicho subarreglo.

3.2.1.2.- ¿Por qué es Necesario un Registro MARC?

No es posible producir un catálogo automatizado con tan sólo incorporar en una computadora la información contenida en las fichas del catálogo. La computadora necesitará algunos recursos para poder interpretar la información de un registro catalográfico. Un registro MARC contiene una guía de "claves codificadas" de los datos que incluye, las cuales preceden a cada elemento de información bibliográfica.

El espacio designado para cada uno de estos elementos de información bibliográfica se denomina "campo." Los registros, en forma de archivos sencillos de computadora, pueden contener un número fijo de campos y cada campo un número fijo de caracteres.

Sin embargo, para facilitar la adecuada catalogación de los libros y otros materiales de la biblioteca, es necesario que una estructura óptima de los archivos permita que los registros contengan un número ilimitado de campos y que los campos tengan una longitud también ilimitada. Esta flexibilidad es necesaria debido a que no todos los títulos tienen la misma longitud (*La túnica* en comparación con *Alexander y el día terrible, horrible, malo y muy malo*). Algunos libros son parte de una serie y requieren un campo para dicha información, mientras que otros no contienen una mención de serie; por su parte los materiales audiovisuales tienen una descripción física mucho más larga (5 fotobandas : sonido, col. : 35 mm. + manual del maestro), que la de los libros (403 p. : il. ; 22 cm.).

La computadora no puede contar con que cierto tipo de información comenzará y terminará en la misma posición en cada registro; por ejemplo, la mención de responsabilidad no siempre iniciará en el carácter 145 del registro, o terminará en la 207ª posición. Debido a esto cada registro MARC contiene una pequeña "tabla de contenido" del registro que se ajusta a una norma predefinida.

"Señaladores" de los datos: La computadora debe tener los elementos necesarios para poder leer e interpretar un registro bibliográfico.

Si los registros bibliográficos han sido marcados en forma adecuada y guardados en un archivo de computadora, se pueden preparar programas de computación que provean signos de puntuación y estructuren la información en forma correcta para la impresión de juegos de fichas catalográficas o para el despliegue de dicha información en una pantalla de computadora. Se pueden preparar programas que busquen y localicen ciertos tipos de información dentro de campos específicos, y que desplieguen también listas de materiales que cumplan ciertos criterios de búsqueda.

¿Por qué se necesita una norma? Usted podría diseñar su propio método de organización de información bibliográfica, pero con ello podría estar aislando a su biblioteca, limitando sus opciones y embarcándose en un enorme trabajo. La aplicación de las normas MARC evita la duplicación de esfuerzos y permite que las bibliotecas compartan sus recursos de la mejor forma. La decisión de utilizar MARC hace posible que las bibliotecas obtengan información catalográfica previsible y confiable. Si una biblioteca desarrollara un sistema propio que no utilizara registros MARC, no podría obtener las ventajas que ofrece una norma de amplia aplicación cuyo principal propósito es promover la transmisión e intercambio de la información.

La aplicación de las normas MARC permite a las bibliotecas utilizar sistemas comerciales de automatización de bibliotecas para administrar sus operaciones. Existen numerosos sistemas, disponibles para bibliotecas de todos tamaños, diseñados para trabajar con el formato MARC. Estos sistemas son mantenidos y mejorados por los distribuidores, por lo que las bibliotecas pueden beneficiarse con los adelantos de la tecnología de computación. Las normas MARC permiten también que las bibliotecas reemplacen un sistema por otro con la seguridad de que sus datos continuarán siendo compatibles.

3.2.1.3.- La Terminología Usada por MARC y su Definición

Esta sección explica cómo se debe leer, entender y utilizar un registro MARC. Trata sobre lo que los bibliotecarios que usan sistemas automatizados de bibliotecas verán, y necesitarán entender, en las pantallas de sus computadoras, cuando agreguen, modifiquen o examinen registros. Se hará énfasis en aquellas áreas de uso frecuente en la catalogación de libros y materiales audiovisuales en bibliotecas escolares o en bibliotecas públicas pequeñas. Sin embargo lo expuesto en esta sección se aplica igualmente a todas las formas de materiales, incluyendo grabaciones sonoras, **programas para computadora**, mapas, y otros materiales **no libros.**

Algunos cambios recientemente aprobados, y parcialmente implementados, del formato bibliográfico MARC 21 tienen que ver con el concepto de Integración del Formato. La "Integración del Formato" significa que los mismos "señaladores" son utilizados para marcar los datos de los registros de todos los tipos de publicaciones, en vez de tener diferentes conjuntos de "señaladores" para cada tipo individual.

Los nombres distintivos de estos "señaladores" son: *campo*, *etiqueta*, *indicador*, *subcampo*, *código de subcampo* y *designador de contenido*. A continuación se explican dichos términos de MARC 21.

Los CAMPOS se marcan mediante ETIQUETAS.

Campo: Cada registro bibliográfico se divide en unidades lógicas llamadas campos. Hay un campo para el autor, un campo para la información del título, y así subsecuentemente. Estos campos se subdividen en uno o varios "subcampos." Como se mencionó anteriormente los nombres textuales de los campos son demasiado largos para reproducirlos dentro de cada registro MARC, por lo que se les ha representado mediante etiquetas de tres dígitos. (Si bien los catálogos en línea despliegan los nombres de los campos, esto se debe a que dicho nombres son provistos opcionalmente por los programas lógicos del sistema, no por el registro MARC).

Etiqueta: Cada campo está asociado a un número de tres dígitos llamado "etiqueta." Cada etiqueta identifica al campo (tipo de datos) que le sigue. Aún cuando los datos presenten, en forma impresa o desplegados en pantalla, los indicadores inmediatamente después de la etiqueta (dando la impresión de formar un número de cinco dígitos), la etiqueta siempre estará formada por los tres primeros dígitos.

En seguida se presenta un ejemplo de un campo. El número 100 es la etiqueta que lo define como un campo de asiento principal bajo nombre personal (autor).

100 1# $a Pirsig, Robert M.

En los registros MARC se usan con mucha frecuencia el 10% de las etiquetas, el 90% restante se usa rara u ocasionalmente. Aún después de un contacto breve con el Formato MARC se puede escuchar a los bibliotecarios hablar en "MARC-ense." Los bibliotecarios que trabajan con registros MARC memorizan con rapidez los números de las etiquetas de los campos usados con mayor frecuencia de los tipos de materiales que catalogan.

Algunos campos son definidos con mayor detalle mediante INDICADORES.

Indicadores: De las dos posiciones de caracteres que le siguen a cada etiqueta (con excepción de los campos 001 al 009), una o ambas pueden estar ocupadas por indicadores. En algunos campos se utiliza únicamente la primera o la segunda posición; en otros campos se usan las dos, y en algunos como el 020 y el 300 no se usa ninguna. Cuando una posición de indicador no se usa se dice que "no está definida", y dicha posición se deja en blanco. Por regla convencional se representa a los espacios dejados en blanco en los indicadores (no definidos) mediante el símbolo "#".

Cada indicador puede contener un valor numérico del 0 al 9. A pesar de que los dos indicadores juntos pueden parecer un solo número de dos dígitos, son en realidad dos números individuales. Los valores permisibles en los indicadores, asi como su significado, se detallan en la documentación MARC 21. En el ejemplo que se presenta a continuación, los tres primeros dígitos corresponden a la etiqueta (el 245 lo define como el campo del título) y los siguientes dos dígitos (un 1 y un 4) son los valores de los indicadores. El 1 es el primer indicador, y el 4 es el segundo indicador.

245 14 $a The emperor's new clothes / $c adapted from Hans Christian Andersen and illustrated by Janet Stevens.

El valor 1 en el ***primer indicador*** del campo del título indica que habrá un asiento secundario bajo título en el catálogo. En el contexto de los catálogos de fichas esto significa que se imprimirá una ficha bajo el título de este material y que se agregará un asiento de "título" en el trazado. Un primer indicador con valor 0 significaría que el título ocupa el asiento principal; y la ficha sería impresa en párrafo francés y no se requeriría un asiento secundario de título en el trazado (ya que el título funcionaría como asiento principal).

Caracteres que no se indizan en el ordenamiento alfabético: El segundo indicador del campo del título es uno de los indicadores más interesantes; este muestra el número de caracteres al inicio del campo (incluyendo espacios en blanco) que no deberán ser tomados en cuenta por la computadora en el proceso de ordenamiento alfabético. En el título *The emperor's new clothes* el valor del segundo indicador es 4, de manera que los primeros cuatro caracteres (la "*T,*" la "*h,*" la "*e,*" y el espacio) serán ignorados y el título será alfabetizado bajo "*emperor's*".

Los SUBCAMPOS se marcan mediante CODIGOS DE SUBCAMPO y DELIMITADORES.

Subcampos: La mayoría de los campos contienen varios elementos de información. Cada tipo de datos dentro de un campo se llama *subcampo*, y cada subcampo está antecedido por un *código de subcampo*. Los campos 001 al 009 no tienen subcampos.

Por ejemplo, el campo de la descripción física de un libro (definido por la etiqueta 300) incluye un subcampo para la extensión (número de páginas), un subcampo para otros detalles físicos (material ilustrativo), y un subcampo para las dimensiones (en centímetros):

300 ## $a 675 p. : $b il. ; $c 24 cm.

Códigos de subcampo: Los códigos de subcampo están formados por una letra minúscula (ocasionalmente mediante un número) antecedida por un delimitador. El delimitador está formado por un símbolo que se utiliza para separar los diferentes subcampos. Cada códigos de subcampo indica el tipo de dato que le sigue. (La documentación MARC enumera y describe los códigos que son válidos para cada campo en el formato bibliográfico MARC 21).

Delimitadores: Los diferentes programas de cómputo utilizan diversos signos para representar a los delimitadores, ya sea en forma impresa o desplegada en pantalla; se usan, por ejemplo, la daga doble (‡) la arroba (@), el signo de dolar ($), el guión bajo (_), o el signo gráfico"‡". En esta publicación se usa el signo "$" como la porción del delimitador de cada código de subcampo.

En el ejemplo de arriba los códigos de subcampo son: **$a** para la extensión, **$b** para los otros detalles físicos, y **$c** para las dimensiones.

La denominación DESIGNADORES DE CONTENIDO se usa para referirse en conjunto a las etiquetas, los indicadores y los códigos de subcampo.

Los tres tipos de designadores de contenido (etiquetas, indicadores, y códigos de subcampo) son la clave del sistema de notación MARC 21. En su libro *MARC for Library Use* (2nd ed. (Boston: G.K. Hall & Co., 1989), p. 5), Walt Crawford llama al sistema MARC un sistema de "notación taquigráfica." Los tres tipos de designadores de contenido son los símbolos taquigráficos que marcan y explican el contenido de un registro bibliográfico.

3.2.2.- Dublín Core

3.2.2.1.- ¿Qué es el Dublin Core?

El estándar [ISO15836:2003] de metadatos Dublin Core es un simple pero eficaz conjunto de elementos para describir una amplia gama de recursos de red. La norma del Dublin Core conlleva dos niveles: Simple y Cualificado. El Dublin Core Simple conlleva quince elementos; el Dublin Core Cualificado conlleva un elemento adicional, la audiencia *[Audience]*, así como un grupo de elementos de matización (denominados por ello, cualificadores) que refinan la semántica de los elementos de tal forma que pueden ser útiles para la recuperación/localización de recursos en Internet *[resource discovery]*. La semántica del Dublin core se ha establecido por un grupo internacional e interdisciplinar de profesionales de la biblioteconomía, la Informática, la codificación textual, la comunidad museística, y otros campos teórico-prácticos relacionados.

Otra forma de ver el Dublin Core es como un "pequeño lenguaje para realizar una clase particular de declaraciones *[statements]* sobre recursos". En este lenguaje, hay dos clases de términos --elementos (nombres) y cualificadores (adjetivos)-- que pueden ordenarse en un patrón simple de una sentencia o declaración *[statement]*. Los sujetos, implícitos o sobreentendidos, de este lenguaje, son los propios recursos.

En la diversidad del mundo de Internet, el Dublin Core puede parecer una "lengua franca de metadatos para turistas digitales": captada fácilmente, pero no necesariamente al punto de expresar relaciones complejas o conceptos.

El conjunto de elementos básicos del Dublin Core se perfila en [6]. Cada elemento es opcional y puede repetirse. La mayor parte de los elementos tienen asimismo un conjunto limitado de cualificadores o refinamientos, atributos que pueden usarse para matizar más (no para extender) el significado de un elemento. La Iniciativa de Metadatos Dublin Core (DCMI) ha establecido formas normalizadas para matizar los elementos y promover el uso de esquemas *[schemes]* de codificación y vocabulario. El conjunto completo de elementos y matizaciones de elementos [7] de acuerdo a las "buenas prácticas" de la DCMI está disponible, con un registro formal en proceso.

El Dublin Core conlleva otros tres principios que se mencionan aquí, como críticos para entender cómo se debe pensar sobre las relaciones de los metadatos con los recursos subyacentes que describen

1. **El Principio uno a uno *[One-to-One Principle]*.** En general, los metadatos Dublin Core describen una manifestación o una versión de un recurso, más que asumir que las manifestaciones sustituyen una a la otra. Por ejemplo, una imagen jpeg de la Mona Lisa tiene mucho en común con la pintura original, pero no es lo mismo que la pintura. De esta forma la imagen digital debería describirse tal y como es, probablemente con el creador de la imagen digital como Creador o Contribuidor, más que el pintor de la Mona Lisa original.

   La relación entre los metadatos para el original y para la reproducción es parte de la descripción de metadatos, y ayuda al usuario a determinar si necesita ir al Louvre para ver el original, o su necesidad puede satisfacerse con una reproducción.

2. **El principio de simplificación *[Dumb-down Principle]*.** La cualificación de las propiedades del Dublin Core se rige por una regla, conocida coloquialmente como el Principio *Dumb-Down* [expresión del lenguaje coloquial para determinar que algo complejo se simplifica para hacerlo más fácil de entender]. De acuerdo a esta regla, un cliente debería poder ignorar cualquier cualificador y utilizar el valor como si estuviera sin cualificar *[unqualified]*. Aunque esto puede conllevar algunas veces la pérdida de especificidad, el valor del elemento que permanece (sin el cualificador) puede seguir siendo correcto y útil para la localización/recuperación. La cualificación se considera, por tanto, sólo para matizar, no para extender el alcance semántico de una propiedad.

3. **Valores apropiados.** La mejor práctica *[best practice]* para un elemento particular o cualificador, puede variar por el contexto, pero normalmente un implementador no puede siempre predecir que el que va a interpretar los metadatos será siempre una máquina. Esto puede imponer ciertas restricciones en la forma de construir los metadatos, pero la necesidad de utilidad para la localización/recuperación *[discovery]* debería tenerse en cuenta.

A pesar de que el Dublin Core se desarrolló originalmente con la mirada puesta en la descripción de documentos entendidos como objetos de información *[document-like objects]* (porque tradicionalmente los recursos textuales se entendían bastante bien), los metadatos DC pueden aplicarse también a otros recursos. Su aptitud para utilizarse con recursos no documentales específicos dependerá en cierta medida, de cómo sus metadatos se parecen a los metadatos de un documento convencional y también de qué propósito tienen esos metadatos.

3.2.2.2.- El Dublin Core tiene como objetivos:

Simplicidad de creación y mantenimiento

El conjunto de elementos Dublin Core se ha mantenido tan reducido y simple como ha sido posible para permitir a los no especialistas, crear registros descriptivos simples para recursos de información, de una manera fácil y barata, asegurando la recuperación eficaz de los recursos en entorno de red.

Semántica normalmente entendida

Las diferencias en la terminología y en las prácticas descriptivas entre un campo del conocimiento y otro impiden la localización/recuperación *[discovery]* de información a través de la amplitud del espacio de Internet. El Dublin Core puede ayudar al "turista digital" --alguien no especializado que busca información-- a encontrar su camino a través de un conjunto de elementos común, cuya semántica es universalmente entendida y soportada. Por ejemplo, los científicos preocupados por localizar artículos por un autor particular, y alumnos de arte interesados en trabajos de un artista particular, pueden estar de acuerdo en la importancia del elemento "creator". Tal convergencia en un conjunto de elementos común, a pesar de que sea ligeramente más genérica, aumenta la visibilidad y la accesibilidad de todos los recursos, tanto dentro de una disciplina determinada, como más allá de ésta.

Alcance internacional

El Conjunto de Elementos Dublin Core se desarrolló originalmente en inglés, pero se han creado versiones en otras muchas lenguas [8], por ejemplo finlandés, noruego, tailandés, japonés, francés, portugués, alemán, griego, indonesio y español. . El Grupo de Interés especial en localización e internacionalización está coordinando esfuerzos para aunar estas versiones en un registro distribuido.

Aunque los retos técnicos de internacionalización de la World Wide Web no se dirigen directamente por la comunidad de desarrollo del Dublin Core, la participación de representantes de prácticamente todos los continentes, ha asegurado que el desarrollo del estándar considere la naturaleza, multilingüe y multicultural, del universo de información electrónica.

Extensibilidad

Equilibrando la necesidad de simplicidad en la descripción de recursos digitales con la necesidad de precisión en la recuperación, los desarrolladores del Dublin Core reconocen la importancia de proporcionar un mecanismo para ampliar el conjunto de elementos DC para otras necesidades de recuperación de recursos. Se espera que otras comunidades de expertos de metadatos creen y administren conjuntos de metadatos adicionales, especializados acorde a las necesidades de sus respectivas comunidades. Los elementos de metadatos de dichos conjuntos de metadatos adicionales podrían usarse junto a los metadatos Dublin Core para encontrar la necesidad de interoperabilidad. El Comité de uso de la DCMI está trabajando actualmente en un modelo para llevar a cabo esto en el contexto de los "perfiles de aplicación" *[application profiles].*

Rachel Heery y Manjula Patel, en su artículo "Application profiles: mixing and matching metadata schemas" [9] (Perfiles de aplicación: mezclando y concordando esquemas de metadatos) definen un perfil de aplicación como:

" ... esquemas *[schemas]* que consisten en elementos de datos tomados de uno o más espacios de nombre *[namespaces]*, combinados juntos por los implementadores, y optimizados para una aplicación local particular".

Este modelo permite que diferentes comunidades utilicen elementos DC para la información descriptiva principalmente, permitiendo además extensiones específicas del dominio que tienen sentido dentro de un área más limitada.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
>

  <rdf:Description rdf:about="http://media.example.com/audio/guide.ra">
    <dc:creator>Rose Bush</dc:creator>
    <dc:title>A Guide to Growing Roses</dc:title>
    <dc:description>Describes process for planting and nurturing different kinds of rose bushes.</dc:description>
    <dc:date>2001-01-20</dc:date>
  </rdf:Description>
</rdf:RDF>
```

Ilustración 3.1 Ejemplo XML usando Dublin Core

Este ejemplo utiliza sólo Dublin Core para describir una grabación sonora de una guía para el crecimiento de rosales. Con XML o RDF/XML, el Dublin Core puede mezclarse con otros vocabularios de metadatos. Por ejemplo, la descripción Dublin Core sencilla anterior puede utilizarse junto a otros vocabularios, tales como vCard que describe la filiación del autor y la información de contacto, o un vocabulario más especializado para la "descripción de rosas" que describa los rosales con mayor detalle.

3.3.- Manejo de los Datos

3.3.1.- XML

XML, sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en **bases de datos**, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. [10]

3.3.1.1.- Historia

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (General Markup Language), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (Standard General Markup Language), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información.

En el año 1989 Tim Berners Lee creó la web, y junto con ella el lenguaje HTML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar. Los navegadores web sin embargo siempre han puesto pocas exigencias al código HTML que interpretan y así las páginas web son caóticas y no cumplen con la sintaxis. Estas páginas web dependen fuertemente de una forma específica de lidiar con los errores y las ambigüedades, lo que hace a las páginas más frágiles y a los navegadores más complejos.

Otra limitación de SGML es que cada documento pertenece a un vocabulario fijo, establecido por el DTD. No se pueden combinar elementos de diferentes vocabularios. Asimismo es imposible para un intérprete (por ejemplo un navegador) analizar el documento sin tener conocimiento de su gramática (del DTD). Por ejemplo el navegador sabe que al comienzo de una etiqueta debe cerrar un <p> previamente abierto. Los navegadores resolvieron esto incluyendo lógica ad hoc para el HTML, en vez de incluir un analizador genérico. Ambas opciones de todos modos son muy complejas para los navegadores.

Se buscó entonces definir un subconjunto del SGML que permita:

- Mezclar elementos de diferentes lenguajes. Es decir que los lenguajes sean extensibles.
- La creación de analizadores simples, sin ninguna lógica especial para cada lenguaje.
- Empezar de cero y hacer hincapié en que no se acepte nunca un documento con errores de sintaxis.

Para hacer esto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que necesiten interpretar el documento.

3.3.1.2.- Ventajas del XML

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

3.3.1.3.- Estructura de un documento XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de pedazos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez con notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de este como un elemento, un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo para entender la estructura de un documento XML:

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
    <remitente>
        <nombre>Alfredo Reino</nombre>
        <mail>alf@ibium.com</mail>
    </remitente>
    <destinatario>
        <nombre>Bill Clinton</nombre>
        <mail>president@WhiteHouse.gov</mail>
    </destinatario>
    <asunto>Hola Bill</asunto>
    <texto>
        <parrafo>
                ¿Hola que tal? Hace
                <enfasis> mucho </enfasis>
                que no escribes. A ver si llamas y quedamos para tomar algo.
        </parrafo>
    </texto>
</mensaje>
```

Ilustración 3.2 Documento XML

3.3.1.4.- Documentos XML bien formados

Se llama documentos "bien formados" (del inglés well formed) a los documentos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, ser analizados correctamente por cualquier "parser" (Analizador Sintáctico) que cumpla con la norma. Se separa esto del concepto de validez que se explica más adelante.

Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo puede tener un elemento inicial.

Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos entendibles por las personas.

3.3.1.5.- Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento.

Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo contiene:

- una declaración XML. Es la sentencia que declara al documento como un documento XML.
- una declaración de tipo de documento. Enlaza el documento con su DTD, o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- uno o más comentarios e instrucciones de procesamiento.

Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener al menos un elemento raíz, característica indispensable también para que el documento esté bien formado.

Elementos: Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos: Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

Entidades predefinidas: Entidades para representar caracteres especiales para que no sean interpretados como marcado en el procesador XML.

Secciones CDATA: Es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML

Comentarios: Comentarios a modo informativo para el programador que han de ser ignorados por el procesador. Los comentarios en XML tienen el siguiente formato:

```
<!--- Esto es un comentario --->
<!-- Otro comentario -->
```

Ilustración 3.3 Un Comentario XML

3.3.1.6.- Validez

Que un documento sea "bien formado" solamente habla de su estructura sintáctica básica, es decir que se componga de elementos, atributos y comentarios como XML manda que se escriban. Ahora bien, cada aplicación de XML, es decir cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD (Document Type Definition = Definición de Tipo de Documento) o como XSchema). Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.

Document type definition (DTD)

La DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD se denominan válidos.

XML Schemas (XSD)

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así, una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001.

Un Schema es algo similar a un DTD, define que elementos puede contener un documento XML, como están organizados y que atributos y de qué tipo pueden tener sus elementos.

XML Schema es un lenguaje escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que la DTD, más limitadas en la descripción de los documentos a nivel formal.

Los documentos esquema (usualmente con extensión .xsd de XML Schema Definition (XSD)) se concibieron como una alternativa a las DTD, más compleja, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML. El principal aporte de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

Ventajas de los Schemas frente a los DTDs

- Usan sintaxis de XML, al contrario que los DTDs.
- Permiten especificar los tipos de datos.
- Son extensibles.
- Permiten crear expresiones regulares de validación de datos.

3.3.2.- XPath

XML Path Language, es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos (plain text). XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. XPath fue creado para su uso en el estándar XSLT, en el que se usa para seleccionar y examinar la estructura del documento de entrada de la transformación.

3.3.2.1.- ¿Qué es y para qué sirve?

Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma diferenciada.

El tratamiento del fichero XML comienza por la localización del mismo a lo largo del conjunto de documentos existentes en el mundo. Para llevar a cabo esta localización de forma unívoca, se utilizan los URI (Unifom Resource Identifiers), de los cuales los URL (Unifom Resource Locators) son sin duda los más conocidos.

Una vez localizado el documento XML, la forma de seleccionar información dentro de él es mediante el uso de XPath, que es la abreviación de lo que se conoce como XML Path Language. Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.

XPath en sí es un lenguaje sofisticado y complejo, pero distinto de los lenguajes procedurales que solemos usar (C, C++, Basic, Java...). Además, como casi todo en el mundo de XML, aún está en estado de desarrollo, por lo que no es fácil encontrar herramientas que incorporen todas sus funcionalidades.

XPath es a su vez la base sobre la que se han especificado nuevas herramientas que lo aprovechan para el tratamiento de documentos XML. Herramientas tales como XPointer, XLink y XQL (el lenguaje que maneja los documentos XML como si de una base de datos se tratase), que también están en estado de desarrollo, pero que sin duda cambiarán el modo en que actualmente concebimos la navegación por la Web. Así, XPath sirve para decir cómo debe procesar una hoja de estilo el contenido de una página XML, pero también para poder poner enlaces o cargar en un navegador zonas determinadas de una página XML, en vez de toda la página.

3.3.2.2.- El modelo de datos de XPath

Un documento XML es procesado por un analizador (o parser) construyendo un árbol de nodos. Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen solo texto, comentarios, instrucciones de proceso o incluso que están vacíos y solo tienen atributos

La forma en que XPath selecciona partes del documento XML se basa precisamente en la representación arbórea que se genera del documento. De hecho, los "operadores" de que consta este lenguaje nos recordarán la terminología que se utiliza a la hora de hablar de árboles en informática: raíz, hijo, ancestro, descendiente, etc.

Un caso especial de nodo son los nodos atributo. Un nodo puede tener tantos atributos como desee, y para cada uno se le creará un nodo atributo. No obstante, dichos nodos atributo NO se consideran como hijos suyos, sino más bien como etiquetas añadidas al nodo elemento.

3.3.2.3.- Tipos de Nodos

Existen distintos tipos de nodos en un árbol generado a partir de un documento XML, a saber: raíz, elemento, atributo, texto, comentario e instrucción de procesamiento (respectivamente; root, elements, attribute, text, comment y processing instruction).

3.3.2.4.- Nodo Raíz

Se identifica por /. No se debe confundir el nodo raíz con el elemento raíz del documento. Así, si el documento XML de nuestro ejemplo tiene por elemento raíz a libro, éste será el primer nodo que cuelgue del nodo raíz del árbol, el cual es: /.

Insisto: / hace referencia al nodo raíz del árbol, pero no al elemento raíz del documento XML, por más que un documento XML solo pueda tener un elemento raíz. De hecho, podemos afirmar que el nodo raíz del árbol contiene al elemento raíz del documento.

3.3.2.5.- Nodo Elemento

Cualquier elemento de un documento XML se convierte en un nodo elemento dentro del árbol. Cada elemento tiene su nodo padre. El nodo padre de cualquier elemento es, a su vez, un elemento, excepto el elemento raíz, cuyo padre es el nodo raíz. Los nodos elemento tienen a su vez hijos, que son: nodos elemento, nodos texto, nodos comentario y nodos de instrucciones de proceso. Los nodos elemento también tienen propiedades tales como su nombre, sus atributos e información sobre los "espacios de nombre" que tiene activos.

Una propiedad interesante de los nodos elemento es que pueden tener identificadores únicos (para ello deben ir acompañados de un DTD que especifique que dicho atributo toma valores únicos), esto permite referenciar a dichos elementos de una forma mucho más directa.

3.3.2.6.- Nodos texto

Por texto vamos a hacer referencia a todos los caracteres del documento que no está marcados con alguna etiqueta. Un nodo texto no tiene hijos, es decir, los distintos caracteres que lo forman no se consideran hijos suyos.

3.3.2.7.- Nodos atributo

Como ya hemos indicado, los nodo atributo no son tanto hijos del nodo elemento que los contiene como etiquetas añadidas a dicho nodo elemento. Cada nodo atributo consta de un nombre, un valor (que es siempre una cadena) y un posible "espacio de nombres".

Aquellos atributos que tienen por valor el valor por defecto asignado en el DTD se tratarán como si el valor se le hubiese al escribir el documento XML. Al contrario, no se crea nodo para atributos no especificados en el documento XML, y con la propiedad #IMPLIED definida en su DTD. Tampoco se crean nodos atributo para las definiciones de los espacios de nombre. Todo esto es normal si tenemos en cuenta que no es necesario tener un DTD para procesar un documento XML.

3.3.2.8.- Nodos comentario y de instrucciones de proceso

Aparte de los nodos indicados, en el árbol también se generan nodos para cada nodo con comentarios y con instrucciones de proceso. Al contenido de estos nodos se puede acceder con la propiedad string-value.

3.4.- Formatos Especiales

Como dice en el segundo objetivo específico del punto 2.- de este informe, existen formatos estándares de archivos, de los cuales vale la pena estudiar algunos más a fondo para extraer cierta información de estos de tal manera que sea útil para el usuario.

El sistema será capaz de extraer cierta información de estos archivos “conocidos”, mediante una implementación de plugins para cada formato, se podrá ir agregando cada vez más formatos al sistema.

Esta interfaz de plugins tendrá dos funciones principales, primero registrar que campos adicionales maneja, para el caso de las configuraciones el sistema, y segundo una función encargada de devolver los campos leídos de un archivo pasando su dirección como parámetro.

A continuación se presentan clasificados por su función los formatos que inicialmente manejará el sistema:

3.4.1.- Compresión

3.4.1.1.- ZIP

Es un formato de fichero bastante simple, que comprime cada uno de los archivos de forma separada. Comprimir cada archivo independientemente del resto de archivos comprimidos permite recuperar cada uno de los ficheros sin tener que leer el resto, lo que aumenta el rendimiento. El problema, es que el resultado de agrupar un número grande de pequeños archivos es siempre mayor que agrupar todos los archivos y comprimirlos como si fuera uno sólo. Éste último comportamiento es el del, también conocido, algoritmo de compresión RAR.

La especificación de ZIP indica que cada archivo puede ser almacenado, o bien sin comprimir, o utilizando una amplia variedad de algoritmos de compresión. Sin embargo, en la práctica, ZIP se suele utilizar casi siempre con el algoritmo de Phil Katz.

ZIP soporta un sistema de cifrado simétrico basado en una clave única. Sin embargo, este sistema de cifrado es débil ante ataques como el ataque de texto plano, ataque del diccionario y el ataque de fuerza bruta. Además, también está soportado el distribuir las partes de un archivo compreso en distintos medios, generalmente disquetes.

Con el tiempo, se han ido incluyendo nuevas características, como nuevos métodos de cifrado. Sin embargo, estas nuevas características no están soportadas por las aplicaciones más utilizadas.

En el proyecto, para leer e interpretar este tipo de archivos se usará la librería “ZLib for .NET” [11] que es una portación de Zlib [12] escrita por Jean-loup Gailly y Mark Adler.

En el Anexo C se encuentra la información técnica del formato.

3.4.1.2.- RAR

Este formato fue desarrollado por Eugene Roshal y lleva su nombre. RAR significa **R**oshal **AR**chive. La primera versión comercial de RAR se lanzó a finales de 1993. Esta primera versión demostró ser más eficaz que la proporcionada por ZIP y contaba con un interfaz de usuario a pantalla completa, por lo que rápidamente se convirtió en el primer competidor de ZIP.

El RAR es más lento que el ZIP pero comprime más y tiene un mayor sistema de redundancia de datos para prevenir errores.

RAR utiliza un algoritmo de compresión basado en el LZSS, que, a su vez, se basaba en el LZ77, de James Storer y Thomas Szymanski (1982). La ventana de búsqueda puede variar entre 64k y 1 Mb.

Rar permite lo que se conoce como compresión sólida que permite comprimir varios ficheros juntos, de forma que una misma ventana de búsqueda se aplica a todo, con lo que el nivel de compresión es mayor.

En el proyecto, para leer e interpretar este tipo de archivos se usará la librería “UnRAR.dll” de RarLab [13] escrita por Alexander L. Roshal.

En el Anexo C se encuentra la información técnica del formato.

3.4.2.- Audio

3.4.2.1.- MP3

MPEG-1 Audio Layer 3, más conocido como MP3, conocido también por su grafía emepetrés, es un formato de audio digital comprimido con pérdida desarrollado por el Moving Picture Experts Group (MPEG) para formar parte de la versión 1 (y posteriormente ampliado en la versión 2) del formato de vídeo MPEG. Su nombre es el acrónimo de MPEG-1 Audio Layer 3.

Este formato fue desarrollado principalmente por Karlheinz Brandenburg, director de tecnologías de medios electrónicos del Instituto Fraunhofer IIS, perteneciente al Fraunhofer-Gesellschaft - red de centros de investigación alemanes - que junto con Thomson Multimedia controla el grueso de las patentes relacionadas con el MP3. La primera de ellas fue registrada en 1986 y varias más en 1991. Pero no fue hasta julio de 1995 cuando Brandenburg usó por primera vez la extensión .mp3 para los archivos relacionados con el MP3 que guardaba en su ordenador. Un año después su instituto ingresaba en concepto de patentes 1,2 millones de euros. Diez años más tarde esta cantidad ha alcanzado los 26,1 millones.

Un fichero Mp3 se constituye de diferentes frames MP3 que a su vez se componen de una cabecera Mp3 y los datos MP3. Esta secuencia de datos es la denominada stream elemental. Cada uno de los Frames son independientes, es decir, una persona puede cortar los frames de un fichero MP3 y después reproducirlos en cualquier reproductor MP3 del Mercado. El grafico muestra que la cabecera consta de una palabra de sincronismo que es utilizada para indicar el principio de un frame valido. A continuación siguen una serie de bits que indican que el fichero analizado es un fichero Standard MPEG y si usa o no la capa 3. Después de todo esto los valores difiieren dependiendo del tipo de archivo MP3. Los rangos de valores quedan definidos en la ISO/IEC 11172-3.

En este tipo de archivos generalmente la información se encuentra encapsulada en formato tag ID3, por lo que es esta la especificación a estudiar, en el ANEXO C se encuentra la información técnica.

3.4.2.2.- WMA

Windows Media Audio o **WMA** es un formato de compresión de audio con pérdida propiedad de Microsoft, aunque recientemente se ha desarrollado de compresión sin pérdida.

Compite con el MP3, antiguo y bastante inferior técnicamente; y Ogg-Vorbis, superior y libre, usando como estrategia comercial la inclusión de soporte en el reproductor Windows Media Player, incluido en su popular sistema operativo Windows.

Aunque el soporte de este formato se ha ampliado desde Windows Media Player y ahora se encuentra disponible en varias aplicaciones y reproductores portátiles, el MP3 continua siendo el formato más popular y por ello más extendido.

A diferencia del MP3, éste formato posee una infrastructura para proteger el Copyright y así hacer más difícil el **"**tráfico ilegal**"** de música.

Este formato está especialmente relacionado con Windows Media Video (WMV) y Advanced Streaming Format (ASF).

3.4.3.- Contenedores de Video

3.4.3.1.- AVI

AVI es el acrónimo de Audio Video Interleave (intercalado de audio y video). Se trata de un formato de archivo que actúa como contenedor de flujos de datos de audio y vídeo.

El formato AVI permite almacenar simultáneamente un flujo de datos de video y varios flujos de audio. El formato concreto de estos flujos no es objeto del formato AVI y es interpretado por un programa externo denominado códec. Es decir, el audio y el video contenidos en el AVI pueden estar en cualquier formato (mp3/divx, u ac3/xvid, por ejemplo entre otros). Por eso se le considera un formato contenedor.

Para que todos los flujos puedan ser reproducidos simultáneamente es necesario que se almacenen de manera entrelazada. De esta manera, cada fragmento de archivo tiene suficiente información como para reproducir unos pocos fotogramas junto con el sonido correspondiente.

Obsérvese que el formato AVI admite varios flujos de datos de audio, lo que en la práctica significa que puede contener varias bandas sonoras en varios idiomas. Es el reproductor multimedia quien decide cuál de estos flujos debe ser reproducido, según las preferencias del usuario.

Los archivos AVI se dividen en fragmentos bien diferenciados denominados chunks. Cada chunk tiene asociado un identificador denominado etiqueta FourCC. El primer fragmento se denomina cabecera y su papel es describir meta-información respecto al archivo, por ejemplo, las dimensiones de la imagen y la velocidad en fotogramas por segundo. El segundo chunk contiene los flujos entrelazados de audio y video. Opcionalmente, puede existir un tercer chunk que actúa a modo de índice para el resto de chunks.

3.4.4.- Imágenes

JPG, GIF, BMP, PNG

El formato JPEG es usado ampliamente para fotografías e imágenes de gran tamaño y variedad de color en la web y por las cámaras digitales. Es un formato comprimido con pérdida de calidad, aunque esta se puede ajustar.

GIF es utilizado popularmente en la web. Formato de 8 bits (256 colores máximo), con soporte de animación por frames. Utiliza la compresión LZW lo que provoca conflictos de patente con el propietario del algoritmo.

Comúnmente usado por los programas de Microsoft Windows y por el sistema operativo propiamente dicho. Se le puede aplicar compresión sin pérdidas, aunque no todos los programas son compatibles.

PNG es gráfico libre con compresión sin pérdida que ofrece profundidades desde 1 hasta 32 bits. Fue diseñado para reemplazar al GIF en la web.

3.4.5.- Documentos

PDF

PDF (del inglés Portable Document Format, Formato de Documento Portátil) es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la información necesaria para la presentación final del documento, determinando todos los detalles de cómo va a quedar, no requiriéndose procesos anteriores de ajuste ni de maquetación. Cada vez se utiliza más también como especificación de visualización, gracias a la gran calidad de las fuentes utilizadas y a las facilidades que ofrece para el manejo del documento, como búsquedas, hiperenlaces, etc.

DOC

Microsoft Word utiliza un formato nativo cerrado y muy utilizado, comúnmente llamado DOC (utiliza la extensión de archivo .doc). Por la amplísima difusión del Microsoft Word, este formato ha convertídose en estándar de facto con el que pueden transferirse textos con formato o sin formato, o hasta imágenes, siendo preferido por muchos usuarios antes que otras opciones como TXT para el texto sin formato o JPG para gráficos; sin embargo, este formato posee la desventaja de tener un mayor tamaño comparado con algunos otros. Por otro lado, la Organización Internacional para la Estandarización ha elegido el formato OpenDocument como estándar para el intercambio de texto con formato, lo cual ha supuesto una desventaja para el formato .doc.

4.- Estudios De Factibilidad

"Todos los sistemas son realizables con recursos limitados y en un tiempo infinito, desafortunadamente el desarrollo de un sistema basado en computadores se caracteriza por la escasez de recursos y la dificultad (si no imposible) de cumplir los plazos de entrega. Es necesario y prudente evaluar la viabilidad de un proyecto lo antes posible. Se puede evitar meses o años de esfuerzo, millones y una inversión profesional incalculable, si un sistema mal concebido es reconocido como tal al principio de la etapa de definición". [14]

En esta sección, se presentará el objetivo de dar a conocer el estudio de factibilidad realizado respecto al "Software para la creación, almacenamiento y búsqueda en índices virtuales de discos". Los aspectos más importantes que se han considerado son:

1. **Factibilidad Técnica:** Consiste en determinar si es posible dar solución a la problemática presentada tomando en cuenta los recursos computacionales disponibles en la organización o si es necesario adquirir otros.

2. **Factibilidad Económica:** Consiste en ver si el proyecto es viable económicamente.

3. **Factibilidad Legal:** El estudio de la factibilidad legal, tiene como objetivo verificar para cada escenario, si el sistema a desarrollar no vulnera las leyes vigentes o la reglamentación propia de la organización. Es decir, pretende observar si no se incurre en infracciones, violaciones u otros que podrían provocar la imposibilidad de poner en ejecución el sistema, o su interrupción en algún momento de su operación rutinaria.

4. **Factibilidad operacional:** Consiste en determinar las posibilidades de llevar a la práctica el desarrollo y uso del sistema, tomando en cuenta el ambiente organizacional que existe en el lugar en que se realiza y las capacidades de los usuarios potenciales.

4.1.- Factibilidad Técnica

Dada la elección de las herramientas y plataformas a usar (punto 5.3) la tecnología requerida desde el punto de vista de los usuarios para conseguir la funcionalidad y rendimiento mínimo del sistema a construir es la siguiente:

- **Software:**
  - ✓ S.O. Microsoft® Windows® 98/me/2000/XP Home/XP PRO
  - ✓ Microsoft® Internet Explorer 5.01 o una versión posterior.
  - ✓ Microsoft® .NET Framework versión 2.0 Redistributable Package.

- **Hardware:**
  - ✓ Procesador Pentium III o compatible superior (500 MHz mínimo)
  - ✓ Memoria 128MB RAM
  - ✓ Disco Duro 10MB de espacio (Aplicación), 5MB aproximadamente base de 100 discos.
  - ✓ Unidad Lectora de Discos.

La tecnología descrita anteriormente, se encuentra en la Pontificia Universidad Católica de Valparaíso, ya que cuenta con los equipos necesarios para la instalación del sistema.

Estos son los requisitos para desarrollar e implementar el sistema.

- **Software:**
  - ✓ Microsoft® Windows® XP Professional SP2
  - ✓ Microsoft® .NET Framework versión 2.0
  - ✓ Microsoft® Visual Basic Express 2005

- **Hardware:**

Requisitos mínimos:
- ✓ Procesador: 600 MHz
- ✓ Memoria RAM: 192 MB
- ✓ Disco Duro: 500MB

Requisitos recomendados:
- ✓ Procesador: 1 GHz
- ✓ Memoria RAM: 512 MB
- ✓ Disco Duro: 1.3GB

Los requisitos y herramientas descritas anteriormente para la implementación de la solución, están ya instaladas y en funcionamiento en los equipos del desarrollador del proyecto.

4.2.- Factibilidad Económica

Desde la perspectiva económica para realizar el proyecto, se tomaron en cuenta los siguientes puntos:

- **Costos previos:** Se cuenta con los equipos necesarios para desarrollar el proyecto, y se cuenta con las licencias de las herramientas necesarias para el desarrollo del software, por lo cual la Escuela de Ingeniería de la Pontificia Universidad Católica de Valparaíso, no deberá incurrir en gastos para adquirir la herramienta MS Visual Studio Express 2005, esta está disponible de manera gratuita para todo aquel que quiera usar y experimentar con las nuevas tecnologías de Microsoft.

En lo que refiere a los costos por horas / hombre, que corresponde a los desarrolladores, por ser un proyecto desarrollado dentro de las exigencias de la Escuela de Ingeniería Informática de la PUCV para otorgar el título de Ingeniero Ejecución Informática, no es considerado, pero se hará de todas formas la estimación de cuanto sería el costo del desarrollo de la aplicación en cuanto a horas / hombre.

- **Costos Hora - Hombre:** Costos asociados por el esfuerzo realizado en el proyecto de parte de los desarrolladores, los cuales se detallan de la siguiente manera:
  - ✓ Sueldo aproximado a un alumno de último año de Ingeniería Ejecución Informática: **500.000 (Líquidos)**
  - ✓ Días hábiles trabajados en un mes regular: **20**
  - ✓ Horas laborales diarias: **8**

Por lo tanto el cálculo aproximado al costo de horas-hombres es el siguiente:

500.000 / 20 = $ 25.000 por día

25.000 / 8 = $ 3.125 X horas / hombre

Horas trabajadas aproximadamente en este proyecto = 720 Horas

$ 3.125 horas / hombre x 720 horas = $ 2.250.000

Costo aproximado del proyecto = $ 2.250.000

- **Costos Relativos al proyecto:** Para la formación del usuario, se entregara un manual de usuario además tendrá una ayuda vía Internet, por lo tanto no hay costos relacionados.

- **Costos Continuos en el tiempo:** No hay costos relacionados con este ítem.

- **Beneficios por ahorro de tiempo:** El ahorro por este concepto depende de cuánto vale el tiempo de la persona o empresa que utiliza el sistema.

- **Beneficios por ahorro de internet y discos:** El ahorro por estos conceptos es realmente marginal debido a los bajos costos de los discos y el acceso a internet.

- **Beneficios por venta de licencias:** La venta de licencias del software desarrollado generará beneficios económicos los cuales se detallan a continuación.

  Tomando en cuenta productos comerciales de similares características al de éste proyecto se calculará un precio promedio y uno base de entrada al mercado para este software.

Producto	Precio por unidad
Advanced Disk Catalog (elcomsoft)	$9.120 CLP (20,00 USD)
Catalogador 2008 (yursoft)	$14.300 CLP (€19,99)
Precio Promedio	**$11.710**

El precio de entrada al mercado será $9.900, por lo tanto para cubrir los costos se debe realizar el siguiente cálculo:

$ 2.250.000 / $ 9.900 = 227,27 ≈ 228 Unidades.

Esto significa que se deben vender al menos 228 licencias para cubrir los costos iníciales del desarrollo del proyecto.

4.3.- Factibilidad Legal

Con relación a la factibilidad legal, no existen problemas, ya que el hardware utilizado en la Pontificia Universidad Católica de Valparaíso y por el desarrollador fue adquirido legalmente. Con respecto al software para desarrollar, no se tendrá que adquirir licencias, ya que el desarrollador ya las poseen, además, la versión de la herramienta MS Visual Studio Express 2005, luego de registrarla gratuitamente es totalmente funcional, por lo tanto su utilización es legal para fines de uso Académico.

Con respecto al sistema en sí, se hizo un estudio del cual se concluye que es legal dentro de los marcos de la ley de propiedad intelectual chilena, a continuación se presenta un extracto de esta donde se ve los artículos en que se fundamenta esta conclusión.

Identificación de la Norma : LEY-17336 PROPIEDAD INTELECTUAL
Fecha de Publicación : 02.10.1970
Fecha de Promulgación : 28.08.1970
Organismo : MINISTERIO DE EDUCACION PUBLICA
Ultima Modificación : LEY-19928 31.01.2004

Título I - DERECHO DE AUTOR
Capítulo I
Naturaleza y objeto de la Protección. Definiciones
…
Art. 3° Quedan especialmente protegidos con arreglo a la presente ley:
…
17) Las compilaciones de datos o de otros materiales, en forma legible por máquina o en otra forma, que por razones de la selección o disposición de sus contenidos, constituyan creaciones de carácter intelectual. Esta protección no abarca los datos o materiales en sí mismos, y se entiende sin perjuicio de cualquier derecho de autor que subsista respecto de los datos o materiales contenidos en la compilación;
…

Párrafo IV
Excepciones al derecho de autor

Artículo 47. Para los efectos de la presente ley no se considera comunicación ni ejecución pública de la obra, inclusive tratándose de fonogramas, su utilización dentro del núcleo familiar, en establecimientos educacionales, de beneficencia u otras instituciones similares, siempre que esta utilización se efectúe sin ánimo de lucro. En estos casos no se requiere remunerar al autor, ni obtener su autorización.

Asimismo, para los efectos de la presente ley, la adaptación o copia de un programa computacional efectuada por su tenedor o autorizada por su legítimo dueño, no constituye infracción a sus normas, siempre que la adaptación sea esencial para su uso en un computador determinado y no se le destine a un uso diverso, y la copia sea esencial para su uso en un computador determinado o para fines de archivo o respaldo. (45)

Las adaptaciones obtenidas en la forma señalada no podrán ser transferidas bajo ningún título, sin que medie autorización previa del titular del derecho de autor respectivo; igualmente, las copias obtenidas en la forma indicada no podrán ser transferidas bajo ningún título, salvo que lo sean conjuntamente con el programa computacional que les sirvió de matriz. (46)
…

4.4.- Factibilidad Operacional

Esta factibilidad comprende una determinación de la probabilidad de que un nuevo sistema se use como se supone.

Deberían considerarse cuatro aspectos de la factibilidad operacional por lo menos.

- Primero, un nuevo sistema puede ser demasiado complejo para los usuarios de la organización o los operadores del sistema. Si lo es, los usuarios pueden ignorar el sistema o bien usarlo en tal forma que cause errores o fallas en el sistema. En este caso, no lo es, ya que una de las características es que sea intuitivo y fácil de usar.

- Segundo, un sistema puede hacer que los usuarios se resistan a él como consecuencia de una técnica de trabajo, miedo a ser desplazados, intereses en el sistema antiguo u otras razones. Para cada alternativa debe explorarse con cuidado la posibilidad de resistirse al cambio al nuevo sistema. Al ser una solución particular este punto no se aplica ya que el usuario usa el sistema porque él lo adquirió no porque se lo impusieron.

- Tercero, un nuevo sistema puede introducir cambios demasiado rápido para permitir al personal adaptarse a él y aceptarlo. Un cambio repentino que se ha anunciado, explicado y “vendido” a los usuarios con anterioridad puede crear resistencia. El sistema actual es muy parecido a tecnologías y entornos ya existentes por lo tanto la adaptación se da de manera muy natural, esta se a comprobado en algunas pruebas de usabilidad realizadas a este, los usuarios reaccionan muy bien ante la interfaz del sistema.

- Una última consideración es la probabilidad de la obsolescencia subsecuente en el sistema. La tecnología que ha sido anunciada pero que aún no está disponible puede ser preferible a la tecnología que se encuentra en una o más de las alternativas que se están comparando, o cambios anticipados en las practicas o políticas administrativas pueden hacerse que un nuevo sistema sea obsoleto muy pronto. En cualquier caso, la implantación de la alternativa en consideración se convierte en impráctica. En este caso el sistema se desarrolla con tecnología totalmente nueva y vigente, pero a su vez madura y bien estudiada como lo es “.NET 2.0”, a su vez dada la filosofía de esta plataforma se puede portar y actualizar fácilmente el sistema a las versiones futuras de esta.

Un resultado frecuente de hallazgos negativos acerca de la factibilidad operacional de un sistema es que éste no se elimina sino que se simplifica para mejorar su uso. Otras posibilidades son que los programas de relaciones públicas o de entrenamiento estén diseñados para enfocarse a sobreponerse a la resistencia a un nuevo sistema, o se desarrollan formas para hacer fases en el nuevo sistema en un largo periodo para que el cambio total, que traumatizaría a los usuarios u operadores, se convierta en una serie de pequeños cambios.

5.- Paradigma, Metodología y Herramientas a utilizar

5.1.- Paradigma

5.1.1.- Elección de un Paradigma

Luego de analizar las ventajas y desventajas de los paradigmas más comunes, se opta por usar el modelo del Proceso Unificado de Desarrollo, por su modularización del problema en sí, y por su desarrollo incremental e iterativo.

Existe una tendencia en la informática actual que no se puede ignorar, se está desarrollando orientado a objetos, usando RUP, algo que no es de extrañar, ya que RUP es el resultado de tres décadas de desarrollo y uso práctico.

5.1.2.- Rational Unified Process

El Proceso Unificado de Rational (RUP, el original inglés Rational Unified Process) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

El RUP está basado en 6 principios clave

1. **Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico.

2. **Balancear prioridades:** Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

3. **Colaboración entre equipos:** El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, etc.

4. **Demostrar valor iterativamente:** Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto.

5. **Elevar el nivel de abstracción:** Este enfoque ha ofrecido una alternativa muy productiva para construir aplicaciones

6. **Enfocarse en la calidad:** El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción

5.1.3.- Ciclo de vida

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

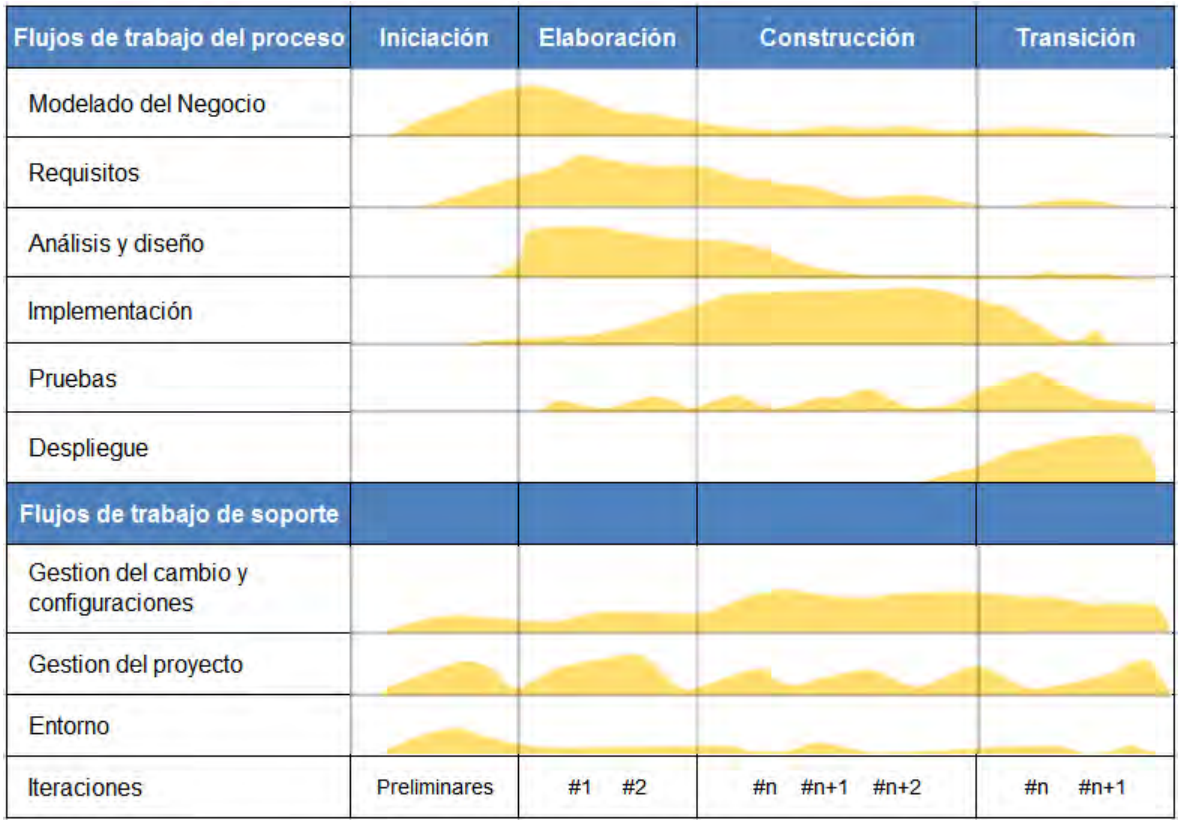


Ilustración 5.1 Un típico perfil de proyecto mostrando el tamaño relativo de las cuatro fases

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- **Inicio:** se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos
- **Elaboración:** se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos
- **Construcción:** se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario
- **Transición:** se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

5.1.4.- Principales características

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

5.1.5.- Entregables del proyecto

En este punto se describen las actividades desarrolladas basadas en RUP para este proyecto. Por las características del proyecto, se han incluido muy pocos artefactos, roles y actividades del modelo, manteniendo las más esenciales. Dicha configuración está basada en la siguiente selección de artefactos:

A continuación se describen brevemente cada uno de los artefactos que se generarán y usarán durante el proyecto:

1. **Flujos de Trabajo**

   Se utilizarán Diagramas de Actividad para modelar los Flujos de Trabajo (workflows) del área problema, tanto los actuales (previos a la implantación de nuevo sistema) como los propuestos, que serán soportados por el sistema desarrollado

2. **Características del Producto Software**

   Es una lista de las características principales del producto, deseables desde una perspectiva de las necesidades del cliente.

3. Modelo de Casos de Uso

El modelo de Casos de Uso presenta la funcionalidad del sistema y los actores que hacen uso de ella. Se representa mediante Diagramas de Casos de Uso.

4. Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: pre-condiciones, pos condiciones, flujo de eventos, requisitos no-funcionales asociados.

5. Modelo de Análisis y Diseño

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación). Está constituido esencialmente por un Diagrama de Clases y algunos Diagramas de Estados para las clases que lo requieran.

6. Modelo de Implementación

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema.

7. Modelo de Pruebas

Para cada Caso de Uso se establecen pruebas de Aceptación que validarán la correcta implementación del Caso de Uso. Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados.

8. Manual de Instalación

Este documento incluye las instrucciones para realizar la instalación del producto.

9. Material de Usuario

Corresponde a un conjunto de documentos y facilidades de uso del sistema.

10. Producto

Todos los ficheros fuente y ejecutable del producto.

5.2.- Metodología

5.2.1.- Elección de una metodología

Como metodología se escoge la Orientada a Objetos la cual define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

5.2.2.- Programación Orientada a Objetos

De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos).

Dada esta propiedad de conjunto de una clase de objetos, que al contar con una serie de atributos definitorios, requiere de unos métodos para poder tratarlos (lo que hace que ambos conceptos están íntimamente entrelazados), el programador debe pensar indistintamente en ambos términos, ya que no debe nunca separar o dar mayor importancia a los atributos en favor de los métodos, ni viceversa. Hacerlo puede llevar al programador a seguir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen esa información por otro (llegando a una programación estructurada camuflada en un lenguaje de programación orientado a objetos).

5.2.3.- Diferencias con respecto a la programación estructurada

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada se escriben funciones y después les pasan datos. Los programadores que emplean lenguajes orientados a objetos definen objetos con datos y métodos y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos.

5.3.- Herramientas

5.3.1.- Para la Documentación

Todos los informes y documentos que se generaron en el proyecto fueron creados a través del procesador de textos MS WORD 2003.

5.3.2.- Para los Diagramas

Para los diagramas se utilizó la Herramienta MS Visio 2003 y SmartDraw.

5.3.3.- Para la planificación de Proyecto

Para realizar la planificación temporal se utilizó MS Project 2003.

5.3.4.- Para el Desarrollo e Implementación del Software

Como herramienta principal de programación se uso el entorno de desarrollo integrado MS Visual Basic Express 2005.

5.3.4.1.- Características de MS Visual Basic Express

Las características del editor de Visual Basic reducen drásticamente los errores de programación en la fase de diseño, tanto para desarrolladores principiantes como avanzados. Al proporcionar una funcionalidad similar a la corrección ortográfica y gramatical de Microsoft Word, por el cual Visual Basic sugiere correcciones para los errores de sintaxis más habituales. Además el compilador advierte a los desarrolladores de la existencia de código semánticamente incorrecto que podría producir errores en tiempo de ejecución, como por ejemplo intentos de obtener acceso a elementos de código ante de la inicialización.

Visual Basic mejora en gran medida la manipulación y recuperación de datos. Está disponible un diseño de datos simplificado desde el entorno de desarrollo para los datos locales y remotos. También incorpora la capacidad de crear aplicaciones enlazadas a datos sin escribir ni una sola línea de código. Ideal para numerosos escenarios de acceso a datos habituales, esta característica permitirá a los desarrolladores generar automáticamente un IU personalizable y enlazada a datos arrastrando y colocando una tabla o varias columnas en un formulario.

Visual Basic proporciona una depuración mejorada. Al incorporar “Editar y continuar”, los desarrolladores pueden modificar y comprobar el código fuente sin necesidad de detener y reiniciar la sesión de depuración. Este ciclo iterativo de desarrollo y depuración, junto con las funciones avanzadas de corrección de errores y análisis de código en modo de interrupción, proporcionan a los desarrolladores que utilizan Visual Basic la experiencia de depuración más eficaz y flexible que hayan conocido jamás.

En resumen Visual Basic se centra en permitir el rápido desarrollo de aplicaciones que abarquen todos los niveles. Las características previstas en el depurador, los diseñadores visuales, el editor de código y el lenguaje incrementan en gran medida la productividad y permiten a los desarrolladores crear unas aplicaciones robustas y elegantes más rápidamente. [15]

Debido a estas características y su nulo costo, sumado a la poca disponibilidad de tiempo de desarrollo, se escogió esta herramienta de desarrollo.

6.- Modelamiento y Diagramas

6.1.- Modelos de Casos de Uso

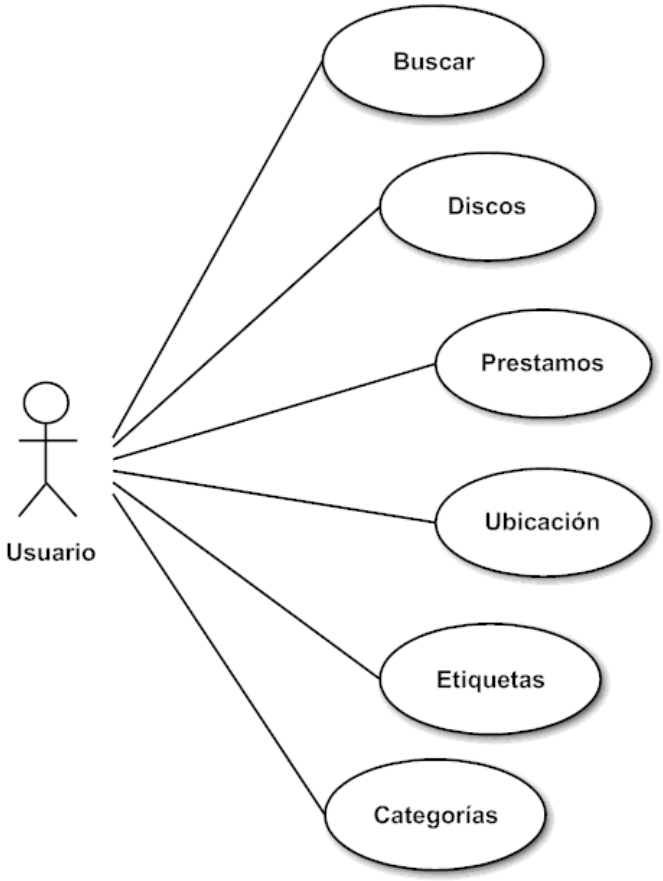


Ilustración 6.1 Caso de Uso General

Aquí se ven los casos Buscar, Discos, Etiquetas, Categorías, Prestamos y Ubicación. Todos ellos interactúan con el usuario ya que esta es una aplicación personal.

El caso Discos tiene relación con todo lo que es administrar y gestionar los discos que son almacenados en la base, luego se desglosará en más casos de uso a medida que se diseñe el Sistema, estos serian por ejemplo, crear base, abrir base, guardar base, mesclar bases, leer disco, eliminar disco, editar etiquetas, etc.

El caso Préstamos tiene relación con todo lo que es administrar y gestionar los préstamos de los discos, así también con la lista de contactos a los cuales son prestados los discos. Esto genera varios casos de uso relacionados, por ejemplo: crear, modificar, eliminar contactos, revisar préstamos, asignar préstamos, eliminar préstamos, etc.

El caso Ubicación tiene relación con todo lo que es la ubicación física de los discos en el lugar donde se almacenan, esto genera casos de uso como crear, modificar y eliminar modelo de ubicación, ubicar discos, navegar discos, generar orden, etc.

6.1.1.- Buscar

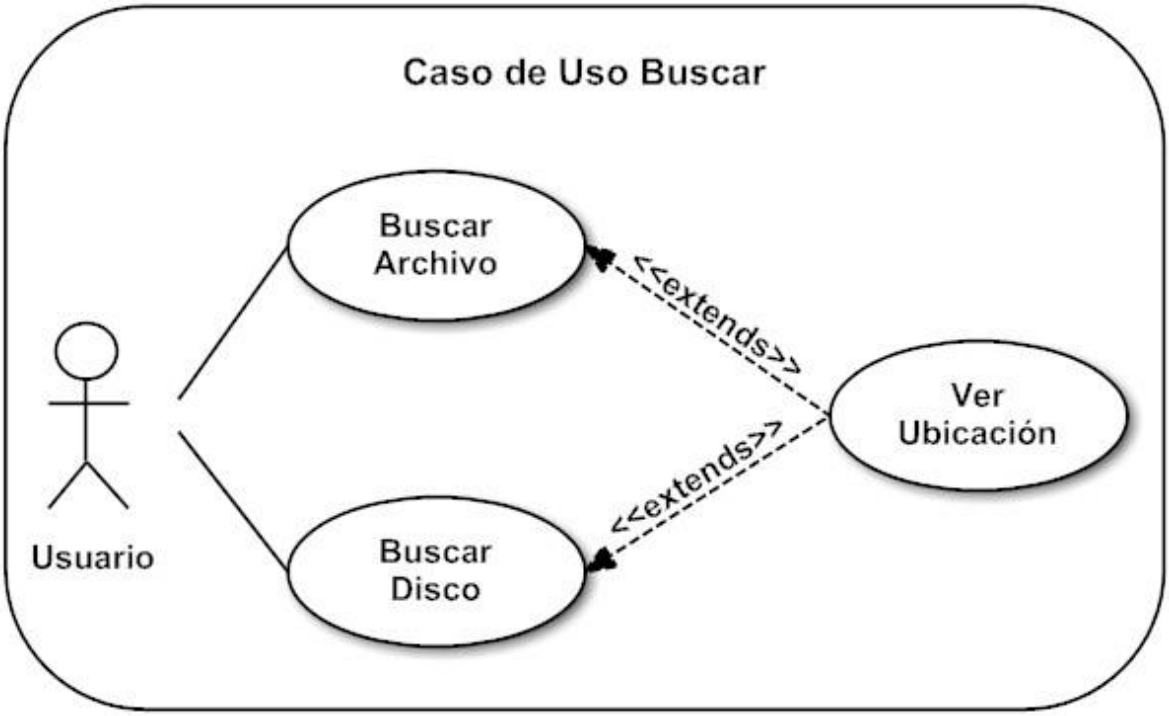


Ilustración 6.2 Caso de uso Buscar

Nombre:	**Buscar Archivo**
Descripción:	Permite encontrar en que disco está el archivo.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta. (ver Caso Abrir Base)
Flujo Principal: 1. Usuario hace clic en el menú, opción “búsqueda de archivo”. 2. Sistema despliega el modulo de búsqueda, un formulario para introducir términos respectivos a el archivo a buscar. 3. Usuario entra en el modulo de búsqueda. 4. Usuario especifica escribiendo en un cuadro de texto el nombre del archivo a buscar. 5. Sistema realiza búsqueda contra la base abierta y despliega los resultados en forma de malla de datos, donde se ve un conjunto de filas y columnas que contienen todos los datos* de los archivos que coincidieron con el termino buscado. *Todos los datos: nombre, tamaño, fecha de creación, fecha de modificación, extensión, etc..	
Flujo Alternativo: 4.a Usuario especifica en que categoría desea restringir su búsqueda. 4.b Usuario especifica entre que fechas de creación desea buscar el archivo. 4.c Usuario especifica que extensión debe tener el archivo. 4.d Usuario especifica que tamaño o entre que rango debe estar el archivo buscado.	
Post-condiciones:	No hay.

Nombre:	**Buscar Disco**
Descripción:	Permite encontrar un disco en la base.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.

Flujo Principal:

1. Usuario hace clic en el menú, opción ”búsqueda de disco”.
2. Sistema despliega el modulo de búsqueda, un formulario para introducir términos respectivos al disco a buscar.
3. Usuario entra en el modulo de búsqueda.
4. Usuario especifica escribiendo en un cuadro de texto el nombre del disco a buscar.
5. Sistema realiza búsqueda contra la base abierta y despliega los resultados en forma de malla de datos, donde se ve un conjunto de filas y columnas que contienen todos los datos* del o los discos que coincidieron con el termino buscado.

   *Todos los datos: nombre, cantidad de archivos, fecha de creación, prestado, serial, formato, etc..

Flujo Alternativo:

4.a Usuario especifica en que categoría desea restringir su búsqueda.

Post-condiciones:	No hay.

Nombre:	**Ver Ubicación <<extends>> Buscar Archivo.**
Descripción:	Permite encontrar en que disco está el archivo.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta, haber creado anteriormente una ubicación y haber realizado una búsqueda de un archivo.

Flujo Principal:

1. Usuario selecciona el archivo de la lista de resultados que entrego la búsqueda.(ver caso Buscar Archivo)
2. Sistema habilita el conjunto de opciones disponibles para el archivo seleccionado.
3. Usuario hace click en la opción “ver ubicación”.
4. El sistema despliega el modulo de ubicación, donde se puede ver una representación virtual de la ubicación física del disco en donde se encuentra el archivo.
5. El sistema genera una descripción textual de la ubicación.

Flujo Alternativo:

2.a El sistema no puede habilitar la opción de “ver ubicación” porque no se ha creado una.
a.1 El sistema sugiere al usuario que cree una ubicación.
a.2 El Usuario cancela y termina el caso de uso.

Post-condiciones:	No hay.

6.1.2.- Discos

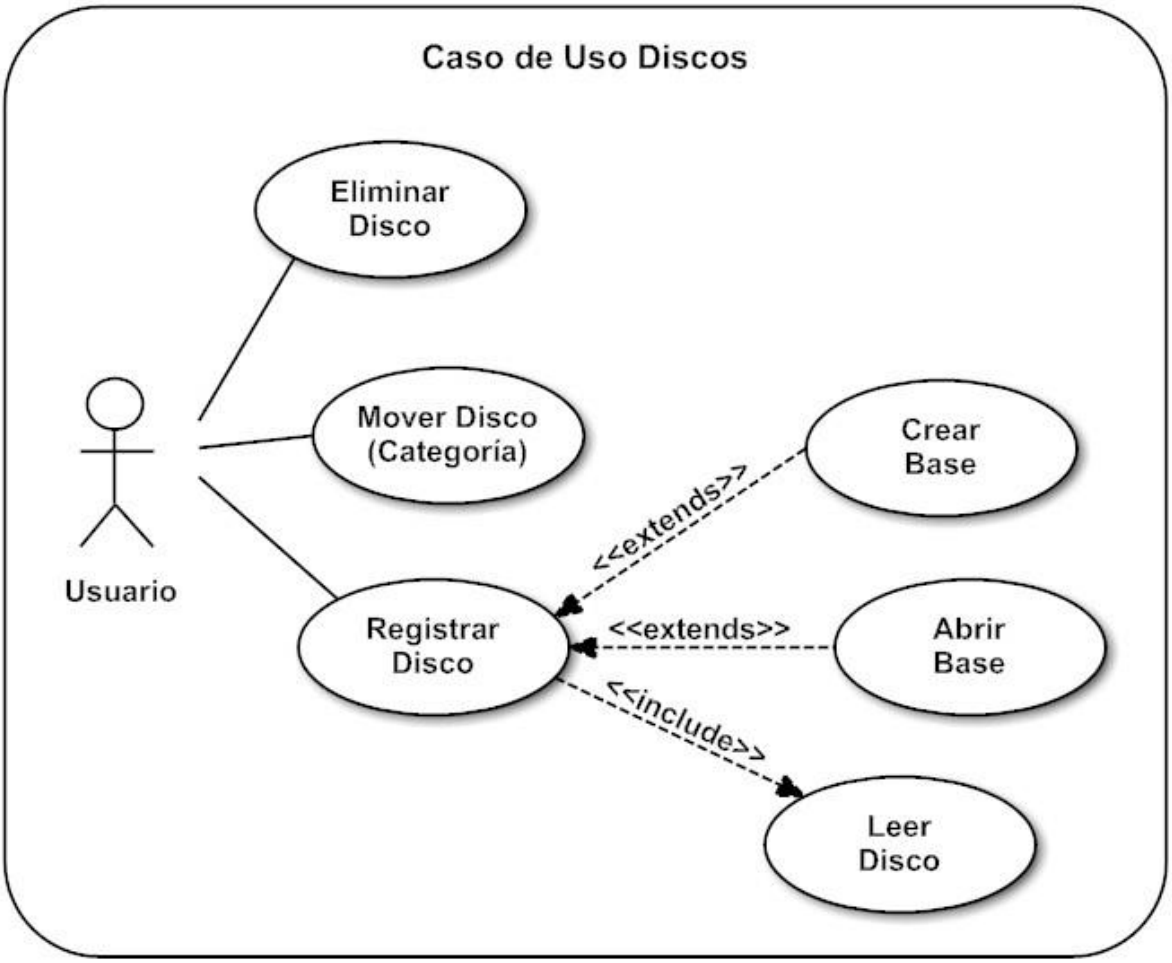


Ilustración 6.3 Caso de uso discos

Nombre:	**Crear Base <<extends>> Registrar disco**
Descripción:	Permite especificar una ubicación en el disco duro donde se va a generar y almacenar la base con la información de los discos.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Post-condiciones:	Una base abierta y lista para empezar a recibir discos.

Nombre:	**Abrir Base <<extends>> Registrar disco**
Descripción:	Permite buscar y seleccionar la base (previamente creada) en la cual se va guardar la información de los discos.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Post-condiciones:	Una base abierta y lista para empezar a recibir discos.

Nombre:	**Registrar Disco <<include>> Leer Disco**
Descripción:	Permite realizar el proceso a través del cual se almacena un disco en la base.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.

Flujo Principal:

1. El usuario abre una base. (Ver Crear Base o Abrir Base)
2. Usuario hace clic en el menú “Unidades”.
3. El sistema escanea y habilita las unidades posibles para efectuar la lectura de disco, luego despliega la lista con ellas.
4. El Usuario selecciona la unidad donde se encuentra el disco que desea almacenar.
5. El sistema lee y asocia la información del disco a la base*.
6. El sistema despliega la vista de explorador de la base con el nuevo disco agregado.

*El sistema guarda automáticamente la base.

Flujo Alternativo:

3.a El sistema escanea y no encuentra ninguna unidad lista. Despliega un mensaje comunicando esto al usuario.
a.1 El usuario introduce un disco en la unidad y vuelve al paso 1.

Post-condiciones:	Base actualizada.

Nombre:	**Eliminar Disco**
Descripción:	Permite borrar por completo las referencias a un disco.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.

Flujo Principal:

1. Usuario hace clic en el menú, opción ”Eliminar disco”.
2. El sistema despliega un formulario con los discos actualmente en la base.
3. El usuario selecciona un disco de la lista y acepta.
4. El sistema advierte que esta operación es totalmente irreversible y pide confirmación para proceder.
5. El usuario confirma.
6. El sistema elimina el disco y guarda la base actualizada.

Flujo Alternativo:

5. a. El usuario cancela la operación.

   a. El sistema no elimina el disco y vuelve a su estado normal.

Post-condiciones:	No hay

Nombre:	**Mover Disco (Categoría)**
Descripción:	Permite cambiar de una a otra categoría un disco en concreto.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.

Flujo Principal:

1. El usuario entra en el modo explorador.
2. Sistema despliega el total de discos en forma de árbol y vista de lista.
3. Usuario selecciona un disco en la vista de árbol, luego hace clic en el menú, opción “Cambiar de Categoría”.
4. El sistema despliega un formulario con la lista de categorías disponibles en la base, exceptuando la del disco actual.
5. El usuarios selecciona una y acepta.
6. El sistema asocia el disco a la nueva categoría y actualiza la base.

Flujo Alternativo:

4.a No hay mas categorías aparte de la actual, el sistema advierte de esto al usuario y le propone crear una nueva categoría o cancelar la operación.
a.1 El usuario acepta crear una nueva categoría. (ver caso de uso Crear Categoría)

Post-condiciones:	No hay

6.1.3.- Contactos

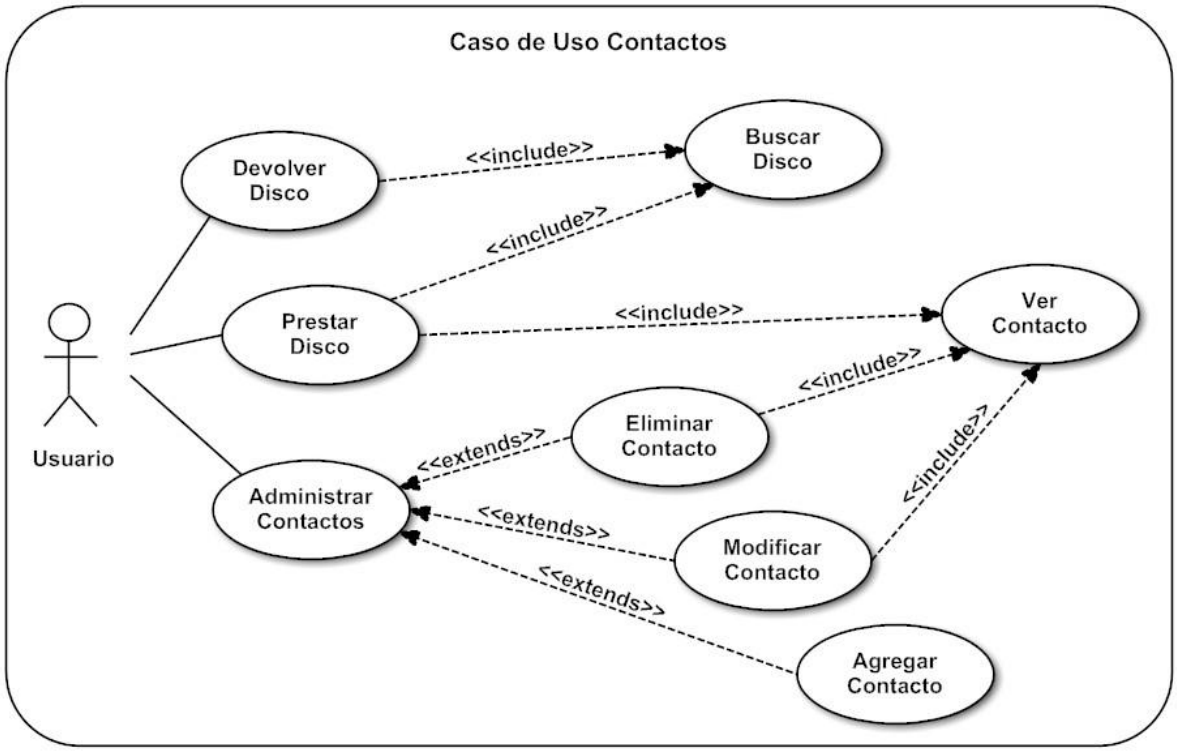


Ilustración 6.4 Caso de uso contactos

Nombre:	**Prestar Disco <<include>> Buscar Disco, Buscar Contacto**
Descripción:	Permite asociar un contacto con un disco.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta, discos y contactos previamente cargados en la base.
Flujo Principal: 1. El usuario hace clic en el menú, opción “Prestar disco”. 2. El sistema presenta un formulario con dos pasos, el primero consiste en seleccionar un disco de la base y el segundo en seleccionar un contacto de la base. 3. El usuario hace clic en el botón del primer paso que lo lleva al caso de uso buscar disco. 4. Luego de seleccionado el disco el usuario hace clic en el botón seleccionar contacto que lo lleva al caso de uso buscar contacto. 5. El sistema habilita luego de haber completado los dos pasos el botón prestar, el cual asocia el disco al contacto en la fecha actual. 6. El usuario hace clic en “Prestar”. 7. El sistema hace la asociación y actualiza la base.	
Flujo Alternativo: 6.a El usuario hace clic en “Cancelar”. 6.a.1 El sistema vuelve a la vista de explorador.	
Post-condiciones:	No hay

Nombre:	**Devolver Disco <<include>> Buscar Disco**
Descripción:	Permite encontrar un disco que está prestado y registrar su devolución.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta, discos prestados previamente.

Flujo Principal:

1. El usuario hace clic en el menú, opción “Devolver disco”.
2. El sistema despliega un formulario con los discos prestados actualmente, un botón buscar disco prestado, la fecha actual , un botón devolver y un botón cancelar.
3. El usuario selecciona el disco de la lista, verifica que los datos sean correctos.
4. Usuario hace clic en “Devolver”
5. El sistema le pide confirmación.
6. El usuario confirma y acepta.
7. El sistema registra la operación y actualiza la base.

Flujo Alternativo:

3.a El usuario hace clic en “BUSCAR DISCO PRESTADO” lo que lo lleva al caso de uso buscar disco, pero tomando en cuenta sólo los disco prestados.
Luego pasa al paso 4.

Post-condiciones:	No hay

Nombre:	**Administrar Contactos**
Descripción:	Permite Agregar, modificar y eliminar contactos.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.

Flujo Principal:

1. El usuario hace clic en el menú, opción “Administrar contactos”
2. El sistema despliega un formulario con la lista de los contactos actuales, teniendo la opción de agregar un nuevo contacto, modificar o eliminar uno existente.
3. El usuario escoge “Agregar un nuevo contacto” (ver caso de uso Agregar contacto).
4. Al terminar de agregar el contacto el caso termina.

Flujo Alternativo:

3.a El usuario escoge “Modificar un contacto” (ver caso de uso Modificar contacto).
3.b El usuario escoge “Eliminar un contacto” (ver caso de uso Eliminar contacto).

Post-condiciones:	No hay

Nombre:	**Buscar Contacto**
Descripción:	Permite encontrar un contacto en la base.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.

Flujo Principal:

1. Usuario hace clic en el menú, opción ”buscar un contacto”.
2. Sistema despliega el modulo de contactos, un formulario para introducir términos respectivos al contacto a buscar.
3. Usuario entra en el modulo de contactos.
4. Usuario especifica escribiendo en un cuadro de texto el nombre del contacto a buscar.
5. Sistema realiza búsqueda contra la base abierta y despliega los resultados en forma de malla de datos, donde se ve un conjunto de filas y columnas que contienen todos los datos* del o los contactos que coincidieron con el termino buscado.
6. El usuario selecciona el que buscaba.

*Todos los datos: nombres, apellidos, titulo, sobrenombre, lista de e-mails, dirección, código postal, ciudad, provincia, país, teléfono, fax, móvil, etc..

Flujo Alternativo:

4.a Usuario especifica escribiendo en un cuadro de texto el e-mail del contacto a buscar.
4.b Usuario especifica escribiendo en un cuadro de texto la dirección del contacto a buscar.
4.c Usuario especifica escribiendo en un cuadro de texto el teléfono del contacto a buscar.

Post-condiciones:	No hay.

Nombre:	**Agregar Contacto <<extends>> Administrar Contactos**
Descripción:	Permite realizar el proceso a través del cual se crea un nuevo contacto.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.

Flujo Principal:

1. Usuario hace clic en el menú, opción “agregar contacto”
2. El sistema despliega un formulario con las entradas de datos* correspondientes a los contactos.
3. El usuario rellena las entradas , nombre y apellido, hace clic en aceptar.
4. El sistema asocia el contacto y actualiza la base.

* datos: nombre, apellido, titulo, sobrenombre, lista de e-mails, dirección, código postal, ciudad, provincia, país, teléfono, fax, móvil, etc..

Flujo Alternativo:

3.a El usuario hace clic en aceptar sin rellenar ningún dato.
a.1 El sistema le advierte que debe llenar a lo menos nombre para crear el contacto.
Vuelve al paso 2.

Post-condiciones:	Base actualizada.

Nombre:	**Eliminar Contacto <<extends>> Administrar Contactos <<include>> Buscar Contacto**
Descripción:	Permite realizar el proceso a través del cual se borra un contacto.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener contactos en la base.

Flujo Principal:

1. Usuario hace clic en el menú, opción “Eliminar contacto”
2. El sistema presenta el formulario de buscar contactos (ver caso de uso BUSCAR CONTACTO)
3. Luego de escogido el contacto el sistema advierte al usuario si realmente desea eliminar el contacto, ya que es una operación irreversible.
4. El usuario confirma, hace clic en aceptar.
5. El sistema elimina el contacto y actualiza la base.

Flujo Alternativo:

4.a El usuario cancela y se termina el caso.

Post-condiciones:	Base actualizada.

Nombre:	**Modificar Contacto <<extends>> Administrar Contactos <<include>> Buscar Contacto**
Descripción:	Permite realizar el proceso a través del cual se modifica un contacto.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener contactos en la base.

Flujo Principal:

1. Usuario hace clic en el menú, opción “Modificar contacto”
2. El sistema presenta el formulario de buscar contactos (ver caso de uso BUSCAR CONTACTO)
3. Luego de escogido el contacto el sistema despliega un formulario con las entradas de datos* correspondientes al contacto.
4. El usuario modifica la entrada, e-mail, hace clic en aceptar.
5. El sistema actualiza el contacto y actualiza la base.

* datos: nombre, apellido, titulo, sobrenombre, lista de e-mails, dirección, código postal, ciudad, provincia, país, teléfono, fax, móvil, etc..

Flujo Alternativo:

4.a El usuario cancela y se termina el caso.
4.b El usuario hace clic en aceptar sin modificar ningún dato.
b.1 El sistema no hace nada y termina el caso.
4.b El usuario modifica cualquier dato o datos y acepta.
Sigue con el paso 5.

Post-condiciones:	Base actualizada.

6.1.4.- Categorías

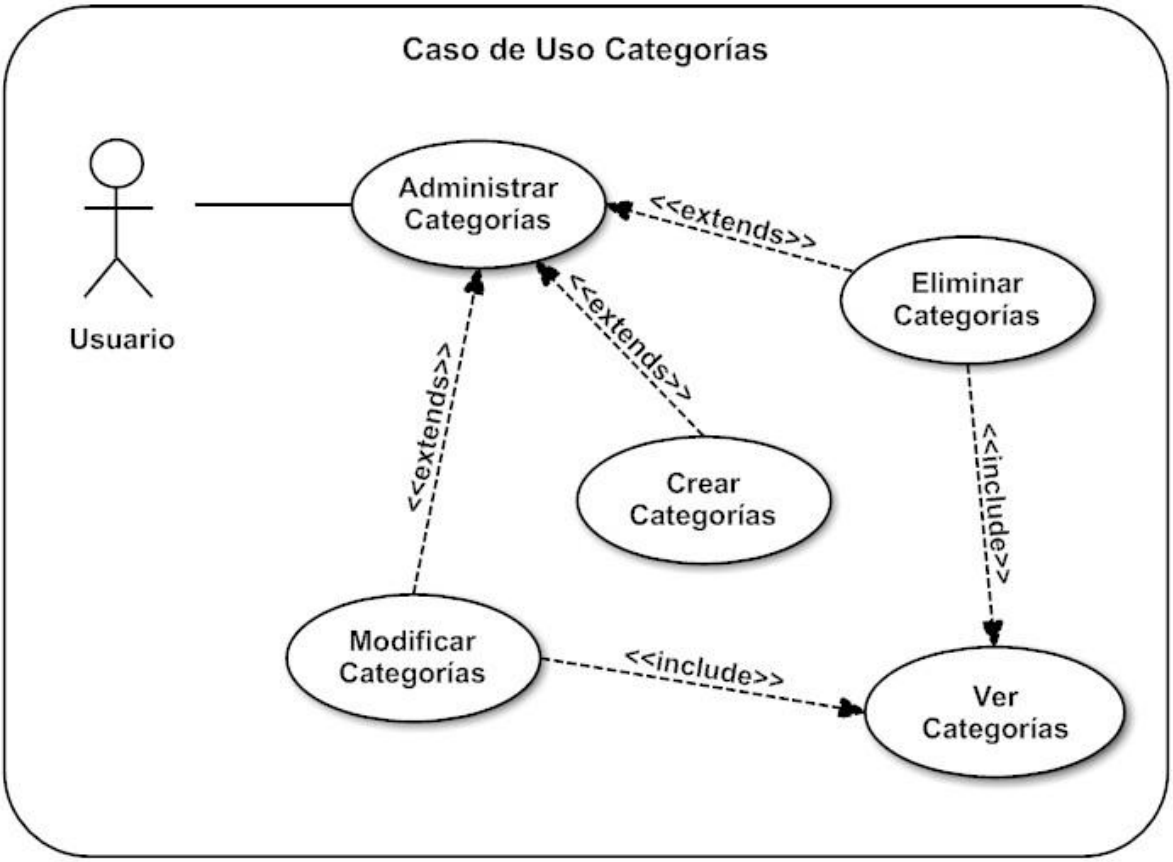


Ilustración 6.5 Caso de uso categorías

Nombre:	**Administrar Categorías**
Descripción:	Permite crear, modificar y eliminar categorías.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta.
Flujo Principal: 1. El usuario hace clic en el menú, opción “Administrar categorías” 2. El sistema despliega un formulario con la lista de las categorías actuales, teniendo la opción de crear una nueva categoría, modificar o eliminar una existente. 3. El usuario escoge “Crear nueva categoría” (ver caso de uso crear categoría). 4. Al terminar de crear la categoría el caso termina.	
Flujo Alternativo: 3.a El usuario escoge “Modificar una categoría” (ver caso de uso Modificar categorías). 3.b El usuario escoge “Eliminar una categoría” (ver caso de uso Eliminar categorías).	
Post-condiciones:	No hay

Nombre:	**Ver Categorías**
Descripción:	Permite encontrar y seleccionar una categoría de la base.
Actores:	Usuario
Pre-condiciones:	Tener una base abierta, categorías creadas previamente.
Flujo Principal: 1. Usuario hace clic en el menú, opción ”Ver categorías”. 2. Sistema despliega un formulario con la lista de categorías disponibles en la base. 3. El usuario selecciona una de la lista y acepta. 4. El sistema guarda la selección para su uso posterior (ver modificación y eliminación), termina el caso.	
Flujo Alternativo: 3.a Usuario hace clic en cancelar y termina el caso de uso sin seleccionar categoría.	
Post-condiciones:	No hay.

Nombre:	**Crear Categoría <<extends>> Administrar Categorías**
Descripción:	Permite realizar el proceso a través del cual se crea una nueva categoría.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Flujo Principal: 1. Usuario hace clic en el menú, opción “crear categoría” 2. El sistema despliega un formulario solicitando el nombre de la categoría. 3. El usuario rellena el nombre y hace clic en aceptar. 4. El sistema asocia la categoría y actualiza la base.	
Flujo Alternativo: 3.a El usuario hace clic en aceptar sin rellenar ningún dato. a.1 El sistema le advierte que debe llenar el nombre para crear la categoría. *Vuelve al paso 2.*	
Post-condiciones:	Base actualizada, categoría creada.

Nombre:	**Modificar Categoría <<extends>> Administrar Categorías <<include>> Ver Categoría**
Descripción:	Permite realizar el proceso a través del cual se modifica una categoría.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener categorías en la base.
Flujo Principal: 1. Usuario hace clic en el menú, opción “Modificar categoría” 2. El sistema presenta el formulario con la entrada de datos nombre de categoría 3. El usuario modifica el nombre hace clic en aceptar. 4. El sistema actualiza la base.	
Flujo Alternativo: 3.a El usuario cancela. a.1 El sistema no actualiza	
Post-condiciones:	Base actualizada.

Nombre:	**Eliminar Categoría <<extends>> Administrar Categorías <<include>> Ver Categoría**
Descripción:	Permite realizar el proceso a través del cual se elimina una categoría.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener categorías en la base.
Flujo Principal: 1. Usuario hace clic en el menú, opción “Eliminar categoría” 2. El sistema presenta el formulario con las categorías. 3. El usuario elimina la categoría y hace clic en aceptar. 4. El sistema actualiza la base.	
Flujo Alternativo: 3.a El usuario cancela. a.1 El sistema no actualiza *Sigue con el paso 2.*	
Post-condiciones:	Base actualizada.

6.1.5.- Etiquetas

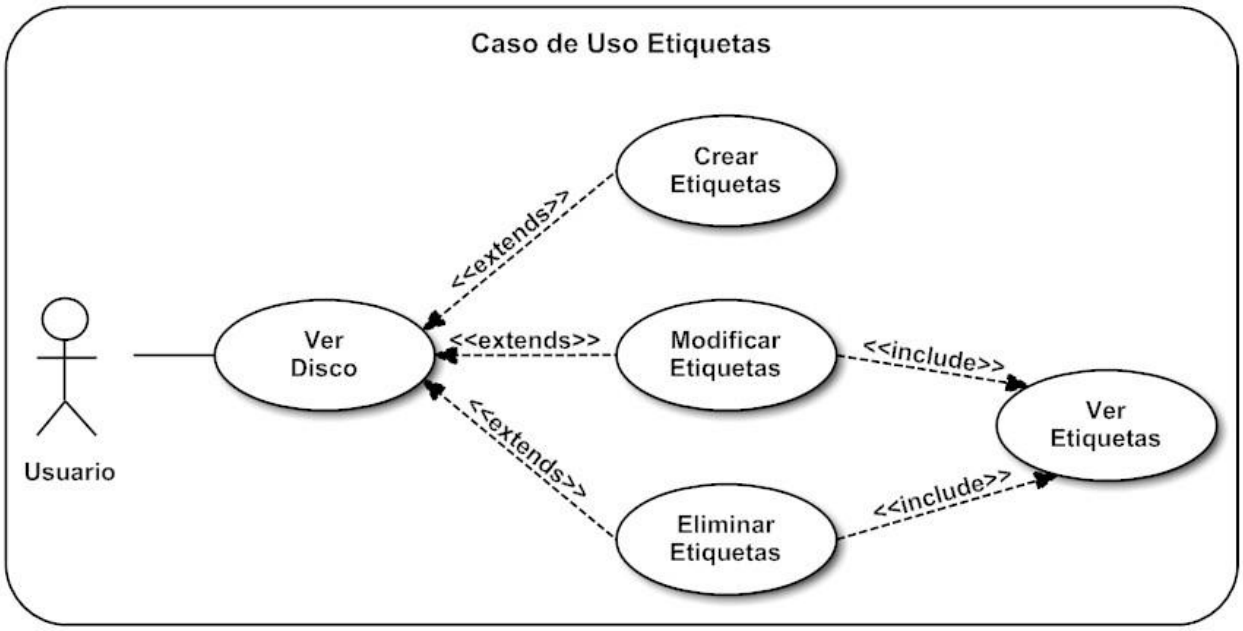


Ilustración 6.6 Caso de uso etiquetas

Nombre:	**Ver Discos**
Descripción:	Permite ver el contenido de los discos y realizar ciertas acciones sobre ellos.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, ingresar al modulo de búsqueda para obtener el disco deseado
Post-condiciones:	No hay.

Nombre:	**Crear Etiqueta <<extends>> Ver Disco**
Descripción:	Permite realizar el proceso a través del cual se agrega una etiqueta.
Actores:	Usuario
Pre-condiciones:	Haber encontrado el disco al cual se le desea agregar la etiqueta
Flujo Principal: 1. Usuario hace clic en el menú, opción “crear etiqueta” 2. El sistema despliega un formulario solicitando el nombre de la etiqueta. 3. El usuario rellena el nombre y hace clic en aceptar. 4. El sistema asocia la etiqueta y actualiza la base.	
Flujo Alternativo: 3.a El usuario hace clic en aceptar sin rellenar ningún dato. a.1 El sistema le advierte que debe llenar el nombre para crear la etiqueta. *Vuelve al paso 2.* 3.b El usuario no acepta b.1 El sistema no actualiza la base.	
Post-condiciones:	Base actualizada, etiqueta creada.

Nombre:	**Modificar Etiquetas**
Descripción:	Permite realizar el proceso a través del cual se modifica una etiqueta del disco encontrado.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener etiquetas sobre el disco encontrado.
Flujo Principal: 1. Usuario hace clic en el menú, opción “Modificar Etiquetas” 2. El sistema presenta el formulario con la entrada de datos nombre de etiqueta (Ver etiquetas) 3. El usuario modifica la etiqueta hace clic en aceptar. 4. El sistema actualiza la base.	
Flujo Alternativo: 3.a El usuario cancela. a.1 El sistema no actualiza	
Post-condiciones:	Base actualizada.

Nombre:	**Eliminar Etiqueta**
Descripción:	Permite realizar el proceso a través del cual se elimina una etiqueta.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa, tener etiquetas sobre el disco encontrado.
Flujo Principal: 1. Usuario hace clic en el menú, opción “Eliminar etiqueta” 2. El sistema presenta el formulario con las etiquetas (Ver etiquetas). 3. El usuario elimina la etiqueta y hace clic en aceptar. 4. El sistema actualiza la base.	
Flujo Alternativo: 3.a El usuario cancela. a.1 El sistema no actualiza	
Post-condiciones:	Base actualizada.

Nombre:	**Ver Etiquetas**
Descripción:	Permite ver y seleccionar etiquetas de un disco previamente seleccionado realizar operaciones (modificar o eliminar) sobre estas.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Post-condiciones:	Una base abierta y lista para empezar a recibir discos.

6.1.6.- Ubicación

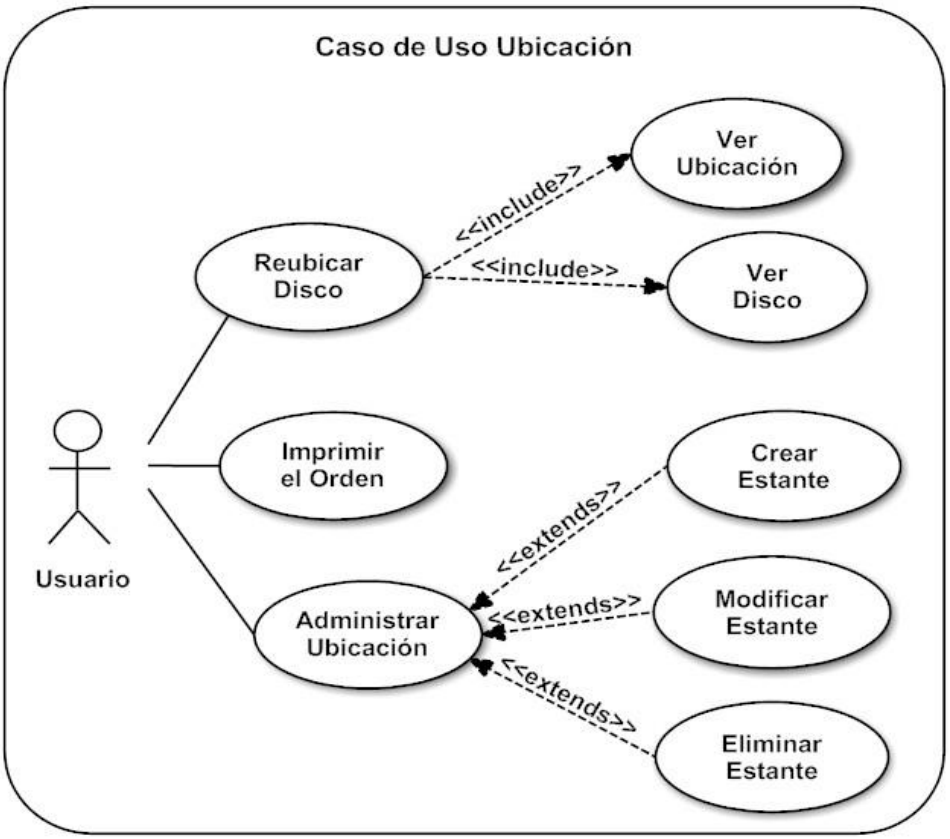


Ilustración 6.7 Caso de uso ubicación

Nombre:	**Administrar Ubicación**
Descripción:	Permite crear, modificar y eliminar estantes.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Flujo Principal: 1. El usuario hace clic en el menú, opción “Administrar Ubicación” 2. El sistema despliega un formulario con las opciones actuales, teniendo la opción de crear estantes, modificar o eliminar deshabilitadas. 3. El usuario escoge “crear estante” (ver caso de uso crear estante). 4. Al terminar de crear el estante el caso termina.	
Flujo Alternativo: 3.a El usuario escoge “Modificar un estante” (ver caso de uso Modificar estante). 3.b El usuario escoge “Eliminar un estante” (ver caso de uso Eliminar estante).	
Post-condiciones:	Base actualizada.

Nombre:	**Ver Ubicación**
Descripción:	Permite ver de manera representativa el contexto donde se ubica un disco en un lugar físico.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Flujo Principal: 1. Usuario hace clic en el menú “Ubicación”. 2. El sistema despliega el modulo de ubicación en el cual se puede ver una lista que representa la estructura de los contenedores de los discos, y una vista de los discos en fila según la naturaleza de su contenedor. 3. El usuario hace clic sobre una figura que representa el disco. 4. El sistema muestra las principales características del disco.	
Flujo Alternativo: 3.a El usuario hace doble clic sobre la figura que representa el disco. a.1 El sistema despliega el modo explorador con el disco seleccionado, mostrando el conten del disco.	
Post-condiciones:	Base actualizada.

Nombre:	**Imprimir el Orden**
Descripción:	Permite imprimir el orden en que se ubican los discos en los contenedores, para ordenar físicamente los discos.
Actores:	Usuario
Pre-condiciones:	Abrir el Programa.
Flujo Principal: 1. Usuario hace clic en el menú “Ubicación”. 2. El sistema despliega el modulo de ubicación en el cual se puede ver una lista que representa la estructura de los contenedores de los discos, y una vista de los discos en fila según la naturaleza de su contenedor. 3. El usuario hace clic sobre la opción “Imprimir Ubicación”. 4. El sistema muestra un formulario en el cual se puede elegir que estantes incluir en la impresión así como también que campos de los disco incluir. 5. El usuario selecciona algunas opciones y acepta. 6. El sistema genera el informe y lo manad a impresión.	
Flujo Alternativo: 4.a El Sistema no detecta una impresora, le advierte al usuario. Termina el caso. 5.a El usuario cancela. Termina el caso.	
Post-condiciones:	Base actualizada.

Nombre:	**Crear Estante <<extends>> Administrar Ubicación**
Descripción:	Permite realizar el proceso a través del cual se agrega un estante y así poder distribuir una lista de discos en sus filas.
Actores:	Usuario
Pre-condiciones:	Estar en el modulo de Ubicación
Flujo Principal: 1. Usuario hace clic en la opción “crear estante” 2. El sistema despliega un formulario solicitando el nombre del estante, cantidad de filas, numero de discos por fila. 3. El usuario rellena todos los campos y hace clic en aceptar. 4. El sistema crea el estante y actualiza la base.	
Flujo Alternativo: 3.a El usuario hace clic en aceptar sin rellenar ningún dato. a.1 El sistema le advierte que debe llenar todos los datos para crear el estante. *Vuelve al paso 2.* 3.b El usuario no acepta b.1 El sistema no actualiza la base.	
Post-condiciones:	Base actualizada, estante creado.

Nombre:	**Modificar Estante <<extends>> Administrar Ubicación**
Descripción:	Permite realizar el proceso a través del cual se modifican los parámetros de un estante.
Actores:	Usuario
Pre-condiciones:	Estar en el modulo de Ubicación, haber creado estantes previamente.
Flujo Principal: 1. Usuario hace clic en la opción “Modificar estante” 2. El sistema despliega un formulario mostrando el nombre del estante, cantidad de filas, numero de discos por fila. 3. El usuario cambia el nombre y hace clic en aceptar. 4. El sistema actualiza el estante y actualiza la base.	
Flujo Alternativo: 3.a El usuario hace clic en aceptar sin cambiar ningún dato. a.1 El sistema comprueba que no hubo cambios y no hace nada. 3.b El usuario cambia el numero de filas y acepta. *Pasa al paso 4.* 3.c El usuario cambia el numero de discos por fila y acepta. *Pasa al paso 4.*	
Post-condiciones:	Base actualizada, estante modificado.

Nombre:	**Eliminar Estante <<extends>> Administrar Ubicación**
Descripción:	Permite realizar el proceso a través del cual se borra un estante.
Actores:	Usuario
Pre-condiciones:	Estar en el modulo de Ubicación, haber creado estantes previamente.
Flujo Principal: 1. Usuario hace clic en la opción “Eliminar estante” 2. El sistema despliega un formulario mostrando una lista con los nombres de los estantes. 3. El usuario selecciona el primero y hace clic en aceptar. 4. El sistema elimina el estante y actualiza la base.	
Flujo Alternativo: 3.a El usuario hace clic en cancelar. *Termina el caso.*	
Post-condiciones:	Base actualizada, estante modificado.

Nombre:	**Reubicar Disco <<include>> Ver Ubicación, Ver Disco.**
Descripción:	Permite realizar el proceso a través del cual se cambia la posición de un disco dentro del mismo u otro estante.
Actores:	Usuario
Pre-condiciones:	Estar en el modulo de Ubicación, haber creado estantes previamente, tener seleccionado un disco.
Flujo Principal: 1. Usuario hace clic en la opción “Reubicar Disco” 2. El sistema despliega un formulario mostrando una breve descripción del disco, la posición actual y los campos necesarios para definir la nueva posición. 3. El usuario selecciona una nueva posición y hace clic en aceptar. 4. El sistema verifica la nueva posición (está ocupada por otro disco), y pregunta si desea correr todos los discos o intercambiar solo las posiciones. 5.El usuario elige intercambiar posiciones y acepta. 6.El sistema actualiza las posiciones de ambos discos y actualiza la base.	
Flujo Alternativo: 5a El usuario elige correr todos los discos. El sistema re posiciona los discos tomando en cuenta la nueva lista de discos, finalmente actualiza la base. *Termina el caso.*	
Post-condiciones:	Base actualizada, estante modificado.

6.2.- Flujos de Trabajo (Workflows)

Aquí se muestran las principales actividades que el sistema remplazaría.

6.2.1.- Flujos de Trabajo Antes de la implantación del sistema

6.2.1.1.- Flujo "Buscar Archivo"

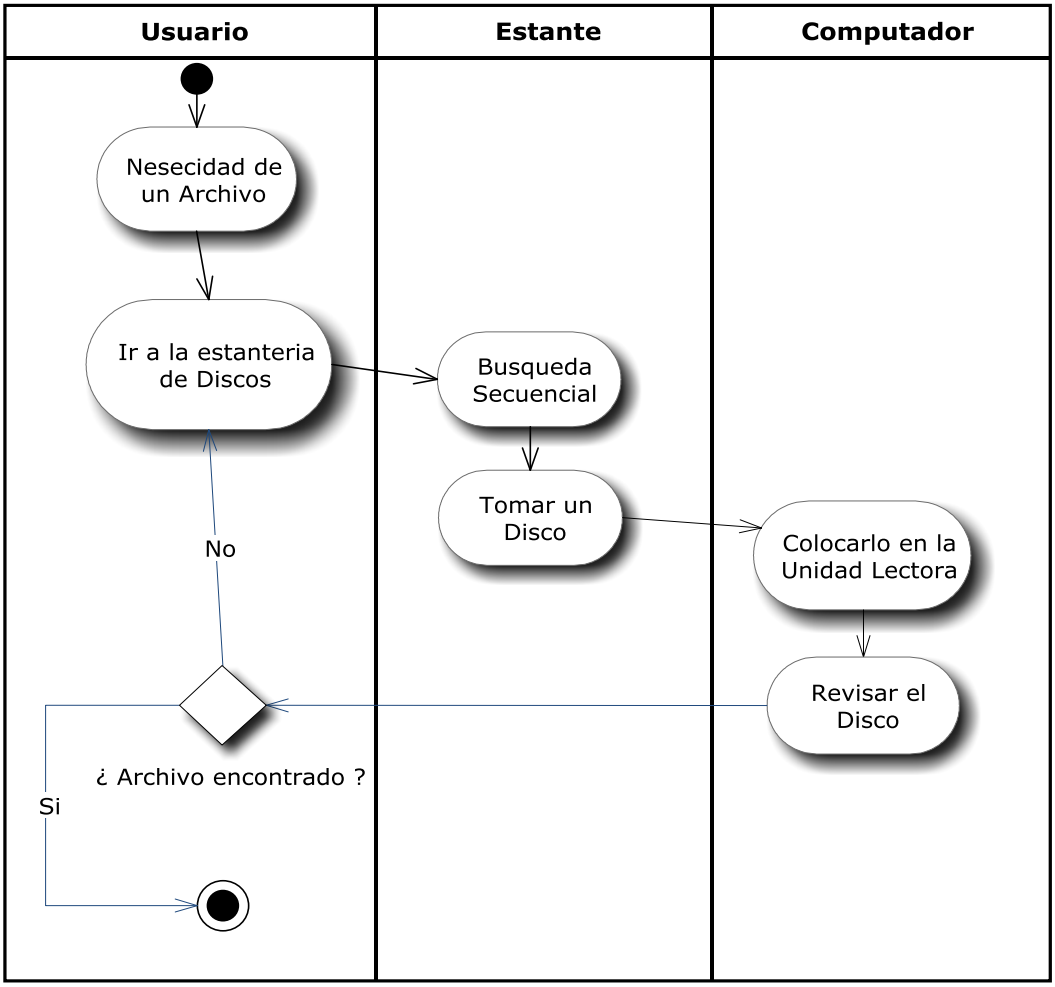


Ilustración 6.8 Diagrama de Actividad Buscar un Archivo en un Disco

Al no encontrar el "Archivo" en el primer disco se produce un loop, en el cual puede llegar a perderse mucho tiempo en cada iteración tratando de encontrar el archivo, hasta quizás nunca encontrarlo y percatarse de que fue prestado.

6.2.2.- Flujos de Trabajo Después de la implantación del sistema

6.2.2.1.- Flujo "Buscar Archivo"

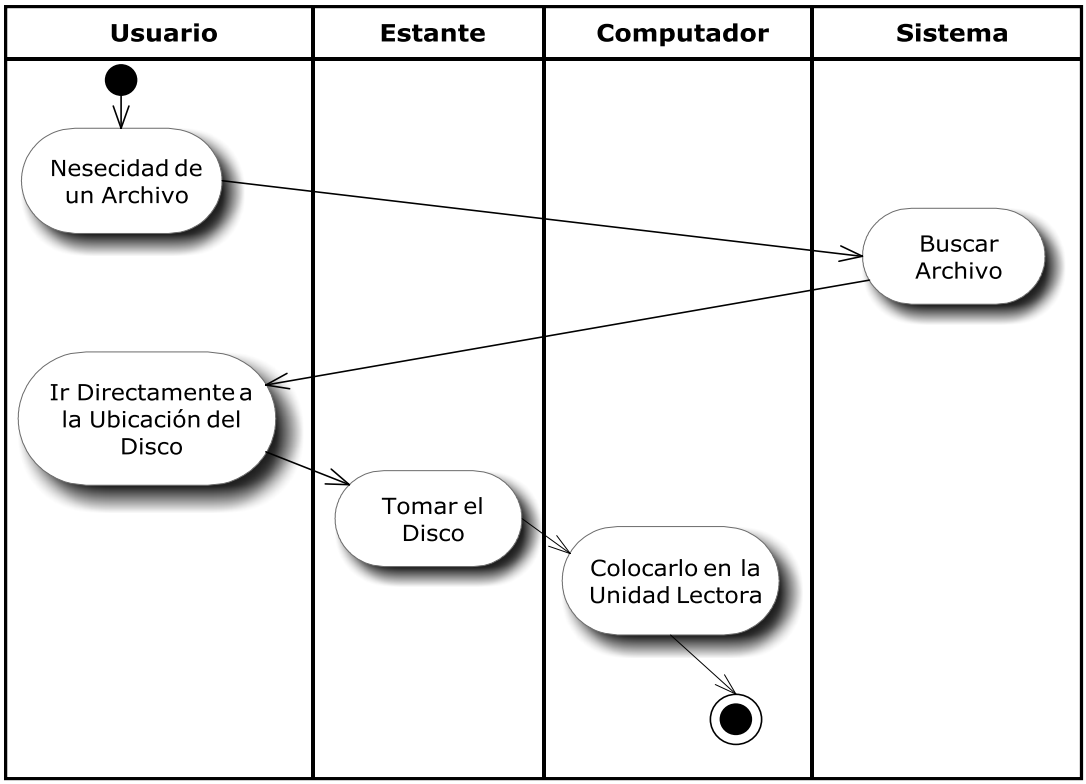


Ilustración 6.9 Diagrama de Actividad Buscar un Archivo en un Disco con Sistema

Gracias al Sistema se ha eliminado el loop que producía pérdida de tiempo, ahora el usuario hace una búsqueda previa en el Sistema el cual ha sido cargado anteriormente con la información de los discos y su ubicación, entonces este indica el disco exacto en el cual se encuentra el "Archivo" y mediante un sistema de representación virtual del almacenamiento de los discos en las estanterías, este entrega la posición exacta del disco en el estante.

6.3.- Arquitectura del Sistema

6.3.1.- Lógica

Desde el punto de vista lógico este sistema cuenta con 2 capas claramente marcadas, una es la capa de manejo de los datos, en este caso es a través de un “parser xml” que hace el trabajo de un motor de base de datos en el cual es posible encomendarle las tareas de administrar los datos de manera optima, incluso tiene un lenguaje de consulta al estilo del SQL, llamado XPath el cual permite realizar consultas similares a las del SQL.

La otra capa es la de presentación y reglas del negocio que se mezclan para darle un mayor desempeño al sistema, debido a la naturaleza del sistema y siendo una aplicación diseña para uso personal y de pequeñas oficinas, conviene que la presentación de los datos , o sea la interfaz con que el usuario interactúa, este estrechamente relacionada con las reglas del negocio, la forma en que se transforma y maneja la información.

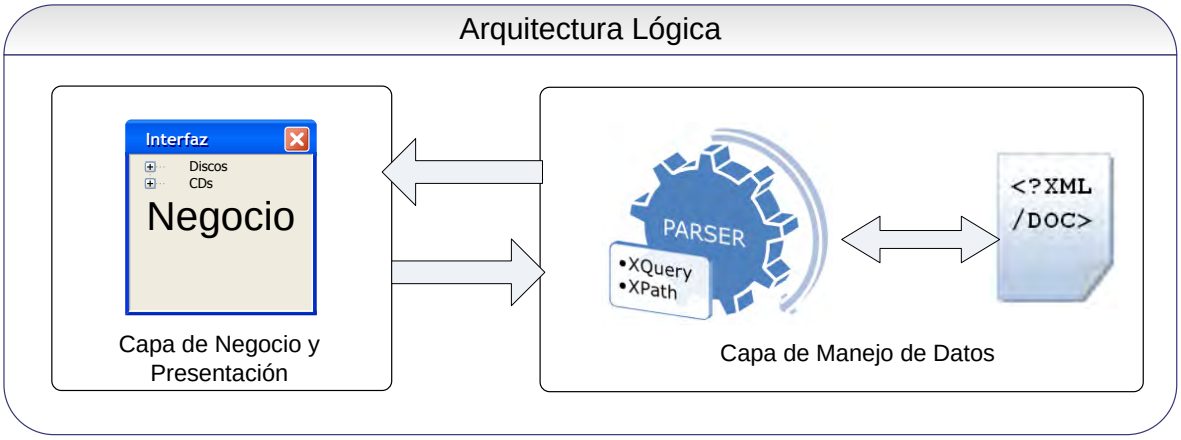


Ilustración 6.10 Arquitectura Lógica

6.3.2.- Física

Este sistema está formado por un solo nivel donde conviven las dos capas lógicas del software. Esto significa que toda la aplicación reside en un solo equipo físicamente.

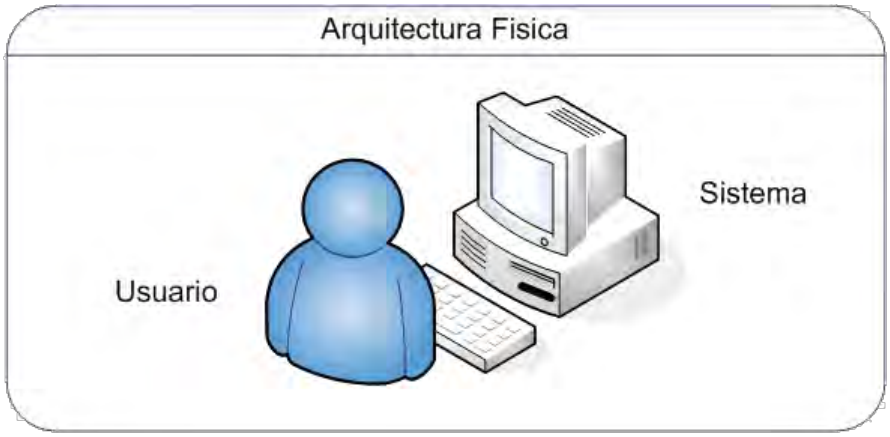


Ilustración 6.11 Arquitectura Física

6.4.- Diagrama del esquema de datos XML

En el “ANEXO A” de este informe se encuentra la definición del esquema y en el “ANEXO B” una base de datos de ejemplo perteneciente a este sistema.

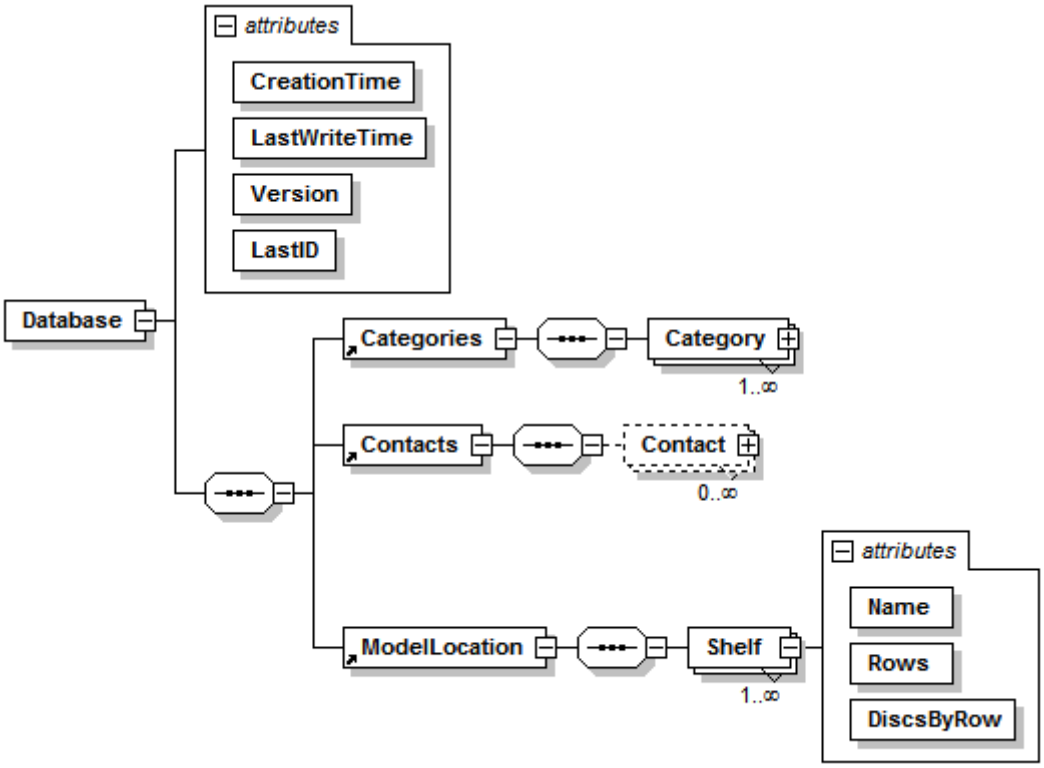


Ilustración 6.12 Esquema XML Parte 1

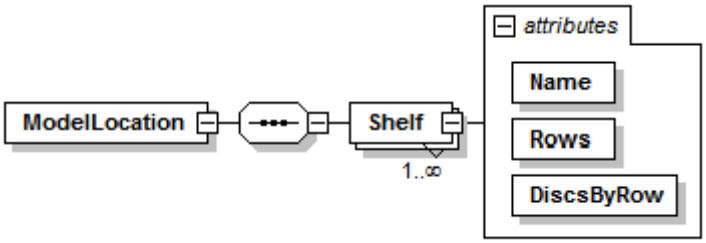


Ilustración 6.13 Esquema XML Parte 2

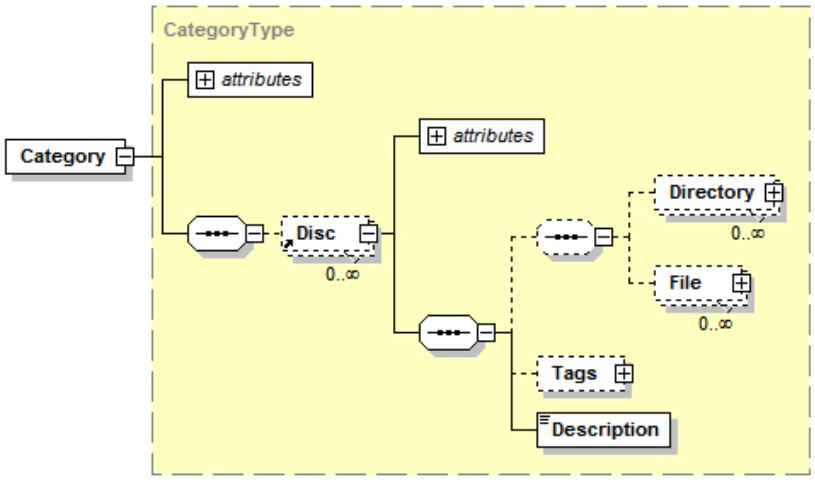


Ilustración 6.14 Esquema XML Parte 3

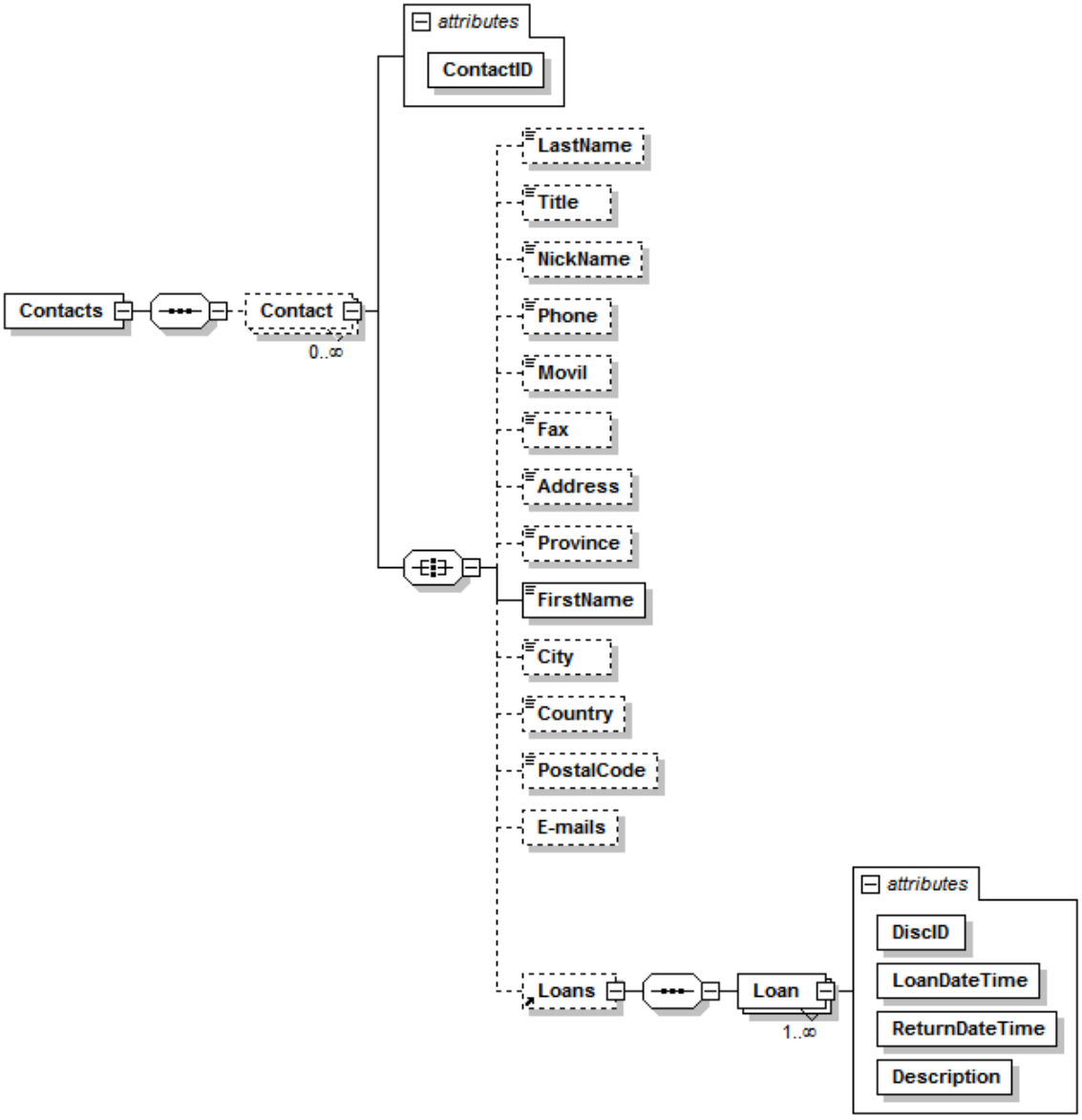


Ilustración 6.15 Esquema XML Parte 4

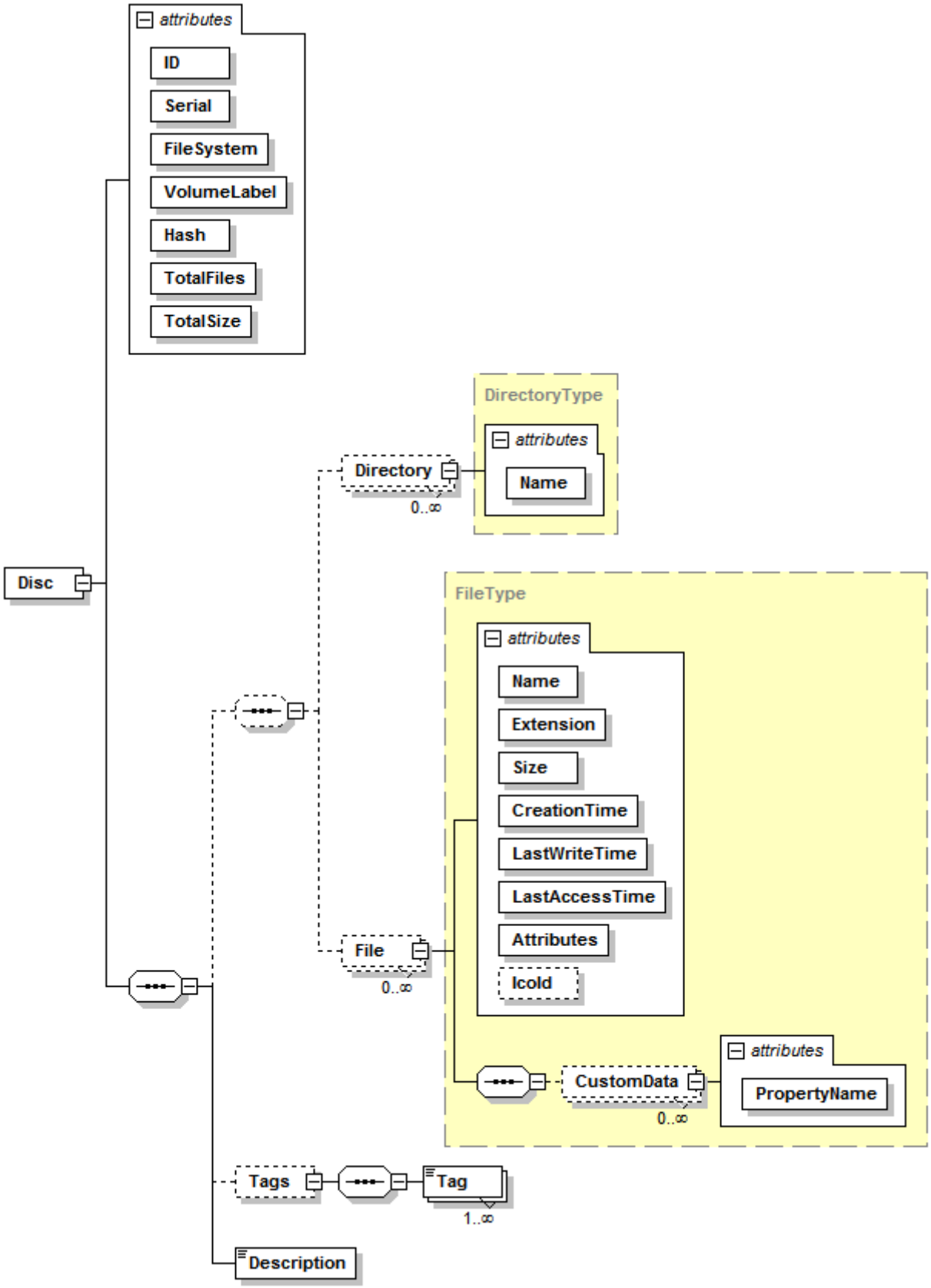


Ilustración 6.16 Esquema XML Parte 5

6.5.- Diagrama de clases

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modificara para satisfacer los detalles de las implementaciones que serán vistos más adelante.

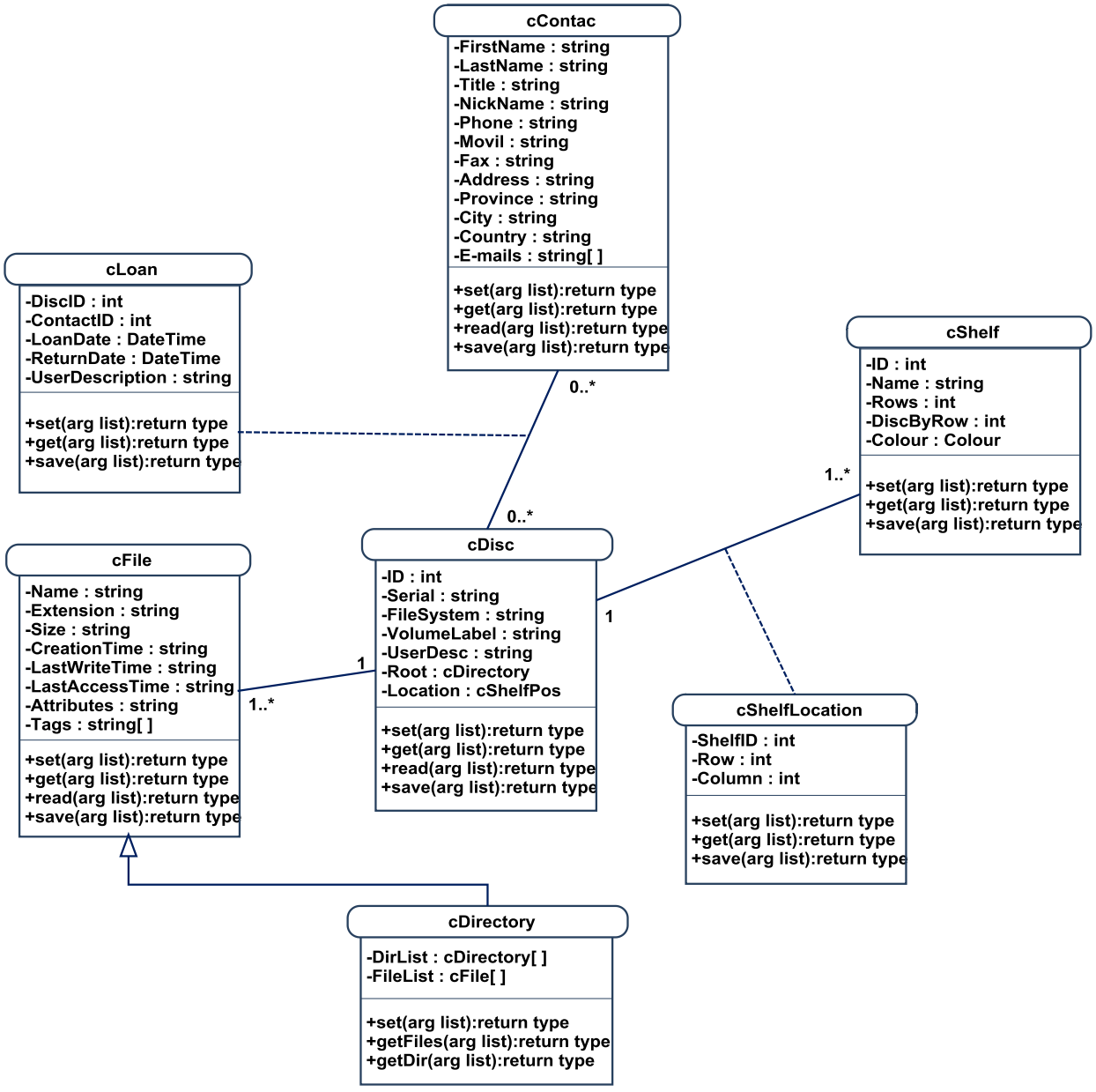


Ilustración 6.17 Diagrama de Clases

6.6.- Diseño preliminar de Interfaces de Usuario

Los diseños ya implementados de las interfaces se encuentran en el ANEXO E.

Ilustración 6.18 Interfaz Preliminar del Modo Explorador

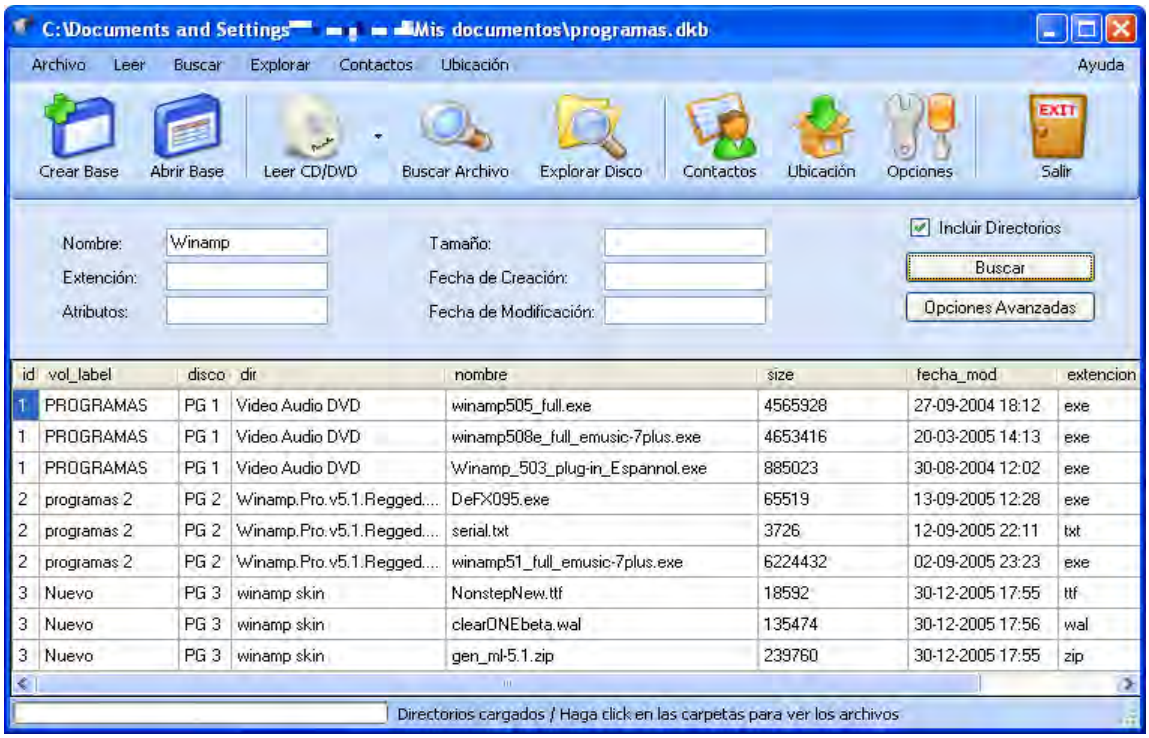


Ilustración 6.19 Interfaz preliminar del Modo Búsqueda de Archivo

7.- Pruebas a Utilizar en el Proyecto

Dado que el enfoque de Pruebas de Caja Blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental y que el sistema a realizar se desarrollará en un ambiente de escritorio, es que se utilizará el enfoque de pruebas de Caja Negra. Por que como se dirá a continuación la Prueba de la Caja Negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas intentan encontrar errores de distintas categorías, como por ejemplo; Funciones incorrectas o ausentes, Errores de interfaz, Errores en estructuras de datos o en accesos a bases de datos externas, Errores de rendimiento y Errores de iniciación y de terminación.

Para este caso se utilizará el enfoque de Pruebas Caja Negra, Pruebas de Causa – Efecto, y las pruebas de interfaces gráficas de usuario también se realizaran pruebas de usabilidad. En el siguiente punto se detallan los enfoques y pruebas a utilizar.

La aplicación de estos planes de pruebas estará a cargo de personal voluntario, externo al desarrollo del proyecto, serán pruebas realizadas de forma cerrada por un par de alumnos de la escuela de ingeniería de informática de la Pontificia Universidad Católica De Valparaíso.

Dentro del desarrollo de este proyecto no está contemplada la apertura de la aplicación al público en general, pero se espera que una vez terminadas las evaluaciones internas del proyecto la aplicación entre en un estado de pruebas beta abierto a todos los usuarios potenciales.

También se realizaron pruebas de usabilidad, las del tipo “Test de usabilidad”, las tareas se encuentran en el ANEXO F.

Se adjunta parte del Plan de Pruebas Causa – Efecto y Pruebas de Interfaces gráficas de usuarios en el ANEXO D.

7.1.- Enfoque de La Caja Negra

Los métodos de prueba de la caja negra se centran en los requisitos funcionales del software. O sea, la prueba de la caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de la caja negra *no* es una alternativa a las técnicas de prueba de la caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de la caja blanca.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de iniciación y de terminación.

A diferencia de la prueba de la caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de la caja negra tiende a aplicarse durante posteriores fases de prueba. Ya que la prueba de la caja negra ignora intencionadamente la estructura de control, centra su atención en el campo de información. Las pruebas se diseñan para responder a las siguientes preguntas:

- ¿Cómo se prueba la validez funcional?
- ¿*Qué clases* de entrada compondrán unos buenos casos de prueba?
- ¿Es el sistema particularmente sensible a ciertos valores de entrada?
- ¿De qué forma están aislados los límites de una clase de datos?
- ¿Qué volúmenes y miles de datos tolerará el sistema?
- ¿Qué efectos sobre la operación del sistema tendrán combinaciones específicas de datos?

Mediante las técnicas de prueba de la caja negra se deriva un conjunto de casos de prueba que satisfacen los siguientes criterios:

- Casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable.
- Casos de prueba que nos dicen algo sobre la presencia o ausencia de clases de errores en lugar de un error asociado solamente con la prueba, en particular, que se encuentre disponible.

7.2.- Técnicas del enfoque de caja negra

7.2.1.- Partición equivalente

La partición equivalente es un método de prueba de la caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (p. ej.: procesamiento incorrecto de todos los datos de caracteres) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una *condición de entrada. Una clase de equivalencia* representa un conjunto de estados válidos o inválidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

1. Si una condición de entrada especifica un *rango,* se define una clase de equivalencia válida y dos inválidas.

2. Si una condición de entrada requiere un *valor* específico, se define una clase de equivalencia válida y dos inválidas.

3. Si una condición de entrada especifica un miembro de un *conjunto,* se define una clase de equivalencia válida y una inválida.

4. Si una condición de entrada es *lógica,* se define una clase válida y una inválida.

7.2.2.- Análisis de valores límite

Por razones que no están del todo claras, los errores tienden a darse más en los límites del campo de entrada que en el "centro". Por ello, se ha desarrollado el *análisis de valores limites* (AVL) como técnica de prueba. El análisis de valores límite nos lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los "bordes" de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL deriva casos de prueba también para el campo de salida.

Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

1. Si una condición de entrada especifica un *rango* delimitado por los valores *a y* b, se deben diseñar casos de prueba para los valores a *y b* y para los valores justo por debajo y justo por encima de *a y* b, respectivamente.

2. Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.

3. Aplicar las directrices 1 y 2 a las condiciones de salida. Por ejemplo, supongamos que se requiere una tabla de "temperatura frente a presión" como salida de un programa de análisis de ingeniería. Se deben diseñar casos de prueba que creen un informe de salida que produzca el máximo (y el mínimo) número permitido de entradas en la tabla.

4. Si las estructuras de datos internas tienen límites preestablecidos (p. ej.: un array que tenga un límite definido de 100 entradas), hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

La mayoría de los ingenieros de software llevan a cabo de forma intuitiva alguna forma de AVL. Aplicando las directrices que se acaban de exponer, la prueba de límites será más completa y, por tanto, tendrá una mayor probabilidad de detectar errores.

7.3.- Pruebas de Usabilidad

7.3.1.- La Usabilidad

La definición de usabilidad conforme a la norma ISO 9241, parte 11 dice: "la usabilidad es el rango en el cual un producto puede ser usado por unos usuarios específicos para alcanzar ciertas metas especificadas con efectividad, eficiencia y satisfacción en un contexto de uso especificado" [16] .De hecho, la usabilidad no se limita a sistemas computacionales exclusivamente, sino que es un concepto aplicable a cualquier elemento en el cual se va a producir una interacción entre un humano y un dispositivo.

En el caso de los sistemas computacionales, la usabilidad va a abarcar desde el proceso de instalación de la aplicación hasta el punto en que el sistema sea utilizado por el usuario, incluyendo también el proceso de mantenimiento.

La usabilidad tiene cinco atributos definidos [17]

1. **Facilidad de aprendizaje.**

   ¿Cuánto le toma al usuario típico de una comunidad aprender la manera en como se usan los comandos relevantes a un conjunto de tareas? Se refiere a que tan rápido el usuario va a aprender a usar un sistema con el cual no había tenido contacto previamente. Este punto se refiere a la consecución de tareas básicas por parte de un usuario novato.

2. **Velocidad de desempeño.**

   ¿Cuánto le toma a un usuario completar un grupo de tareas específicas (benchmark tasks)? Una vez que el usuario ha aprendido a utilizar el sistema, se va a ponderar el lograr la velocidad con que puede completar una tarea específica.

3. **Tasas de error por parte de los usuarios.**

   ¿Cuántos y qué errores hace la gente al ejecutar un grupo de tareas específicas? Este apartado apunta hacia los errores cometidos por el usuario. Este atributo se refiere a aquellos errores que comete el usuario al utilizar el sistema. Una aplicación ideal evitaría que el usuario cometiera errores y funcionaría de manera óptima a cualquier petición por parte del usuario. En la práctica esto difícilmente se logra. Es vital que una vez que se produzca un error el sistema se lo haga saber rápida y claramente a los usuarios, le advierta sobre la severidad del mismo y le provea de algún mecanismo para recuperarse de ese error.

4. **Retención sobre el tiempo.**

   ¿Qué tan bien recuerdan los usuarios la manera en como funciona el sistema después de una hora, un día o una semana? Cuando un usuario ha utilizado un sistema tiempo atrás, y tiene la necesidad de utilizarlo de nuevo la curva de aprendizaje debe de ser significativamente menor que el caso del usuario que nunca haya utilizado dicho sistema. Esto es de primordial importancia para aplicaciones usadas intermitentemente.

5. **Satisfacción subjetiva.**

   ¿Qué tanto le gustaron a los usuarios los distintos atributos del sistema? Este atributo se refiere a la impresión subjetiva del usuario respecto al sistema.

John Cato [18] sugiere además los siguientes atributos:

- **Control.**

  Los usuarios deben de sentir que tienen el control por sobre la aplicación, y no al revés.

- **Habilidades.**

  Los usuarios deben de sentir que el sistema apoya, complementa y realza sus habilidades y experiencia - el sistema tiene respeto por el usuario.

- **Privacidad.**

  El sistema ayuda a los usuarios a proteger su información o la de sus clientes. Es muy importante señalar que los atributos antes mencionados van a tener una ponderación acorde a la actividad que se quiera realizar con un sistema. Algunos sistemas darán una mayor importancia a ciertos atributos por sobre algunos otros. Todo dependerá de las características de la audiencia objetivo y de las circunstancias en las cuales se usará la aplicación.

7.3.2.- Ingeniería de Usabilidad

La Ingeniería de Usabilidad (IU) es un área de HCI (*Human-Computer Interaction*, Interacción Humano-Computador) que da pautas para obtener productos con un alto grado de usabilidad, esto mediante la aplicación de distintos métodos en diferentes etapas del proceso de diseño y desarrollo de una manera estructurada y sistemática. El objetivo principal de la IU es mejorar la interfaz de usuario.

En las primeras etapas de desarrollo del producto, la evaluación de usabilidad será utilizada para decidir sobre distintos diseños de interfaces de usuario y finalmente decidir cuál será en el que se trabajará. En las siguientes etapas, las evaluaciones de usabilidad ayudarán a verificar que el sistema cumpla con los requerimientos iníciales.

En IU se trata de decidir que atributos del concepto de usabilidad deben de ser priorizados, con el fin de lograr metas verificables y medibles de niveles de usabilidad. Por ejemplo,

- Medir el desempeño de un usuario ejecutando una serie de tareas específicas con respecto al tiempo de terminación de las tareas o en base al número de errores cometidos.
- Determinar los niveles de preferencia subjetiva o el grado de satisfacción.
- La facilidad de aprendizaje podría medirse en base al número de tareas completadas en cierto periodo, número de errores cometidos, o respecto al número de veces que utilizó la opción de ayuda.

7.3.3.- Evaluación de usabilidad

La principal actividad en el proceso de usabilidad es la evaluación [16]. La evaluación de la usabilidad puede ayudar a determinar cuál es el nivel actual de la aplicación y si de hecho el diseño elegido realmente funciona. Los datos que se recaban mediante la observación del usuario frente a la aplicación y ver su desempeño, es información muy valiosa que ayudan en definitiva a detectar posibles falencias del sistema.

Existen diferentes técnicas para evaluar un sistema. Su uso depende de variables tales como costo, disponibilidad de tiempo, personal calificado para interpretar los datos, entre otros factores.

A continuación se describen brevemente algunos de estos métodos:

1. **Inspección formal de usabilidad.**

   Un grupo de expertos realizan una especie de juicio de la interfaz, con uno de los participantes actuando como moderador, destacando las fortalezas y las debilidades de la aplicación.

2. **Testeo de usabilidad (*Usability testing*).**

   Se realizan pruebas de desempeño de un grupo de usuarios utilizando el sistema a probar y se graban los resultados para un análisis posterior. Esta actividad se puede desarrollar en un laboratorio con condiciones controladas o directamente en el lugar donde se va a utilizar el sistema.

3. **Pensar en voz alta (*Thinking aloud*).**

   Se le pide al usuario que realice una serie de tareas específicas. El usuario debe de expresar sus acciones oralmente. Dentro de las instrucciones dadas al usuario de prueba no se le pide que *explique* sus acciones, simplemente que cada paso que realice lo diga en voz alta (generalmente el mismo usuario da una serie de explicaciones sin pedírselo de manera explícita).

4. **Evaluación heurística y de estándares.**

   En el área de Interfaces de Usuario existen una serie de estándares y de heurísticas ampliamente aceptadas (y probadas). En este tipo de evaluación un equipo de especialistas en usabilidad realiza una revisión conforme a estas normativas.

5. **Caminata cognitiva.**

   Un grupo de expertos simula la manera en como un usuario *caminaría* por la interfaz al enfrentarse a tareas particulares.

Una manera bastante efectiva, y económica, de recabar información sobre los usuarios es mediante la aplicación de encuestas. Las encuestas deben de ser escritas y revisadas por un panel de especialistas para asegurarse de que se van a evaluar factores críticos de la interfaz. Las encuestas on-line pueden ser colocadas en un sitio web ya en existencia, enviada por correo directamente a un grupo de usuarios, enviada a listas de correos o colocada en grupos de noticias.

8.- Clases y Módulos Desarrollados

Estas son las clases y módulos desarrollados para este proyecto, esta información tiene como propósito mostrar la codificación del sistema de manera interna, también se puede apreciar externamente mediante las interfaces graficas de usuario que se encuentran en el ANEXO E.

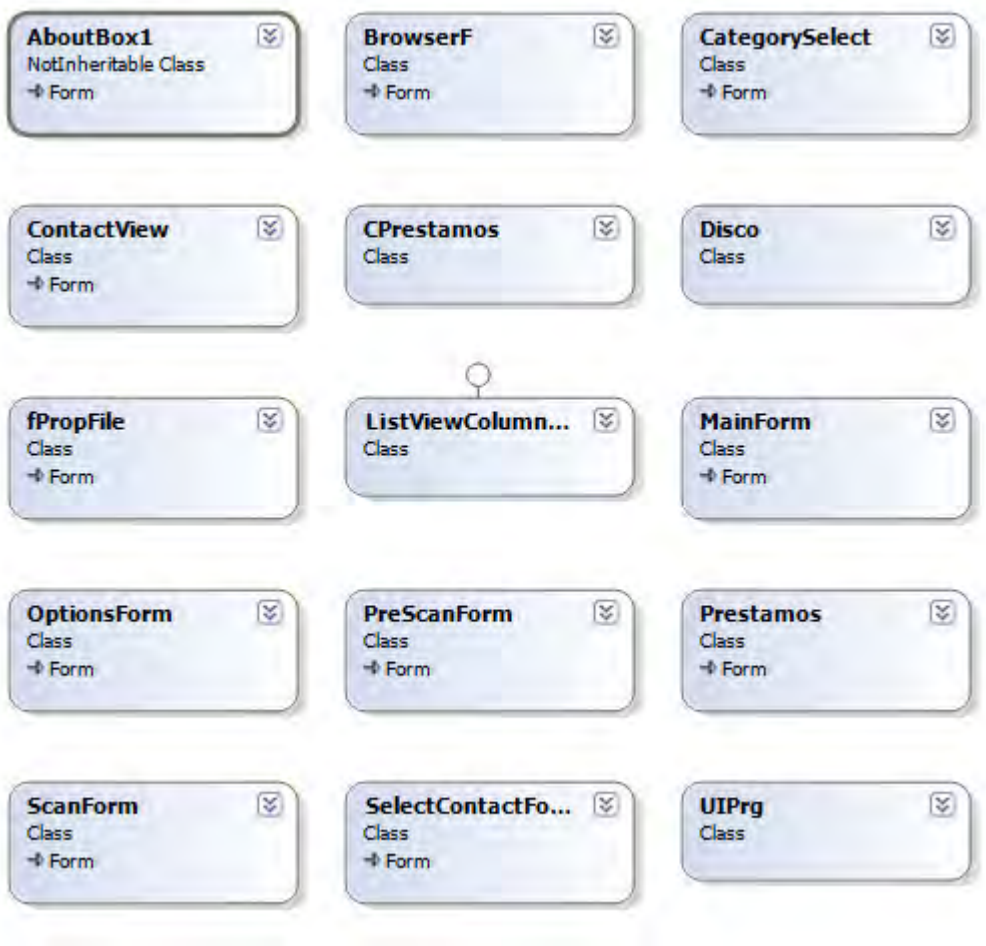


Ilustración 8.1.- Clases de implementación parte 1

El formulario AboutBox1 es el que despliega la información sobre la aplicación, BrowserF es el encargado de desplegar la ayuda la cual está hecha en formato html, CategorySelect permite seleccionar categorías para distintos propósitos, ContactView es el formulario que muestra las propiedades de los contactos, este también tiene incorporado un formulario del tipo Prestamos, en el cual se ve el historial de los prestamos de un contacto.

CPrestamos es una clase que contiene la lógica de los préstamos, la clase Disco representa a los discos reales, fPropFile es el formulario que obtiene y muestra los datos de los archivos en él, dependiendo del archivo también se muestran las propiedades de la interface IPlug.

El ListViewColumnSorter es una clase que extiende la funcionalidad de un ListView Normal de ".NET" dándole la capacidad de ordenarse por columnas haciendo comparaciones personalizadas según un tipo de dato.

OptionsForm es el formulario que maneja las opciones de la aplicación, manteniendo las preferencias de los usuarios entre usos de la aplicación y si es el caso puede recuperar las opciones por defecto de la aplicación. PreScanForm es el encargado de obtener la información previa a la lectura de un disco la cual consiste en la categoría y la descripción del nuevo disco a leer.

SelectContactForm es el formulario encargado de buscar y seleccionar contactos para distintos propósitos, UIProg es una clase que sirve para mostrar y mantener actualizado el progreso de lectura cuando se está extrayendo la información de los discos a través del BackgroundWorker.

Ilustración 8.2.-Clases de implementación parte 2

ContactAdd es el formulario que recopila la información de los nuevos contactos y los agrega, ContactSelect permite seleccionar contactos para diversos propósitos, editDes es el encargado de modificar la descripción de un disco.

fAddEstantes se encarga de crear y especificar las características de los estantes cuando se crean. fPropDisc es el formulario que obtiene y muestra los datos de los discos. La clase RPBusqueda sirve para reportar el progreso de las búsquedas de archivos y discos.

PrestarForm es el formulario encargado de asociar un disco a un contacto y crear un préstamo, para esto también hace uso de los formularios SelectDiscForm y SelectContactForm.

El VisorImp es la clase que genera y despliega la vista previa de impresión de los contenidos de la base de datos, ya sean discos o contactos, para luego imprimirlos.

La interface IPlug es la que define el comportamiento de los plugins de formato del sistema, está formada de dos formularios los cuales sirven para configurar y mostrar las propiedades avanzadas de los archivos que manejan, y un método llamado ReadMetadata al cual se le pasa la ruta del archivo para que sea el plugin el encargado de extraer la información extra de los archivos.

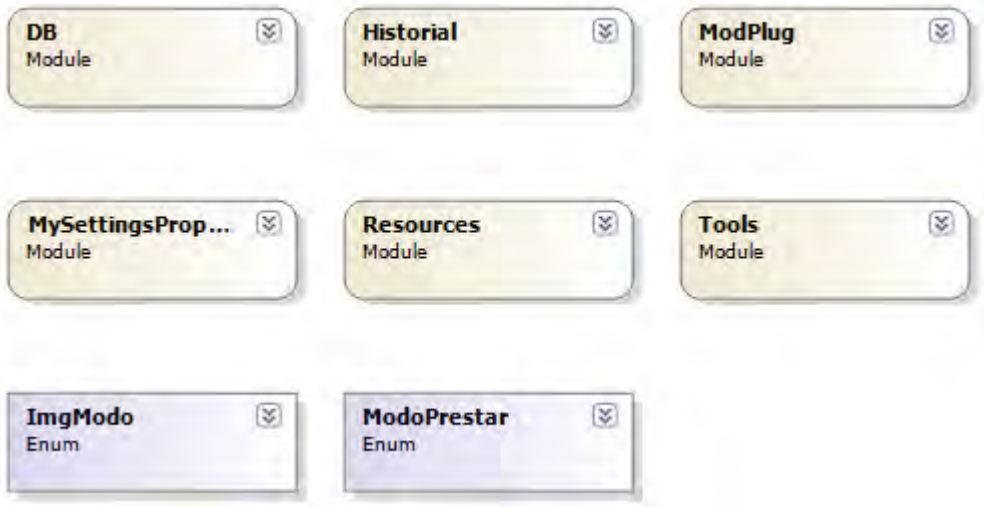


Ilustración 8.3.- Módulos y Enumeraciones de implementación.

El modulo Historial es el que maneja los cambios y la navegación por los discos en el modo de explorador, también se encarga de mantener los botones Atrás, Adelante y Arriba.

ModPlug es el modulo que administra y mantiene los plugins que el sistema detecta al funcionar, se encarga de mantener una lista de estos y llamar al correspondiente según sea necesario.

Resources es un modulo creado automáticamente que se encarga de manejar todos los recursos de la aplicación por ejemplo los iconos de los botones las barras, cadenas de textos, imágenes sonidos, etc.…

Tools es un modulo creado para asistir en tareas que son necesarias en las otras clases algunas tareas tienen que ver por ejemplo: con el manejo de las fechas, el formateo de los datos, manejo de iconos y obtención de información del hardware.

La enumeración ImgModo sirve para manejar el estado de la imagen asociada a un contacto.

La enumeración ModoPrestar se encarga de mantener la coherencia entre los distintos modos en que se puede instanciar el formulario de PrestarForm y reutilizar así su interfaz para distintos propósitos.

8.1.- Detalle de Clases y Módulos Principales

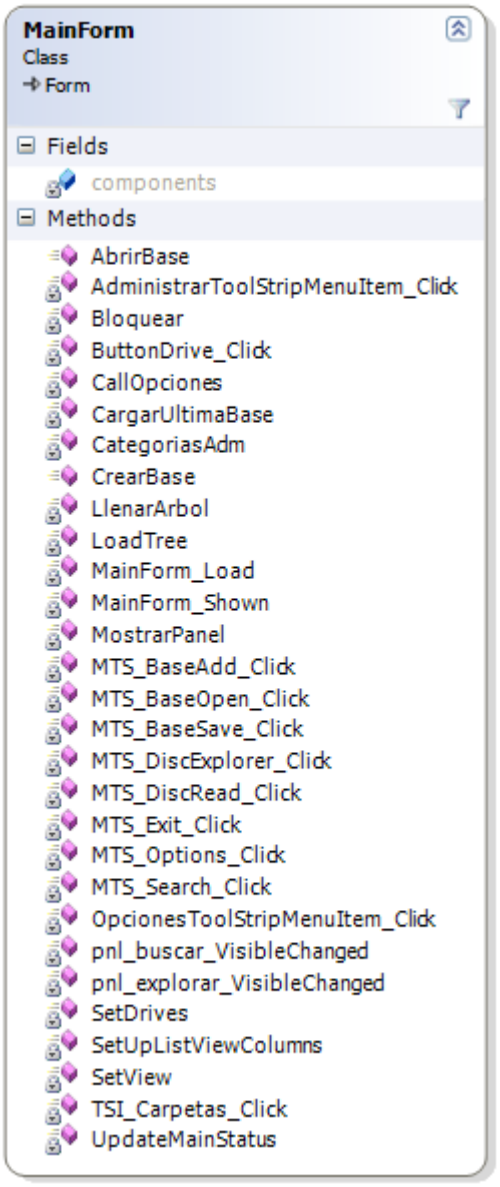


Ilustración 8.4 .- Clase MainForm

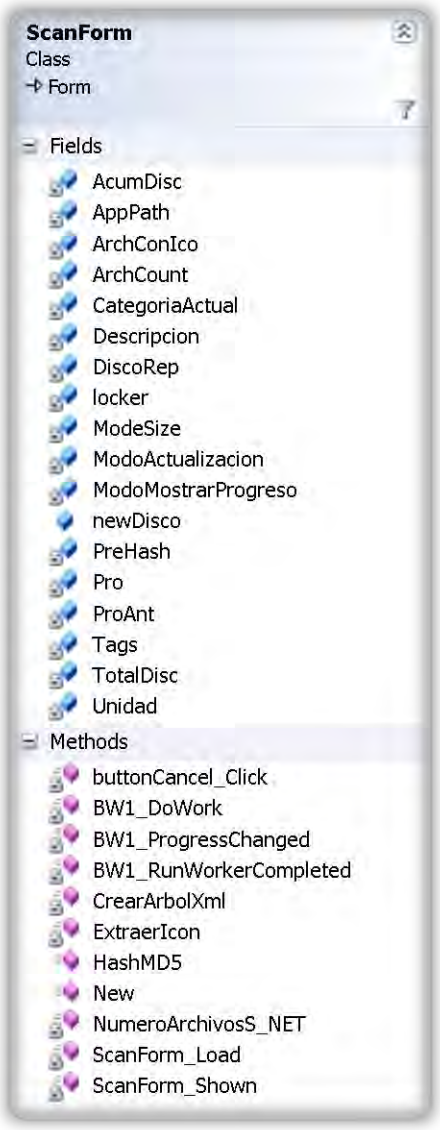


Ilustración 8.5.- Clase ScanForm

Estas son las clases del modulo principal y la de lectura de discos, en la principal se aprecian importantes métodos como SetDrives el cual es el encargado de detectar y configurar las unidades del sistema para su posterior lectura, en general este modulo tiene que ver con la interfaz de usuario y como interactúa este con el sistema a través de él.

La clase ScanForm es la encargada de extraer los datos de los discos, para ello se utiliza un **BackgroundWorker** [19] el cual permite al formulario ejecutar la extracción de datos y la creación del árbol XML de forma asincrónica. El método HashMD5 es el encargado de calcular el valor hash correspondiente a cada disco para efectos de comparación y búsqueda de discos duplicados, ExtraerIcon utiliza el módulo Tools para extraer los iconos de los tipos de archivo ejecutables, para así conservar la misma imagen original que el usuario asocia a estos.

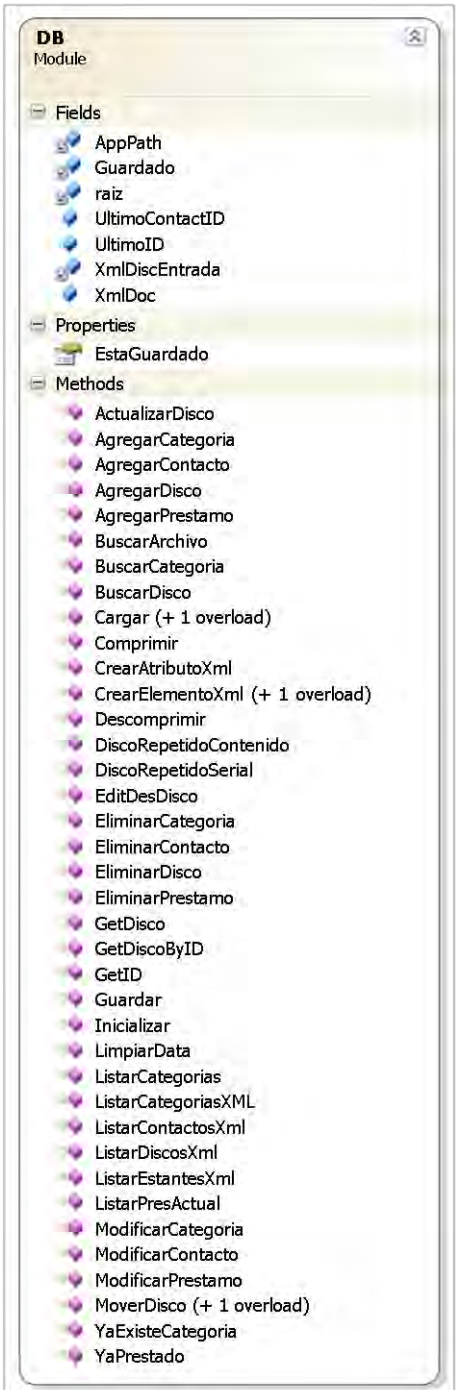


Ilustración 8.8.- Módulo DB

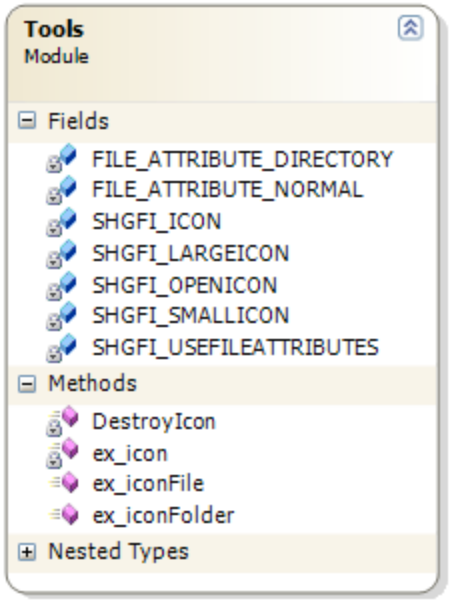


Ilustración 8.6.- Módulo Tools

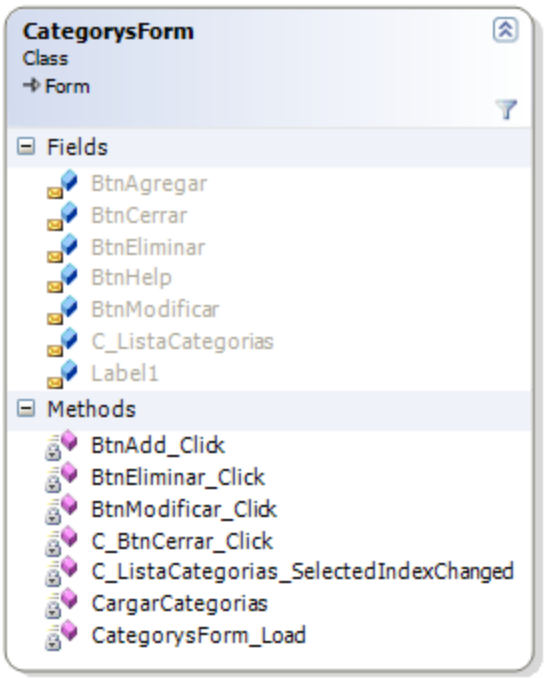


Ilustración 8.7.- Clase CategoryForm

El módulo DB es uno de los más importantes ya que está encargado de todas las funciones que tienen que ver con la base, es el “Parser” entre el sistema y los datos.

El módulo Tools es el encargado de suplir al sistema de todas las funciones de utilería necesarias para llevar a cabo la lógica de la aplicación, por ahora se puede ver que cumple funciones con respecto a los iconos, esto será ampliado a medida que se implemente el sistema.

La clase CategoryForm es del tipo UI la cual se encarga de mediar entre el usuario y el DB el cual se encarga realmente de agregar, modificar y eliminar las categorías.

9.- Conclusiones

En primer lugar, gracias a los estudios realizados en el estado del arte se logra entender los conceptos centrales en torno a la catalogación en general, conceptos de los cuales cabe destacar el orden y respeto por los estándares establecidos, para la mejor y más rápida comunicación entre las entidades, la simplicidad para enfrentar los problemas y ante todo la cooperación y mantención de la información por parte de los usuarios, sin la cual ningún sistema de catalogación funcionaría.

Además se puede decir que es mucho más influyente el estándar del Dublín Core el cual está hecho en estos tiempos y tiene muy buena documentación acerca de la implementación con las tecnologías asociadas a XML.

El estándar MARC se uso para sacar ideas sobre cómo clasificar y de qué manera codificar la información que se va a manejar de los discos y archivos, pero el uso de códigos de sub-campo y otros no va con la tendencia actual, ya que este proyecto está pensado en manejar las bases internamente como archivos XML, el formato MARC contradice algunos principios básicos del XML, el cual debe ser legible humanamente, independiente y libre.

Respecto a los temas de tecnología como el uso de XML y .NET se cumple con las premisas de respetar estándares y contribuir a la mejor comunicación, también hay que señalar que estas tecnologías son muy utilizadas actualmente y no solo por seguir una moda sino porque realmente permiten desarrollar soluciones de una manera más rápida y eficiente, en este caso el tiempo no era un recurso infinito, pero gracias al uso de estas tecnologías se logró cumplir con los plazos y llegar a buen término de este proyecto, los logros, la madurez y la buena documentación de XML y .NET se deben al trabajo de muchas personas e instituciones que están detrás de estas tecnologías en el caso de XML es la W3C y de .NET es Microsoft.

Para el manejo de los datos se escogió XML ya que su uso se hace cada vez más popular, debido a sus ventajas ya antes nombradas, a la vez que tiene un muy buen soporte por parte del Framework .NET, en el cual hay librerías muy completas y maduras especialmente diseñadas para el manejo de esta tecnología. Además como los datos que va a manejar el sistema son estructuras de directorios y archivos, calza de manera ideal con el sentido jerárquico de los documentos XML, en los cuales fácilmente se puede representar este tipo de estructuras.

También se puede decir que siempre se está en constante aprendizaje, con respecto a la tecnología XPath se tuvo que aprender una nueva forma de trabajar con grandes cantidades de datos, pero de una manera totalmente diferente a la acostumbrada con SQL, en XML los datos están almacenados de manera jerárquica lo cual implica que todas las consultas realizadas mediante XPath son como expresiones regulares que navegan hasta los datos para obtener resultados, a primera vista parecen menos eficientes que las tablas, pero a la larga se saca mas partido debido a la relación de jerarquía que hay entre la información y la manera en que se almacena, sin decir que la compatibilidad es una de sus mayores potencialidades.

Gracias a la metodología utilizada, se pudo considerar la realización de algunas modificaciones, correspondiente a las iteraciones que van surgiendo a lo largo del proyecto, lo cual facilita el cumplimiento de los plazos estipulados al comienzo del Proyecto.

Finalmente cabe destacar que los objetivos planteados al comienzo de este sistema, han sido cumplidos, destacando como un punto de mucha importancia, que este proyecto, por estar enfocado a la realización de un sistema genérico, puede aceptar nuevos requerimientos o cambios en el camino. Estos nuevos requerimientos pueden ser abordados con facilidad gracias al paradigma, metodología y herramienta escogida, lo cual, permite conseguir un sistema que sea capaz de cumplir con los requerimientos a cabalidad.

Así como se han logrado los objetivos propuestos, también se puede ampliar y mejorar las funciones del sistema, agregar más plugins para los metadatos y seguir mejorando de manera continua en el tiempo el sistema, para que se mantenga vigente en el tiempo.

10.- Referencias

En orden de aparición:

[1]. **Miniwatts Marketing Group.** Internet World Stats. [En línea] 1 de 03 de 2007. http://www.internetworldstats.com/stats.htm.

[2]. **Sánchez, Beatriz.** Computerworld. [En línea] 12 de 1 de 2007. http://www.idg.es/computerworld/articulo.asp?id=299362422.

[3]. **Senso., Eva Méndez y José A.** SEDIC. *Asociación Española de Documentación e Información.* [En línea] 2004. http://www.sedic.es/autoformacion/metadatos/tema7.htm.

[4]. **Die Deutsche Bibliothek.** The German National Library. [En línea] 2006. http://www.ddb.de/eng/standardisierung/formate/formatumstieg_herst.htm.

[5]. **Library of Congress.** The Library of Congress. [En línea] 2007. http://www.loc.gov/index.html.

[6]. **DCMI .** Using Dublin Core - The Elements. [En línea] 26 de 08 de 2003. http://es.dublincore.org/documents/usageguide/elements.shtml.

[7]. **Board, DCMI Usage.** DCMI Metadata Terms. [En línea] 20 de 12 de 2004. http://dublincore.org/documents/dcmi-terms/.

[8]. **DCMI.** Traducciones de los Documentos de la DCMI. [En línea] 2007. http://es.dublincore.org/resources/translations/.

[9]. **Rachel Heery, Manjula Patel.** Application profiles: mixing and matching metadata schemas. [En línea] Ariadne Issue 25, 24 de 9 de 2000 . http://www.ariadne.ac.uk/issue25/app-profiles/.

[10]. **Wikimedia Foundation, Inc.** Wikipedia. [En línea] 14 de Abril de 2007. http://es.wikipedia.org/wiki/XML.

[11]. **ComponentAce.** Zlib for .NET. [En línea] 2007. http://componentace.com/ZLIB.NET.

[12]. **Adler, Mark.** zLib. [En línea] 2005. http://www.zlib.net/.

[13]. **Roshal, Alexander.** RAR Lab. [En línea] http://www.rarlab.com/rar_add.htm.

[14]. **Pressman, Roger S.** *Ingenieria de Software.* 2002.

[15]. **Reyes, Lic. Giovanni Cuadra.** elGuille. [En línea] 7 de junio de 2005. http://www.elguille.info/colabora/NET2005/gcuadra_VBSQL2005.htm.

[16]. **Xavier Ferré, Natalia Juristo, Helmut Windl, Larry Constantine. ,.** Usability basics for software developers. s.l. : IEEE Software, January/February 2001. p. 22-29.

[17]. **Shneiderman, Ben.** *Designing the user interface.* s.l. : Reading, MA: Addison-Wesley, 1998.

[18]. **Cato, John.** *User-centered web design.* Harlow, England : Addison-Wesley, 2001.

[19]. **Microsoft.** MSDN. [En línea] 2007. http://msdn2.microsoft.com/es-es/library/c8dcext2(VS.80).aspx.

11.- Glosario

En orden alfabético.

Base: Esto se refiere al conjunto de datos que el sistema almacena con respecto a todos los objetos que maneja, en si el formato es un documento XML.

Bugs: Un defecto de software (computer bug en inglés), es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora (software).

Categoría: Es un concepto por el cual se agrupa un conjunto de discos.

CD: El disco compacto (en inglés Compact Disc) es un soporte digital óptico utilizado para almacenar cualquier tipo de información ya sea audio, vídeo, documentos y otros datos.

Contacto: Esto representa el conjunto de datos que identifica a una persona únicamente y contiene la información necesaria para contactar o ubicar a dicha persona.

Disco: Son los medios generalmente DVDs o CDs que se usan para almacenar archivos de forma más duradera.

Dublin Core: Es un sistema de 15 definiciones semánticas descriptivas que pretenden transmitir un significado semántico a las mismas, creado por la iniciativa de Metadatos de Dublin Core (DCMI).

DVD: "Digital Versatile Disc" o "Disco Versátil Digital", es un formato de almacenamiento óptico que puede ser usado para guardar datos, incluyendo películas con alta calidad de vídeo y audio. Se asemeja a los discos compactos en cuanto a sus dimensiones físicas (diámetro de 12 u 8 cm.), pero está codificado en un formato distinto y a una densidad mucho mayor.

Estante: Esto es la representación en el sistema de un contenedor de discos en el cual se pueden guardar cierta cantidad de ellos en cierto número de filas definidas al momento de su creación.

Etiqueta: Es una palabra o varias que sirven para clasificar y especificar el contenido de los discos, su utilidad se ve reflejada al momento de buscar y restringir ese conjunto de búsqueda.

MARC: Machine Readable Catalog, catalogo legible por maquina. Surgió y fue desarrollado por EEUU y Canadá, con el objeto de solucionar los graves inconvenientes existentes con respecto al intercambio de Registros Catalográficos entre bibliotecas de todo el mundo. Debido al desarrollo individual y uso de diferentes formatos que resultan incompatibles entre sí.

.NET: es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Plugin: (o plug-in -en inglés "enchufar"-, también conocido como addin, add-in, addon o add-on) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica. Ésta aplicación adicional es ejecutada por la aplicación principal.

Ubicación: Es la representación virtual de una ubicación física de un disco.

W3C: El World Wide Web Consortium, es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

XML: eXtensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

XPath: Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.

XQuery: Es un lenguaje de consultas diseñado para consultar colecciones de datos XML, semánticamente es similar a SQL.

12.- ANEXOS

ANEXOS

A. ANEXO A : Esquema XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="Database">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Categories"/>
    <xs:element ref="Contacts"/>
    <xs:element ref="ModelLocation"/>
   </xs:sequence>
   <xs:attribute name="CreationTime" use="required"/>
   <xs:attribute name="LastWriteTime" use="required"/>
   <xs:attribute name="Version" use="required"/>
   <xs:attribute name="LastID" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ModelLocation">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Shelf" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="Name" use="required"/>
      <xs:attribute name="Rows" use="required"/>
      <xs:attribute name="DiscsByRow" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="Contacts">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Contact" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:all>
       <xs:element name="LastName" type="xs:string" minOccurs="0"/>
       <xs:element name="Title" type="xs:string" minOccurs="0"/>
       <xs:element name="NickName" type="xs:string" minOccurs="0"/>
       <xs:element name="Phone" minOccurs="0"/>
       <xs:element name="Movil" minOccurs="0"/>
       <xs:element name="Fax" minOccurs="0"/>
       <xs:element name="Address" minOccurs="0"/>
       <xs:element name="Province" minOccurs="0"/>
       <xs:element name="FirstName" type="xs:string"/>
       <xs:element name="City" minOccurs="0"/>
       <xs:element name="Country" minOccurs="0"/>
```

```
      <xs:element name="PostalCode" minOccurs="0"/>
      <xs:element name="E-mails" minOccurs="0">
       <xs:complexType/>
      </xs:element>
      <xs:element ref="Loans" minOccurs="0"/>
     </xs:all>
     <xs:attribute name="ContactID" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Categories">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Category" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Disc">
 <xs:complexType>
  <xs:sequence>
   <xs:sequence minOccurs="0">
    <xs:element name="Directory" type="DirectoryType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="File" type="FileType" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:element name="Tags" minOccurs="0">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Tag" type="xs:string" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="Description" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="ID" use="required"/>
  <xs:attribute name="Serial" use="required"/>
  <xs:attribute name="FileSystem" use="required"/>
  <xs:attribute name="VolumeLabel" type="xs:string" use="required"/>
  <xs:attribute name="Hash" use="required"/>
  <xs:attribute name="TotalFiles" use="required"/>
  <xs:attribute name="TotalSize" use="required"/>
 </xs:complexType>
</xs:element>
<xs:element name="Loans">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Loan" maxOccurs="unbounded">
    <xs:complexType>
```

```
      <xs:attribute name="DiscID" type="xs:int" use="required"/>
      <xs:attribute name="LoanDateTime" type="xs:dateTime" use="required"/>
      <xs:attribute name="ReturnDateTime" type="xs:dateTime" use="required"/>
      <xs:attribute name="Description" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="DirectoryType">
  <xs:attribute name="Name" use="required"/>
 </xs:complexType>
 <xs:complexType name="FileType">
  <xs:sequence>
   <xs:element name="CustomData" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="PropertyName" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
  <xs:attribute name="Extension" use="required"/>
  <xs:attribute name="Size" use="required"/>
  <xs:attribute name="CreationTime" use="required"/>
  <xs:attribute name="LastWriteTime" use="required"/>
  <xs:attribute name="LastAccessTime" use="required"/>
  <xs:attribute name="Attributes" use="required"/>
  <xs:attribute name="IcoId"/>
 </xs:complexType>
 <xs:complexType name="CategoryType">
  <xs:sequence>
   <xs:element ref="Disc" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Name" use="required"/>
 </xs:complexType>
 <xs:element name="Category" type="CategoryType"/>
</xs:schema>
```

B. ANEXO B : Ejemplo de una Base de Datos del sistema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Base xml para el prototipo-->
<Database CreationTime="09-11-2007 0:13:05" LastWriteTime="18-11-2007 18:35:45" Version="0.1" LastID="5">
 <Categories>
  <Category Name="default">
   <Disc ID="1" Serial="AA51A018" FileSystem="UDF" VolumeLabel="New"
Hash="7be3d397a326fe71a5148391118af61e" TotalFiles="187" TotalSize="4611769108">
    <Description>mesclados</Description>
    <Directory Name="Emulacion Cps3">
     <Directory Name="Mame version">
      <File Name="Mame v0.86u3 Cps3 Chd'S(Jojo,Jojoba,Sfiii,Sfiii2,Sfiii3,Warzard)" Extension="rar" Size="285601713"
CreationTime="128291332010000000" LastWriteTime="128291332010000000"
LastAccessTime="128291332010000000" Attributes="R">
       <CustomData TotalSize="283994287">
        <Entry Name="MAME v0.86u3 CPS3 CHD's\jojo" Size="0" DateTime="824791920" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\jojoba" Size="0" DateTime="824791915" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii" Size="0" DateTime="824791910" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii2" Size="0" DateTime="824791907" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii3" Size="0" DateTime="824791894" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\warzard" Size="0" DateTime="824791891" Attributes="16"
PackSize="0" CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's" Size="0" DateTime="824791891" Attributes="16" PackSize="0"
CRC="0"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\jojo\jojo.chd" Size="45110952" DateTime="824786133" Attributes="32"
PackSize="44567155" CRC="-1874096655"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\jojoba\jojoba.chd" Size="52907030" DateTime="824786155"
Attributes="32" PackSize="52548529" CRC="1652250115"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii\sfiii.chd" Size="37715715" DateTime="824786170" Attributes="32"
PackSize="37716159" CRC="-1179872291"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii2\sfiii2.chd" Size="43246653" DateTime="824786189"
Attributes="32" PackSize="43229870" CRC="-1681389619"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\sfiii3\sfiii3.chd" Size="67850966" DateTime="824786213"
Attributes="32" PackSize="67599998" CRC="1519673800"/>
        <Entry Name="MAME v0.86u3 CPS3 CHD's\warzard\warzard.chd" Size="37162971" DateTime="824786223"
Attributes="32" PackSize="37135151" CRC="287626291"/>
       </CustomData>
      </File>
     </Directory>
     <Directory Name="Cps3 New Emulator">
      <Directory Name="CFG">
       <File Name="jojo" Extension="input" Size="112" CreationTime="128293433530000000"
LastWriteTime="128293433530000000" LastAccessTime="128293433530000000" Attributes="R"/>
       <File Name="sfiii2" Extension="input" Size="112" CreationTime="128307178310000000"
LastWriteTime="128307178310000000" LastAccessTime="128307178310000000" Attributes="R"/>
       <File Name="sfiii3" Extension="input" Size="112" CreationTime="128295213560000000"
LastWriteTime="128295213560000000" LastAccessTime="128295213560000000" Attributes="R"/>
       <File Name="warzard" Extension="input" Size="112" CreationTime="128291135040000000"
LastWriteTime="128291135040000000" LastAccessTime="128291135040000000" Attributes="R"/>
      </Directory>
      <Directory Name="NVDATA">
```

```
      <File Name="jojo" Extension="DAT" Size="353" CreationTime="128293434550000000"
LastWriteTime="128293434550000000" LastAccessTime="128293434550000000" Attributes="R"/>
      <File Name="jojoba" Extension="DAT" Size="527" CreationTime="128293423600000000"
LastWriteTime="128293423600000000" LastAccessTime="128293423600000000" Attributes="R"/>
      <File Name="sfiii" Extension="DAT" Size="357" CreationTime="128294193130000000"
LastWriteTime="128294193130000000" LastAccessTime="128294193130000000" Attributes="R"/>
      <File Name="sfiii2" Extension="DAT" Size="531" CreationTime="128307195760000000"
LastWriteTime="128307195760000000" LastAccessTime="128307195760000000" Attributes="R"/>
      <File Name="sfiii3" Extension="DAT" Size="358" CreationTime="128299869910000000"
LastWriteTime="128299869910000000" LastAccessTime="128299869910000000" Attributes="R"/>
      <File Name="warzard" Extension="DAT" Size="353" CreationTime="128293437670000000"
LastWriteTime="128293437670000000" LastAccessTime="128293437670000000" Attributes="R"/>
    </Directory>
    <Directory Name="Plugins">
      <File Name="BILINEAR" Extension="DLL" Size="4608" CreationTime="126655340720000000"
LastWriteTime="126655340720000000" LastAccessTime="126655340720000000" Attributes="R"/>
      <File Name="bilinearlight" Extension="dll" Size="4608" CreationTime="126655653620000000"
LastWriteTime="126655653620000000" LastAccessTime="126655653620000000" Attributes="R"/>
      <File Name="bilinearplus" Extension="dll" Size="4608" CreationTime="126654642360000000"
LastWriteTime="126654642360000000" LastAccessTime="126654642360000000" Attributes="R"/>
      <File Name="genericscan" Extension="dll" Size="8704" CreationTime="126633730620000000"
LastWriteTime="126633730620000000" LastAccessTime="126633730620000000" Attributes="R"/>
      <File Name="interlaced" Extension="dll" Size="4096" CreationTime="126633730620000000"
LastWriteTime="126633730620000000" LastAccessTime="126633730620000000" Attributes="R"/>
      <File Name="Plugins" Extension="cfg" Size="37" CreationTime="128284153680000000"
LastWriteTime="128284153680000000" LastAccessTime="128284153680000000" Attributes="R"/>
      <File Name="PLUGSRC" Extension="ZIP" Size="7260" CreationTime="126647904560000000"
LastWriteTime="126647904560000000" LastAccessTime="126647904560000000" Attributes="R"/>
      <File Name="scale2x" Extension="dll" Size="4096" CreationTime="126860254160000000"
LastWriteTime="126860254160000000" LastAccessTime="126860254160000000" Attributes="R"/>
      <File Name="scale2x75" Extension="dll" Size="4096" CreationTime="126860254200000000"
LastWriteTime="126860254200000000" LastAccessTime="126860254200000000" Attributes="R"/>
      <File Name="scanclear" Extension="dll" Size="3584" CreationTime="126633730580000000"
LastWriteTime="126633730580000000" LastAccessTime="126633730580000000" Attributes="R"/>
      <File Name="scanclear50" Extension="dll" Size="4096" CreationTime="126633730600000000"
LastWriteTime="126633730600000000" LastAccessTime="126633730600000000" Attributes="R"/>
      <File Name="superscalesrc" Extension="zip" Size="3706" CreationTime="126652574900000000"
LastWriteTime="126652574900000000" LastAccessTime="126652574900000000" Attributes="R"/>
    </Directory>
    <Directory Name="Roms">
      <File Name="jojo" Extension="zip" Size="42956732" CreationTime="128268858320000000"
LastWriteTime="128268858320000000" LastAccessTime="128268858320000000" Attributes="R"/>
      <File Name="jojoba" Extension="zip" Size="50138669" CreationTime="128268856450000000"
LastWriteTime="128268856450000000" LastAccessTime="128268856450000000" Attributes="R"/>
      <File Name="jojobaa" Extension="zip" Size="131760" CreationTime="128268828870000000"
LastWriteTime="128268828870000000" LastAccessTime="128268828870000000" Attributes="R"/>
      <File Name="sfiii" Extension="zip" Size="35910499" CreationTime="128268517180000000"
LastWriteTime="128268517180000000" LastAccessTime="128268517180000000" Attributes="R"/>
      <File Name="sfiii2" Extension="zip" Size="40235338" CreationTime="128268517250000000"
LastWriteTime="128268517250000000" LastAccessTime="128268517250000000" Attributes="R"/>
      <File Name="sfiii3" Extension="zip" Size="64713387" CreationTime="128268519340000000"
LastWriteTime="128268519340000000" LastAccessTime="128268519340000000" Attributes="R"/>
      <File Name="warzard" Extension="zip" Size="35416307" CreationTime="128268764310000000"
LastWriteTime="128268764310000000" LastAccessTime="128268764310000000" Attributes="R"/>
    </Directory>
    <File Name="cps3emulator" Extension="zip" Size="566691" CreationTime="128291133160000000"
LastWriteTime="128291133160000000" LastAccessTime="128291133160000000" Attributes="R"/>
    <File Name="emulator" Extension="exe" Size="950272" CreationTime="128290238760000000"
LastWriteTime="128290238760000000" LastAccessTime="128290238760000000" Attributes="R" IcoId="0"/>
```

```
    <File Name="emulator" Extension="ini" Size="828" CreationTime="128307195760000000"
LastWriteTime="128307195760000000" LastAccessTime="128307195760000000" Attributes="R"/>
    <File Name="readme_cps3" Extension="txt" Size="6187" CreationTime="128290240300000000"
LastWriteTime="128290240300000000" LastAccessTime="128290240300000000" Attributes="R"/>
   </Directory>
  </Directory>
  <Directory Name="Death Note">
   <File Name="dn making02" Extension="zip" Size="3628318" CreationTime="128302366440000000"
LastWriteTime="128302366440000000" LastAccessTime="128302366440000000" Attributes="R"/>
   <File Name="dn-eg" Extension="zip" Size="4029641" CreationTime="128302366680000000"
LastWriteTime="128302366680000000" LastAccessTime="128302366680000000" Attributes="R"/>
   <File Name="DN-EG1-LMN" Extension="ZIP" Size="1240766" CreationTime="128302366880000000"
LastWriteTime="128302366880000000" LastAccessTime="128302366880000000" Attributes="R"/>
   <File Name="DN-EG2-LMN" Extension="ZIP" Size="2273407" CreationTime="128302367070000000"
LastWriteTime="128302367070000000" LastAccessTime="128302367070000000" Attributes="R"/>
   <File Name="DN-EG3-LMN" Extension="ZIP" Size="4400360" CreationTime="128302367160000000"
LastWriteTime="128302367160000000" LastAccessTime="128302367160000000" Attributes="R"/>
   <File Name="DN-EG4-LMN1" Extension="ZIP" Size="723967" CreationTime="128302367280000000"
LastWriteTime="128302367280000000" LastAccessTime="128302367280000000" Attributes="R"/>
   <File Name="doa" Extension="zip" Size="4274085" CreationTime="128302367470000000"
LastWriteTime="128302367470000000" LastAccessTime="128302367470000000" Attributes="R"/>
   <File Name="how to read dn 3" Extension="zip" Size="2118290" CreationTime="128302367540000000"
LastWriteTime="128302367540000000" LastAccessTime="128302367540000000" Attributes="R"/>
   <File Name="how to read dn 4" Extension="zip" Size="2556315" CreationTime="128302367630000000"
LastWriteTime="128302367630000000" LastAccessTime="128302367630000000" Attributes="R"/>
   <File Name="Lent" Extension="zip" Size="10521302" CreationTime="128302367730000000"
LastWriteTime="128302367730000000" LastAccessTime="128302367730000000" Attributes="R"/>
   <File Name="moonwalk" Extension="zip" Size="3935349" CreationTime="128302367870000000"
LastWriteTime="128302367870000000" LastAccessTime="128302367870000000" Attributes="R"/>
   <File Name="nolimit" Extension="zip" Size="5735447" CreationTime="128302367960000000"
LastWriteTime="128302367960000000" LastAccessTime="128302367960000000" Attributes="R"/>
   <File Name="unfair" Extension="zip" Size="7112684" CreationTime="128302368100000000"
LastWriteTime="128302368100000000" LastAccessTime="128302368100000000" Attributes="R"/>
   <File Name="yomikiri01" Extension="zip" Size="900797" CreationTime="128302368170000000"
LastWriteTime="128302368170000000" LastAccessTime="128302368170000000" Attributes="R"/>
   <File Name="yomikiri02" Extension="zip" Size="7020438" CreationTime="128302368270000000"
LastWriteTime="128302368270000000" LastAccessTime="128302368270000000" Attributes="R"/>
   <File Name="yomikiri03" Extension="zip" Size="3695456" CreationTime="128302368350000000"
LastWriteTime="128302368350000000" LastAccessTime="128302368350000000" Attributes="R"/>
   <File Name="yomikiri04" Extension="zip" Size="3388972" CreationTime="128302368420000000"
LastWriteTime="128302368420000000" LastAccessTime="128302368420000000" Attributes="R"/>
   <File Name="yomikiri05" Extension="zip" Size="3991988" CreationTime="128302368500000000"
LastWriteTime="128302368500000000" LastAccessTime="128302368500000000" Attributes="R"/>
   <File Name="yomikiri06" Extension="zip" Size="3462457" CreationTime="128302368590000000"
LastWriteTime="128302368590000000" LastAccessTime="128302368590000000" Attributes="R"/>
   <File Name="yomikiri07" Extension="zip" Size="3719906" CreationTime="128302368660000000"
LastWriteTime="128302368660000000" LastAccessTime="128302368660000000" Attributes="R"/>
   <File Name="yomikiri08" Extension="zip" Size="6493452" CreationTime="128302368760000000"
LastWriteTime="128302368760000000" LastAccessTime="128302368760000000" Attributes="R"/>
   <File Name="yomikiri09" Extension="zip" Size="363553" CreationTime="128302368830000000"
LastWriteTime="128302368830000000" LastAccessTime="128302368830000000" Attributes="R"/>
   <File Name="yteg" Extension="zip" Size="1303787" CreationTime="128302368910000000"
LastWriteTime="128302368910000000" LastAccessTime="128302368910000000" Attributes="R"/>
   <File Name="Death Note - Another Note" Extension="zip" Size="20642387" CreationTime="128303760810000000"
LastWriteTime="128303760810000000" LastAccessTime="128303760810000000" Attributes="R"/>
   <File Name="dn 62.5" Extension="zip" Size="2135004" CreationTime="128302365880000000"
LastWriteTime="128302365880000000" LastAccessTime="128302365880000000" Attributes="R"/>
   <File Name="dn making01" Extension="zip" Size="3952869" CreationTime="128302366200000000"
LastWriteTime="128302366200000000" LastAccessTime="128302366200000000" Attributes="R"/>
```

```
  </Directory>
  <Directory Name="Progs">
   <Directory Name="SWF Decompiler">
    <File Name="Sothink.SWF.Decompiler.v3.7.70607.incl.keygen-FFF.by.ChingLiu" Extension="rar" Size="3504081"
CreationTime="128292961270000000" LastWriteTime="128292961270000000"
LastAccessTime="128292961270000000" Attributes="R">
     <CustomData TotalSize="3532277">
      <Entry Name="Keygen.exe" Size="184832" DateTime="910465103" Attributes="32" PackSize="182163" CRC="-
1630881282"/>
      <Entry Name="FILE_ID.DIZ" Size="427" DateTime="889985559" Attributes="32" PackSize="180" CRC="-
943545639"/>
      <Entry Name="Setup.exe" Size="3335137" DateTime="919099578" Attributes="32" PackSize="3317840"
CRC="1998793045"/>
      <Entry Name="FFF.NFO" Size="11881" DateTime="910465122" Attributes="32" PackSize="3277" CRC="-
370857012"/>
     </CustomData>
    </File>
   </Directory>
   <Directory Name="Strata 3d CX 5.0">
    <File Name="Strata 3D CX v5.0" Extension="bin" Size="465674832" CreationTime="128294241490000000"
LastWriteTime="128294241490000000" LastAccessTime="128294241490000000" Attributes="R"/>
   </Directory>
   <Directory Name="Mame Screen saver">
    <File Name="MameScreenSaverSetup" Extension="zip" Size="684464" CreationTime="128295189030000000"
LastWriteTime="128295189030000000" LastAccessTime="128295189030000000" Attributes="R"/>
   </Directory>
   <Directory Name="Launchy (buscador windows)">
    <File Name="LaunchySetup125" Extension="zip" Size="1060211" CreationTime="128301605390000000"
LastWriteTime="128301605390000000" LastAccessTime="128301605390000000" Attributes="R"/>
    <File Name="LSIncludedSkins100" Extension="zip" Size="161859" CreationTime="128301600850000000"
LastWriteTime="128301600850000000" LastAccessTime="128301600850000000" Attributes="R"/>
    <File Name="LSNewSkins" Extension="zip" Size="1972142" CreationTime="128301601050000000"
LastWriteTime="128301601050000000" LastAccessTime="128301601050000000" Attributes="R"/>
   </Directory>
   <Directory Name="Adobe Creative Suit CS3">
    <File Name="MONITO_ADOBE_CS3" Extension="mdf" Size="2293039104" CreationTime="128312886160000000"
LastWriteTime="128312886160000000" LastAccessTime="128312886160000000" Attributes="R"/>
    <File Name="MONITO_ADOBE_CS3" Extension="mds" Size="4314" CreationTime="128312886160000000"
LastWriteTime="128312886160000000" LastAccessTime="128312886160000000" Attributes="R"/>
   </Directory>
  </Directory>
  <Directory Name="Futurama-Simpsons">
   <File Name="Futurama-Simpsons_Infinitely_Secret_Crossover_Crisis_1_Spanish" Extension="cbr" Size="9787500"
CreationTime="128189860400000000" LastWriteTime="128189860400000000"
LastAccessTime="128189860400000000" Attributes="R"/>
   <File Name="Futurama-Simpsons_Infinitely_Secret_Crossover_Crisis_2_Spanish" Extension="cbr" Size="9407493"
CreationTime="128189792680000000" LastWriteTime="128189792680000000"
LastAccessTime="128189792680000000" Attributes="R"/>
   <File Name="Simpsons-Futurama Crossover Crisis Ii 1 Spanish" Extension="cbr" Size="10706755"
CreationTime="128189796000000000" LastWriteTime="128189796000000000"
LastAccessTime="128189796000000000" Attributes="R"/>
   <File Name="Simpsons-Futurama_Crossover_Crisis_II_2_Spanish" Extension="cbr" Size="10090749"
CreationTime="128189890600000000" LastWriteTime="128189890600000000"
LastAccessTime="128189890600000000" Attributes="R"/>
  </Directory>
  <Directory Name="Prog comics">
   <File Name="comic reader setup" Extension="zip" Size="1064365" CreationTime="128190540420000000"
LastWriteTime="128190540420000000" LastAccessTime="128190540420000000" Attributes="R"/>
  </Directory>
```

```
   <Directory Name="Clasic Project">
    <File Name="The Classic Project Megamix Vol 1" Extension="mp3" Size="61744848"
CreationTime="128309527510000000" LastWriteTime="128309527510000000"
LastAccessTime="128309527510000000" Attributes="R"/>
    <File Name="The Classic Project Megamix Vol 2" Extension="mp3" Size="80830553"
CreationTime="128309507220000000" LastWriteTime="128309507220000000"
LastAccessTime="128309507220000000" Attributes="R"/>
    <File Name="The Classic Project Megamix Vol 3" Extension="mp3" Size="96396288"
CreationTime="128309434890000000" LastWriteTime="128309434890000000"
LastAccessTime="128309434890000000" Attributes="R"/>
    <File Name="The Classic Project Megamix Vol 4" Extension="mp3" Size="102103928"
CreationTime="128309487330000000" LastWriteTime="128309487330000000"
LastAccessTime="128309487330000000" Attributes="R"/>
    <File Name="The Classic Project Megamix Vol 5" Extension="mp3" Size="94960450"
CreationTime="128309549330000000" LastWriteTime="128309549330000000"
LastAccessTime="128309549330000000" Attributes="R"/>
   </Directory>
  </Disc>
 </Category>
 <Category Name="Programas">
  </Category>
</Categories>
<Contacts>
 <Contact ContactID="1">
  <FirstName>Juan</FirstName>
  <LastName>Perez</LastName>
  <NickName>Juanito</NickName>
  <Address>pedro montt #666</Address>
  <City>valparaíso</City>
  <Country>Chile</Country>
  <Emails>juanp@gmail.com</Emails>
  <Fax>238776543</Fax>
  <Movil>098876544</Movil>
  <Phone>2345617</Phone>
  <PostalCode>12345</PostalCode>
  <Province>valparaíso</Province>
  <Title>sr</Title>
  <Loans>
   <Loan DiscoID="5" LoanDateTime="128398945665000000" ReturnDateTime="47966796000000000"
Description="Devuelvelo luego"/>
  </Loans>
 </Contact>
</Contacts>
   <ModelLocation>
    <Shelf Name="el Amarillo" DiscsByRow="60" Rows="4"/>
    <Shelf Name="el negro" DiscsByRow="100" Rows="3"/>
   </ModelLocation>
  </Database>
```

Este ejemplo es creado de forma manual para demostrar cómo se almacenarían los datos en un documento XML correspondiente a un par de discos y personas.

C. ANEXO C : Cabezeras de formato

Definición de la cabera de un archivo comprimido ZIP:

```
A.  Local file header:
      local file header signature     4 bytes  (0x04034b50)
      version needed to extract       2 bytes
      general purpose bit flag        2 bytes
      compression method              2 bytes
      last mod file time              2 bytes
      last mod file date              2 bytes
      crc-32                          4 bytes
      compressed size                 4 bytes
      uncompressed size               4 bytes
      filename length                 2 bytes
      extra field length              2 bytes

      filename (variable size)
      extra field (variable size)


B.  Data descriptor:
      crc-32                          4 bytes
      compressed size                 4 bytes
      uncompressed size               4 bytes

    This descriptor exists only if bit 3 of the general
    purpose bit flag is set (see below).  It is byte aligned
    and immediately follows the last byte of compressed data.


C.  Central directory structure:
    [file header] . . .  end of central dir record

    file header:
      central file header signature   4 bytes  (0x02014b50)
      version made by                 2 bytes
      version needed to extract       2 bytes
      general purpose bit flag        2 bytes
      compression method              2 bytes
      last mod file time              2 bytes
      last mod file date              2 bytes
      crc-32                          4 bytes
      compressed size                 4 bytes
      uncompressed size               4 bytes
      filename length                 2 bytes
      extra field length              2 bytes
      file comment length             2 bytes
```

```
  disk number start               2 bytes
  internal file attributes        2 bytes
  external file attributes        4 bytes
  relative offset of local header 4 bytes

  filename (variable size)
  extra field (variable size)
  file comment (variable size)


End of central dir record:

  end of central dir signature    4 bytes  (0x06054b50)
  number of this disk             2 bytes
  number of the disk with the
  start of the central directory  2 bytes
  total number of entries in
  the central dir on this disk    2 bytes
  total number of entries in
  the central dir                 2 bytes
  size of the central directory   4 bytes
  offset of start of central
  directory with respect to
  the starting disk number        4 bytes
  zipfile comment length          2 bytes
  zipfile comment (variable size)
```

Definición de una cabera RAR y como Leerla :

```
Un archivo consiste en un conjunto de bloques de tamaño variable. El orden
de estos bloques puede variar, pero el primer bloque debe ser un bloque de
marcador seguido por un bloque de cabezera de archivo.

Cada uno de lo bloques comienza con los siguientes campos:
HEAD_CRC       2 bytes     CRC of total block or block part
HEAD_TYPE      1 byte      Block type
HEAD_FLAGS     2 bytes     Block flags
HEAD_SIZE      2 bytes     Block size
ADD_SIZE       4 bytes     Optional field - added block size

   Field ADD_SIZE present only if (HEAD_FLAGS & 0x8000) != 0
   Total block size is HEAD_SIZE if (HEAD_FLAGS & 0x8000) == 0
and HEAD_SIZE+ADD_SIZE if the field ADD_SIZE is present - when
(HEAD_FLAGS & 0x8000) != 0.

   In each block the followings bits in HEAD_FLAGS have the same meaning:
  0x4000 - if set, older RAR versions will ignore the block
           and remove it when the archive is updated.
           if clear, the block is copied to the new archive
           file when the archive is updated;
  0x8000 - if set, ADD_SIZE field is present and the full block
           size is HEAD_SIZE+ADD_SIZE.

  Declared block types:
HEAD_TYPE=0x72          marker block
HEAD_TYPE=0x73          archive header
HEAD_TYPE=0x74          file header
HEAD_TYPE=0x75          old style comment header
HEAD_TYPE=0x76          old style authenticity information
HEAD_TYPE=0x77          old style subblock
HEAD_TYPE=0x78          old style recovery record
HEAD_TYPE=0x79          old style authenticity information
HEAD_TYPE=0x7a          subblock
   Comment block is actually used only within other blocks and doesn\'t
exist separately.

   Archive processing is made in the following manner:
1. Read and check marker block
2. Read archive header
3. Read or skip HEAD_SIZE-sizeof(MAIN_HEAD) bytes
4. If end of archive encountered then terminate archive processing,
   else read 7 bytes into fields HEAD_CRC, HEAD_TYPE, HEAD_FLAGS,
   HEAD_SIZE.
5. Check HEAD_TYPE.
   if HEAD_TYPE==0x74
     read file header ( first 7 bytes already read )
     read or skip HEAD_SIZE-sizeof(FILE_HEAD) bytes
     if (HEAD_FLAGS & 0x100)
       read or skip HIGH_PACK_SIZE*0x100000000+PACK_SIZE bytes
```

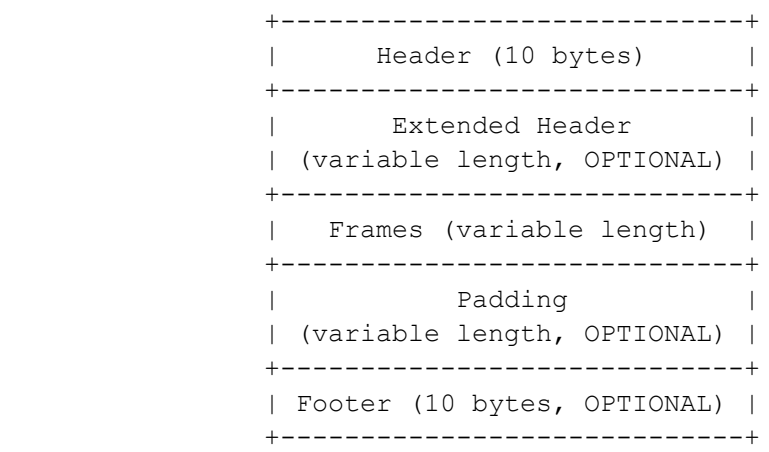
```
      else
        read or skip PACK_SIZE bytes
    else
      read corresponding HEAD_TYPE block:
        read HEAD_SIZE-7 bytes
        if (HEAD_FLAGS & 0x8000)
          read ADD_SIZE bytes
6. go to 4.


                                Block Formats

   Marker block ( MARK_HEAD )

HEAD_CRC        Always 0x6152
2 bytes
HEAD_TYPE       Header type: 0x72
1 byte
HEAD_FLAGS      Always 0x1a21
2 bytes
HEAD_SIZE       Block size = 0x0007
2 bytes

   The marker block is actually considered as a fixed byte
sequence: 0x52 0x61 0x72 0x21 0x1a 0x07 0x00

   Archive header ( MAIN_HEAD )

HEAD_CRC        CRC of fields HEAD_TYPE to RESERVED2
2 bytes
HEAD_TYPE       Header type: 0x73
1 byte
HEAD_FLAGS      Bit flags:
2 bytes
                0x0001  - Volume attribute (archive volume)
                0x0002  - Archive comment present
                          RAR 3.x uses the separate comment block
                          and does not set this flag.
                0x0004  - Archive lock attribute
                0x0008  - Solid attribute (solid archive)
                0x0010  - New volume naming scheme (\'volname.partN.rar\')
                0x0020  - Authenticity information present
                          RAR 3.x does not set this flag.
                0x0040  - Recovery record present
                0x0080  - Block headers are encrypted
                0x0100  - First volume (set only by RAR 3.0 and later)
                other bits in HEAD_FLAGS are reserved for
                internal use
HEAD_SIZE       Archive header total size including archive comments
2 bytes
RESERVED1       Reserved
2 bytes
```

```
RESERVED2       Reserved
4 bytes

   File header (File in archive)

HEAD_CRC        CRC of fields from HEAD_TYPE to FILEATTR
2 bytes         and file name
HEAD_TYPE       Header type: 0x74
1 byte
HEAD_FLAGS      Bit flags:
2 bytes
                0x01 - file continued from previous volume
                0x02 - file continued in next volume
                0x04 - file encrypted with password
                0x08 - file comment present
                       RAR 3.x uses the separate comment block
                       and does not set this flag.
                0x10 - information from previous files is used (solid flag)
                       (for RAR 2.0 and later)
                bits 7 6 5 (for RAR 2.0 and later)
                     0 0 0    - dictionary size   64 KB
                     0 0 1    - dictionary size  128 KB
                     0 1 0    - dictionary size  256 KB
                     0 1 1    - dictionary size  512 KB
                     1 0 0    - dictionary size 1024 KB
                     1 0 1    - dictionary size 2048 KB
                     1 1 0    - dictionary size 4096 KB
                     1 1 1    - file is directory
               0x100 - HIGH_PACK_SIZE and HIGH_UNP_SIZE fields
                       are present. These fields are used to archive
                       only very large files (larger than 2Gb),
                       for smaller files these fields are absent.
               0x200 - FILE_NAME contains both usual and encoded
                       Unicode name separated by zero. In this case
                       NAME_SIZE field is equal to the length
                       of usual name plus encoded Unicode name plus 1.
               0x400 - the header contains additional 8 bytes
                       after the file name, which are required to
                       increase encryption security (so called 'salt').
               0x800 - Version flag. It is an old file version,
                       a version number is appended to file name as ';n'.
              0x1000 - Extended time field present.
              0x8000 - this bit always is set, so the complete
                       block size is HEAD_SIZE + PACK_SIZE
                       (and plus HIGH_PACK_SIZE, if bit 0x100 is set)
HEAD_SIZE       File header full size including file name and comments
2 bytes
PACK_SIZE       Compressed file size
4 bytes
UNP_SIZE        Uncompressed file size
4 bytes
```

```
HOST_OS         Operating system used for archiving
1 byte                 0 - MS DOS
                       1 - OS/2
                       2 - Win32
                       3 - Unix
                       4 - Mac OS
                       5 - BeOS
FILE_CRC        File CRC
4 bytes
FTIME           Date and time in standard MS DOS format
4 bytes
UNP_VER         RAR version needed to extract file
1 byte
                Version number is encoded as
                10 * Major version + minor version.
METHOD          Packing method
1 byte
                0x30 - storing
                0x31 - fastest compression
                0x32 - fast compression
                0x33 - normal compression
                0x34 - good compression
                0x35 - best compression
NAME_SIZE       File name size
2 bytes
ATTR            File attributes
4 bytes
HIGH_PACK_SIZE  High 4 bytes of 64 bit value of compressed file size.
4 bytes         Optional value, presents only if bit 0x100 in HEAD_FLAGS
                is set.
HIGH_UNP_SIZE   High 4 bytes of 64 bit value of uncompressed file size.
4 bytes         Optional value, presents only if bit 0x100 in HEAD_FLAGS
                is set.
FILE_NAME       File name - string of NAME_SIZE bytes size
SALT            present if (HEAD_FLAGS & 0x400) != 0
8 bytes
EXT_TIME        present if (HEAD_FLAGS & 0x1000) != 0
variable size
other new fields may appear here.
```

Definición del estándar ID3 para los formatos MP3, FLAC, Ogg-Vorbis, Monkey's Audio y Musepack.

Estructura general del Tag:



ID3v2 header

The first part of the ID3v2 tag is the 10 byte tag header, laid out as follows:

```
ID3v2/file identifier      "ID3"
ID3v2 version              $04 00
ID3v2 flags                %abcd0000
ID3v2 size             4 * %0xxxxxxx
```

The first three bytes of the tag are always "ID3", to indicate that this is an ID3v2 tag, directly followed by the two version bytes. The first byte of ID3v2 version is its major version, while the second byte is its revision number. In this case this is ID3v2.4.0. All revisions are backwards compatible while major versions are not. If software with ID3v2.4.0 and below support should encounter version five or higher it should simply ignore the whole tag. Version or revision will never be $FF.

The version is followed by the ID3v2 flags field, of which currently four flags are used.

a - Unsynchronisation
Bit 7 in the 'ID3v2 flags' indicates whether or not unsynchronisation is applied on all frames (see section 6.1 for details); a set bit indicates usage.

b - Extended header
The second bit (bit 6) indicates whether or not the header is followed by an extended header. The extended header is described in section 3.2. A set bit indicates the presence of an extended header.

c - Experimental indicator

```
      The third bit (bit 5) is used as an 'experimental indicator'. This
      flag SHALL always be set when the tag is in an experimental stage.

    d - Footer present
      Bit 4 indicates that a footer (section 3.4) is present at the very
      end of the tag. A set bit indicates the presence of a footer.

    All the other flags MUST be cleared. If one of these undefined flags
    are set, the tag might not be readable for a parser that does not
    know the flags function.

    The ID3v2 tag size is stored as a 32 bit synchsafe integer (section
    6.2), making a total of 28 effective bits (representing up to 256MB).

    The ID3v2 tag size is the sum of the byte length of the extended
    header, the padding and the frames after unsynchronisation. If a
    footer is present this equals to ('total size' - 20) bytes, otherwise
    ('total size' - 10) bytes.

    An ID3v2 tag can be detected with the following pattern:
      $49 44 33 yy yy xx zz zz zz zz
    Where yy is less than $FF, xx is the 'flags' byte and zz is less than
    $80.
```

Definicion de la cabecera AVI:

AVI files contain a 56-byte header, starting at offset 32 within the file.

offset	size	description
0	4	time delay between frames in microseconds
4	4	data rate of AVI data
8	4	padding multiple size, typically 2048
12	4	parameter flags
16	4	number of video frames
20	4	number of preview frames
24	4	number of data streams (1 or 2)
28	4	suggested playback buffer size in bytes
32	4	width of video image in pixels
36	4	height of video image in pixels
40	4	time scale, typically 30
44	4	data rate (frame rate = data rate / time scale)
48	4	starting time, typically 0
52	4	size of AVI data chunk in time scale units

D. ANEXO D : Plan de Pruebas

Módulo: Categorías.

Prueba Causa - Efecto

1.- Ingreso de una categoría al sistema.

Los datos de las categorías son únicamente el nombre, este debe ser único.

La acción se ejecuta al presionar el botón Ingresar del formulario “Categorías”

Datos de prueba correctos:

Categoría: “*Programas*”

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Categoría:	Acción: La categoría no se crea.	
Segundo	Categoría: “*default*”	Mensaje: La categoría “default” ya existe.	
Tercero	Categoría: “*Default*”	Mensaje: La categoría “default” ya existe.	
Cuarto	Categoría: “*Programas*”	Acción: La categoría se agrega.	

2.- Modificación de una categoría del sistema.

Los datos de las categorías son únicamente el nombre, este debe ser único.

La acción se ejecuta al presionar el botón Modificar del formulario “Categorías”

Datos de prueba correctos:

Categoría: “*Respaldos*”

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Categoría:	Acción: La categoría no se modifica.	
Segundo	Categoría: “*default*”	Mensaje: La categoría “default” ya existe.	
Tercero	Categoría: “*Default*”	Mensaje: La categoría “default” ya existe.	
Cuarto	Categoría: “*Respaldos*”	Acción: La categoría se modifica.	

3.- Eliminación de una categoría del sistema.

La acción se ejecuta al presionar el botón Eliminar del formulario “Categorías”

Datos de prueba correctos:

Categoría: “*Respaldos*”

Casos	Datos de entrada	Resultado esperado	Resultado obtenido
Primero	Categoría: “*default*”	Mensaje: No se puede eliminar “*default*” mientras contenga discos.	
Segundo	Categoría: “*Respaldos*”	Mensaje: ¿Está seguro que desea eliminar “default”?.	

Módulo: Discos.

Prueba Causa - Efecto

1.- Ingreso de un Disco al sistema.

La acción se ejecuta al presionar el botón Leer CD/DVD del formulario “Principal”

Datos de prueba correctos:

Disco: “*Archivos*”

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Disco Virgen	Mensaje: Sin Disco	
Segundo	Disco de Audio	Acción: El disco se lee y se guarda en la base.	
Tercero	Disco de Video	Acción: El disco se lee y se guarda en la base.	
Cuarto	Disco de Archivos	Acción: El disco se lee y se guarda en la base.	

Módulo: Principal.

Prueba de interfaces gráficas de Usuario

1.- Crear una Base.

Prueba funcional. Se selecciona del Menú la opción “Crear Base”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Crear Base	Se abre el dialogo común de Guardar archivos	

1.1.- Guardar.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Guardar	Se carga la base en blanco y se ve su contenido en el explorador, también se asocia la ubicación seleccionada para crear una copia de la base cada vez que se guarda.	

2.- Abrir una Base.

Prueba funcional. Se selecciona del Menú la opción “Abrir Base”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Abrir Base	Se abre el dialogo común de abrir archivos	

2.1.- Abrir.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Abrir	Se carga la base de datos seleccionada y se ve su contenido en el explorador.	

3.- Guardar una Base.

Prueba funcional. Se selecciona del Menú la opción “Guardar”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Guardar	Se guarda la base en la ubicación elegida anteriormente al momento de crear la base, la base esta comprimida en formato zip.	

4.- Buscar.

Prueba funcional. Se selecciona del Menú la opción “Buscar”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Buscar	Se muestra el formulario de las Búsquedas.	

5.- Explorar Disco.

Prueba funcional. Se selecciona del Menú la opción “Explorar Disco”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Explorar Disco	Se muestra el formulario Explorador.	

6.- Contactos.

Prueba funcional. Se selecciona del Menú la opción “Contactos”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Contactos	Se muestra el formulario Contactos.	

7.- Ubicación.

Prueba funcional. Se selecciona del Menú la opción “Ubicación”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Ubicación	Se muestra el formulario Ubicación.	

8.- Opciones.

Prueba funcional. Se selecciona del Menú la opción “Opciones”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Opciones	Se muestra el formulario Opciones.	

9.- Ayuda.

Prueba funcional. Se selecciona del Menú la opción “Ayuda”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Ayuda	Se carga la Ayuda de la aplicación como html pre compilado.	

10.- Salir.

Prueba funcional. Se selecciona del Menú la opción “Salir”.

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	Presionar opción Salir	Mensaje: ¿Está seguro que desea salir? Aceptar: Se cierra la aplicación. Cancelar: Vuelve a la aplicación.	

Módulo: Contactos.

Prueba Causa - Efecto

1.- Ingreso de un contacto al sistema.

Los datos de los contactos son: nombre, apellido, titulo, sobrenombre, lista de e-mails, dirección, código postal, ciudad, provincia, país, teléfono, fax, móvil. El nombre es el único dato obligatorio.

La acción se ejecuta al presionar el botón Nuevo del formulario “Contactos”

Datos de prueba correctos:

Nombre: “*José*”

Apellido: “*Montoya*”

Lista de e-mails: “*youngblood@vtr.net josemontoya@gmail.com*”

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	nombre: apellido: titulo: sobrenombre: lista de e-mails: dirección: código postal : ciudad : provincia: país: teléfono: fax: móvil:	Mensaje: Debe ingresar por lo menos el Nombre del contacto para agregarlo. Acción: El contacto no se crea.	
Segundo	nombre: José … teléfono: xxx fax: xxx móvil: xxx	Mensaje: No puede ingresar letras en campos de números telefónicos. Acción: El contacto no se crea.	
Tercero	nombre: José apellido: Montoya … lista de e-mails: youngblood@vtr.net josemontoya@gmail.com …	Acción: El contacto se crea.	

2.- Modificación de un contacto al sistema.

La acción se ejecuta al presionar el botón Editar del formulario “Contactos”

Datos de prueba correctos:

Nombre: “*Francisco*”

Apellido: “*Marchant*”

Lista de e-mails: “*youngblood@vtr.net*”

Casos	Clases de Datos	Resultado esperado	Resultado obtenido
Primero	nombre: apellido: titulo: sobrenombre: lista de e-mails: dirección: código postal : ciudad : provincia: país: teléfono: fax: móvil:	Mensaje: Debe ingresar por lo menos el Nombre del contacto para Editarlo. Acción: El contacto no se Modifica.	
Segundo	nombre: Francisco … teléfono: xxx fax: xxx móvil: xxx	Mensaje: No puede ingresar letras en campos de números telefónicos. Acción: El contacto no se modifica.	
Tercero	nombre: Francisco apellido: Marchant … lista de e-mails: youngblood@vtr.net …	Acción: El contacto se modifica correctamente.	

3.- Eliminación de un contacto al sistema.

La acción se ejecuta al presionar el botón Eliminar del formulario “Contactos”

Casos	Datos de entrada	Resultado esperado	Resultado obtenido
Primero	Presiona Eliminar	Mensaje: ¿Esta seguro que desea eliminar el contacto (acción irreversible)?.	

E. ANEXO E : Interfaces Creadas

Módulo: Categorías.

Ilustración 12.1 Ventana principal Categorías.

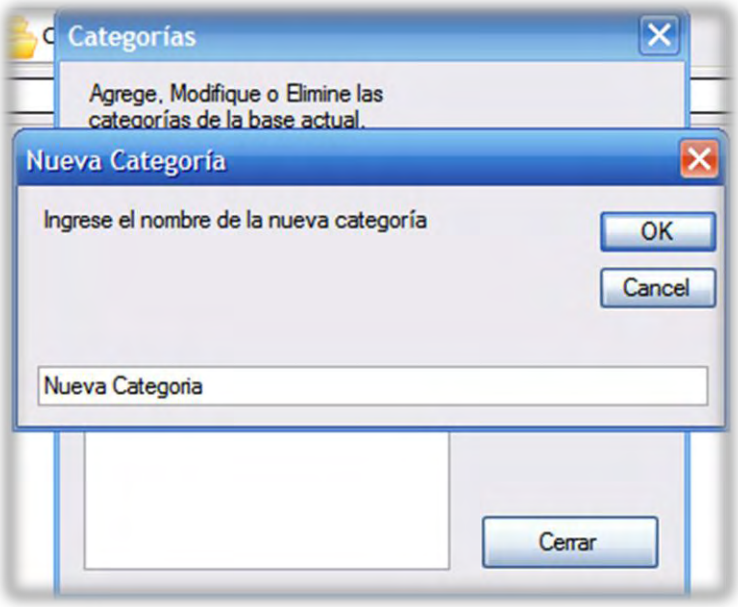


Ilustración 12.2 Ingresar Categorías.

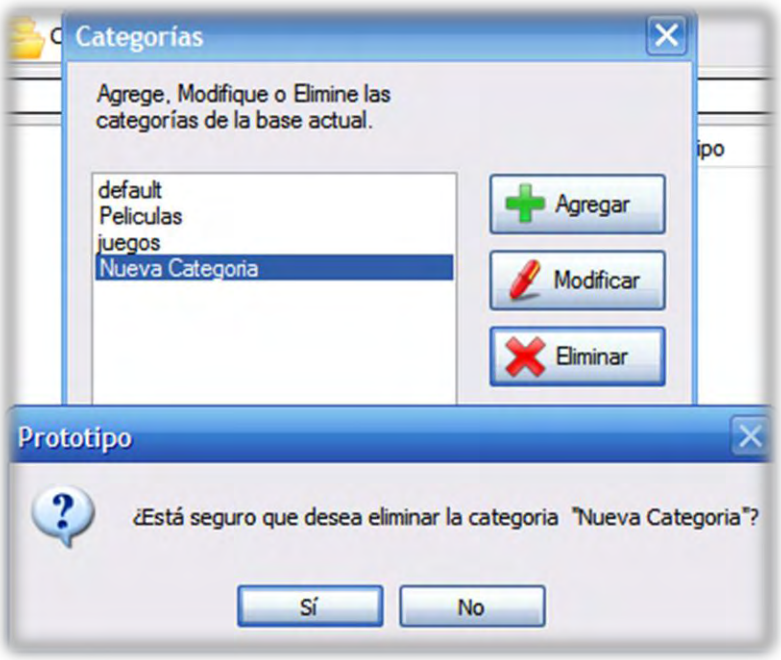


Ilustración 12.3 Eliminar categoría.

Ilustración 12.4 Categoría ya existe.

Módulo: Discos.

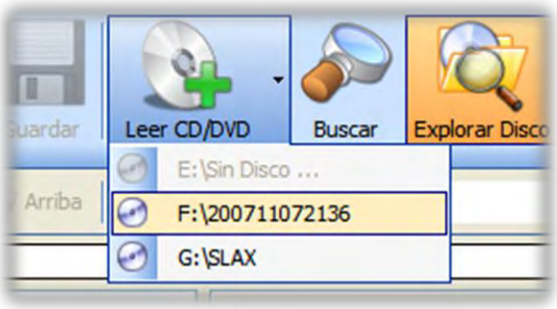


Ilustración 12.5 Menú unidades

Ilustración 12.6 Pre Lectura de Disco

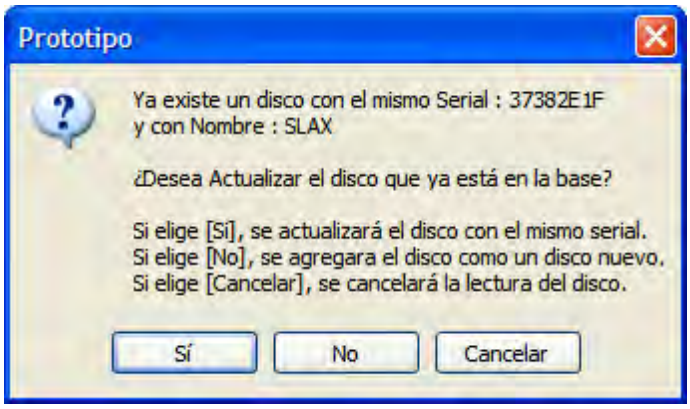


Ilustración 12.7 Mensaje disco repetido

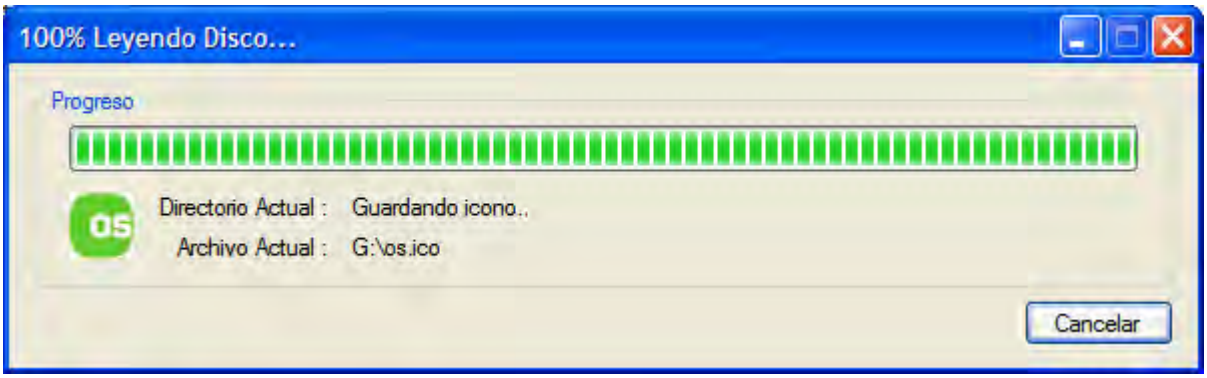


Ilustración 12.8 Lectura de Disco

Módulo: Búsqueda.

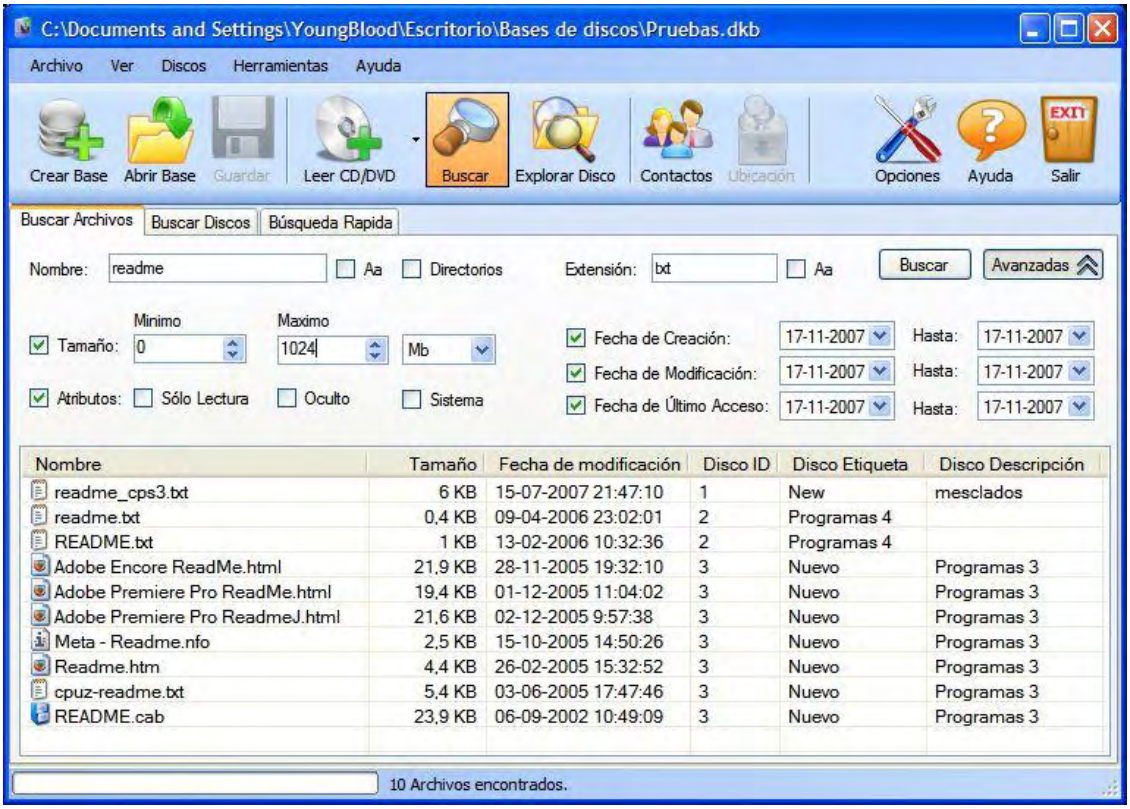


Ilustración 12.9 Búsqueda de archivos

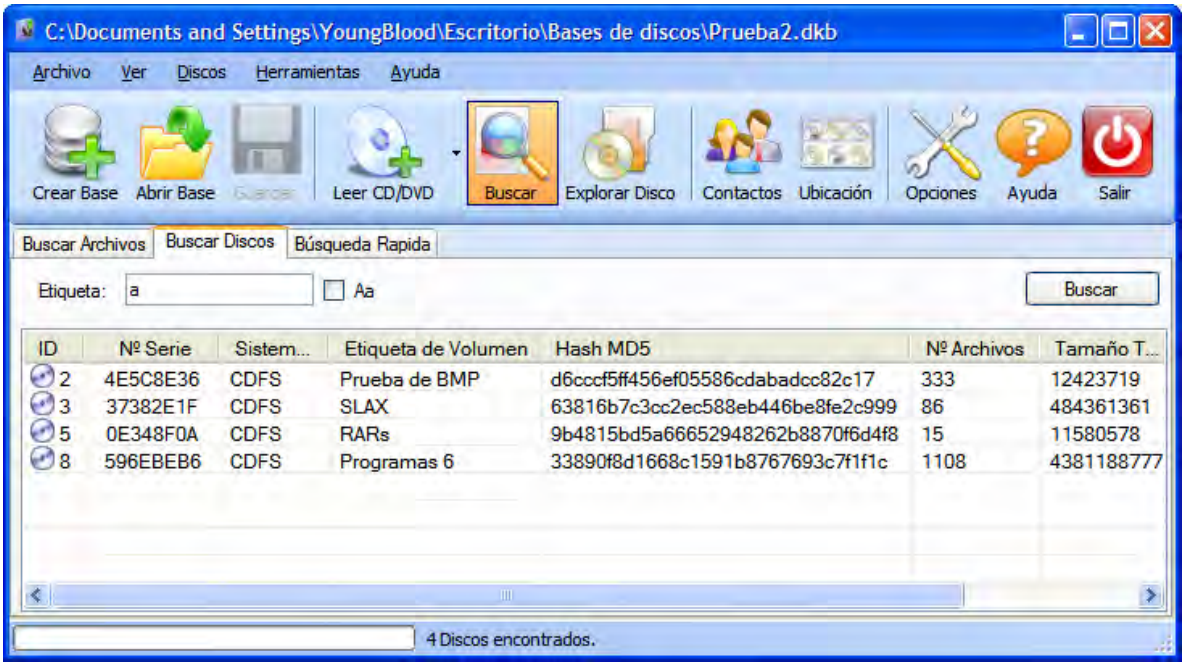


Ilustración 12.10 Búsqueda de discos

Módulo: Principal - Explorador.

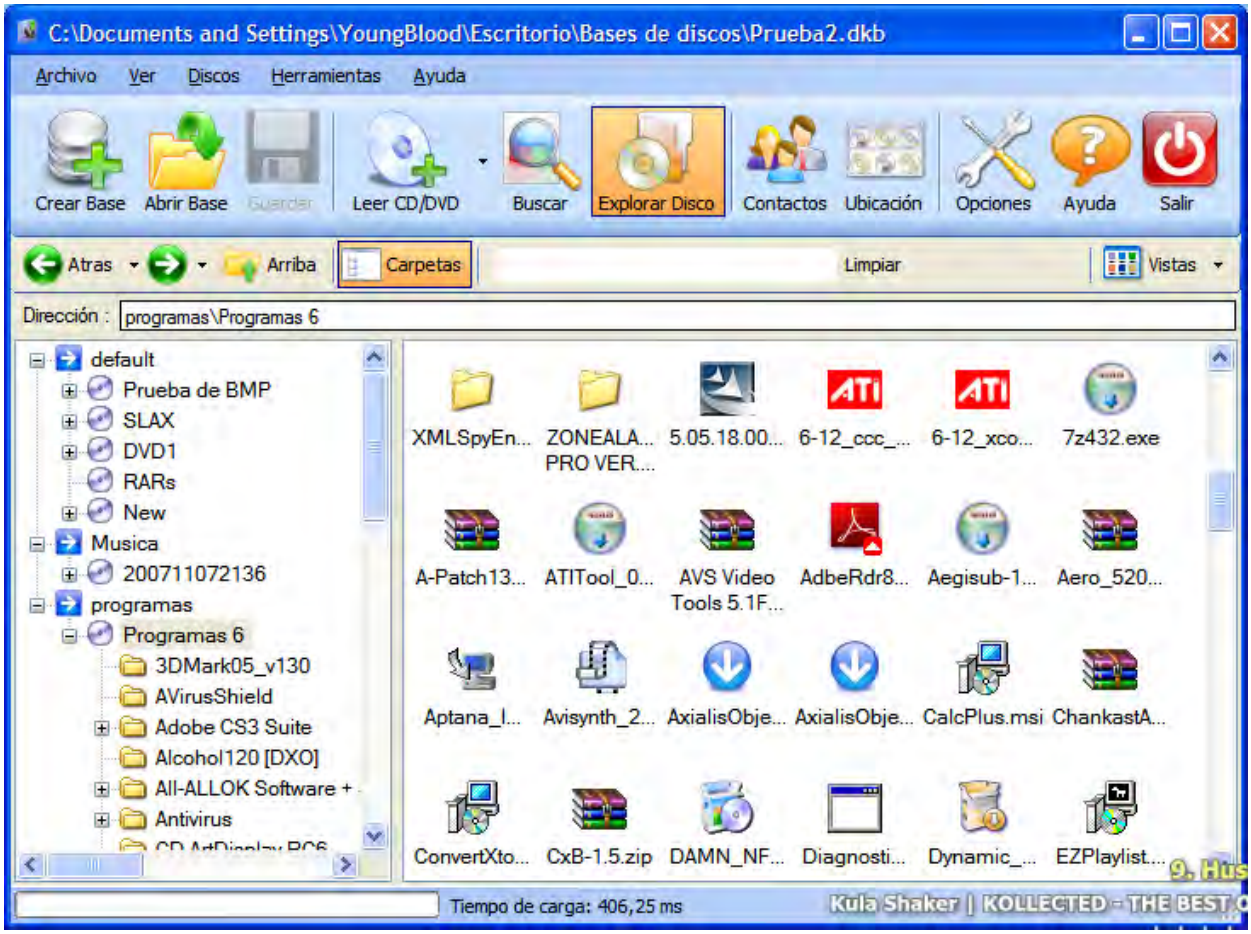


Ilustración 12.11 Explorador de discos

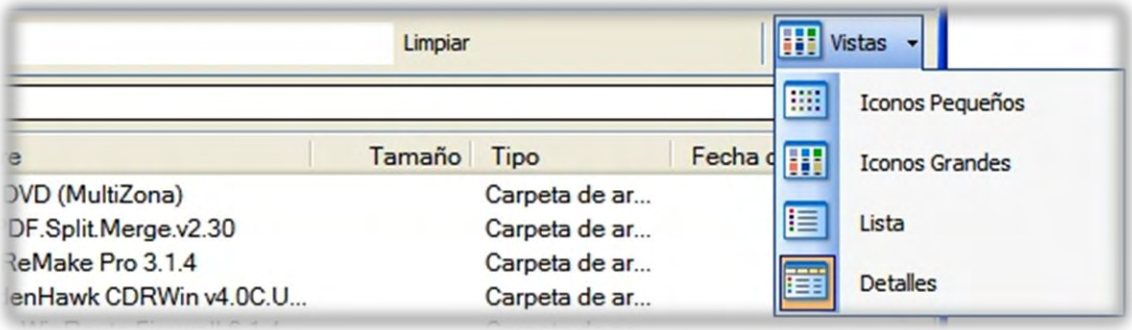


Ilustración 12.12 Vistas del explorador

Ilustración 12.13 Propiedades de un archivo

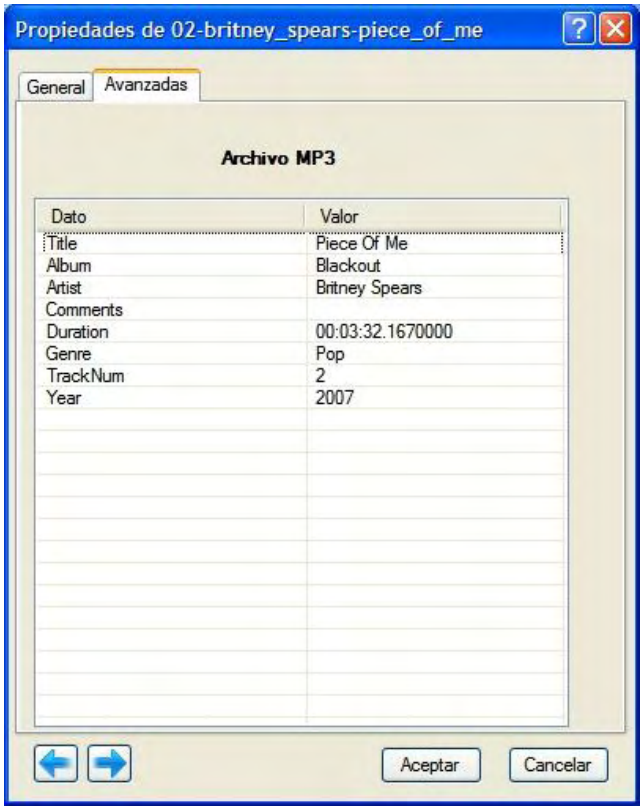


Ilustración 12.14 Propiedades avanzadas de mp3

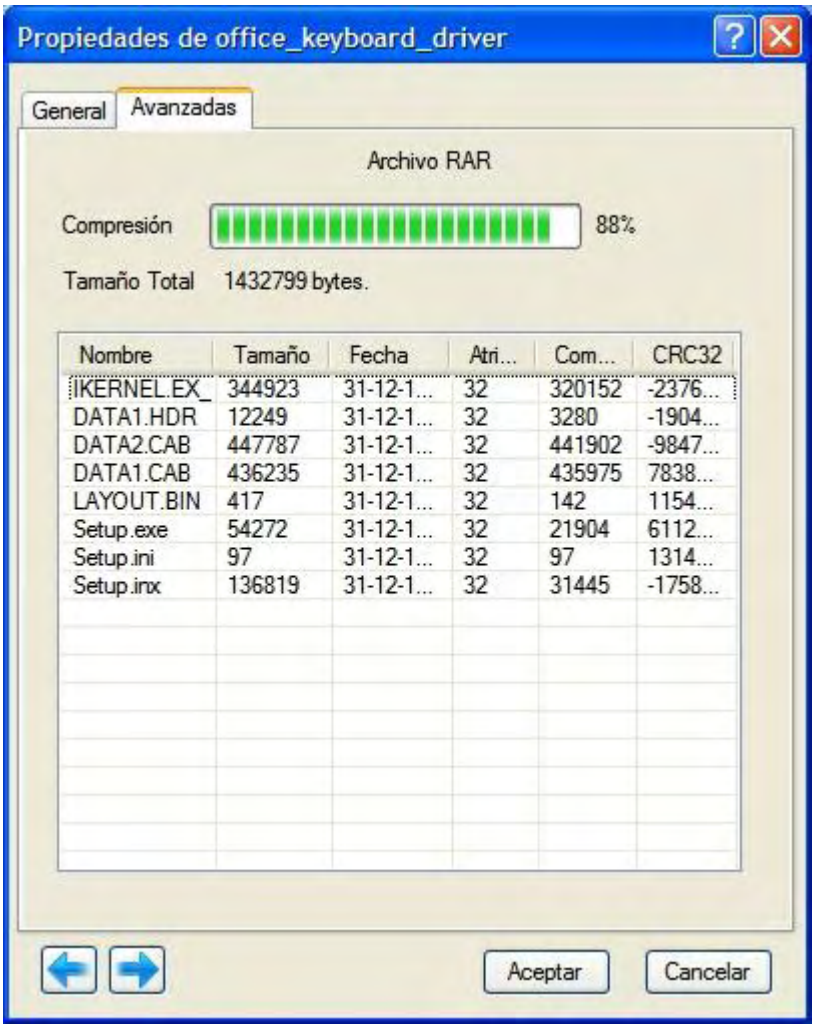


Ilustración 12.15 Propiedades avanzadas de un rar

Ilustración 12.16 Menú contextual de categoría

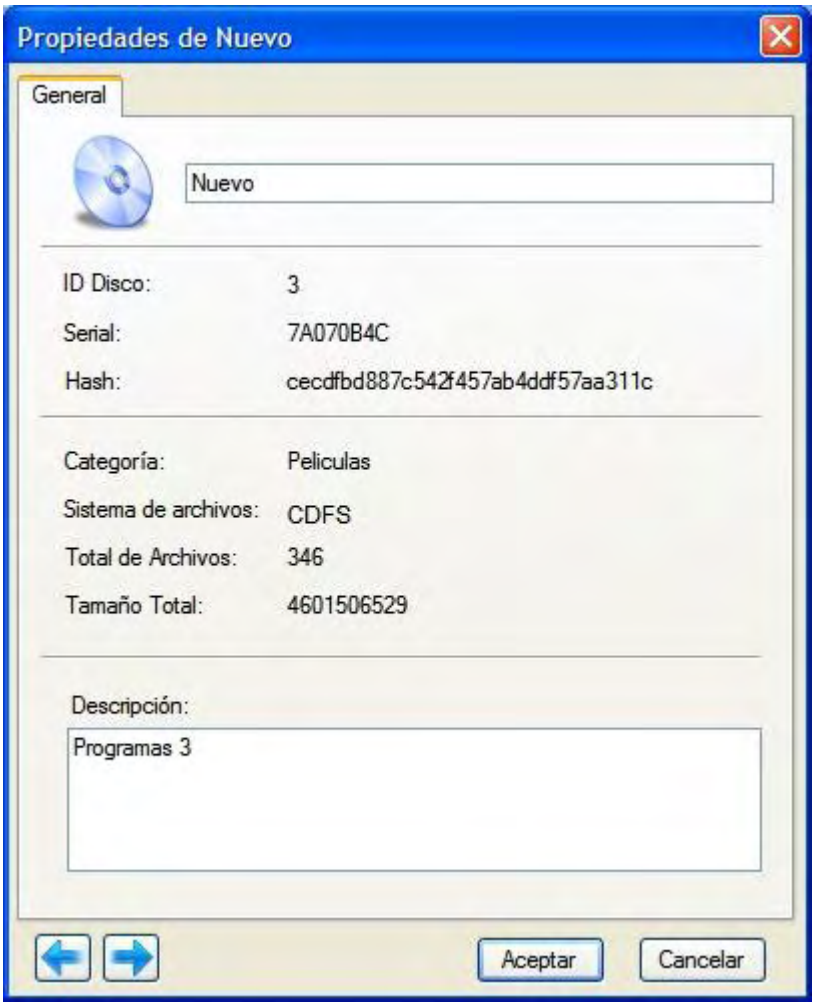


Ilustración 12.17 Propiedades de un disco

Ilustración 12.18 Menú contextual de un disco

Modulo: Contactos.

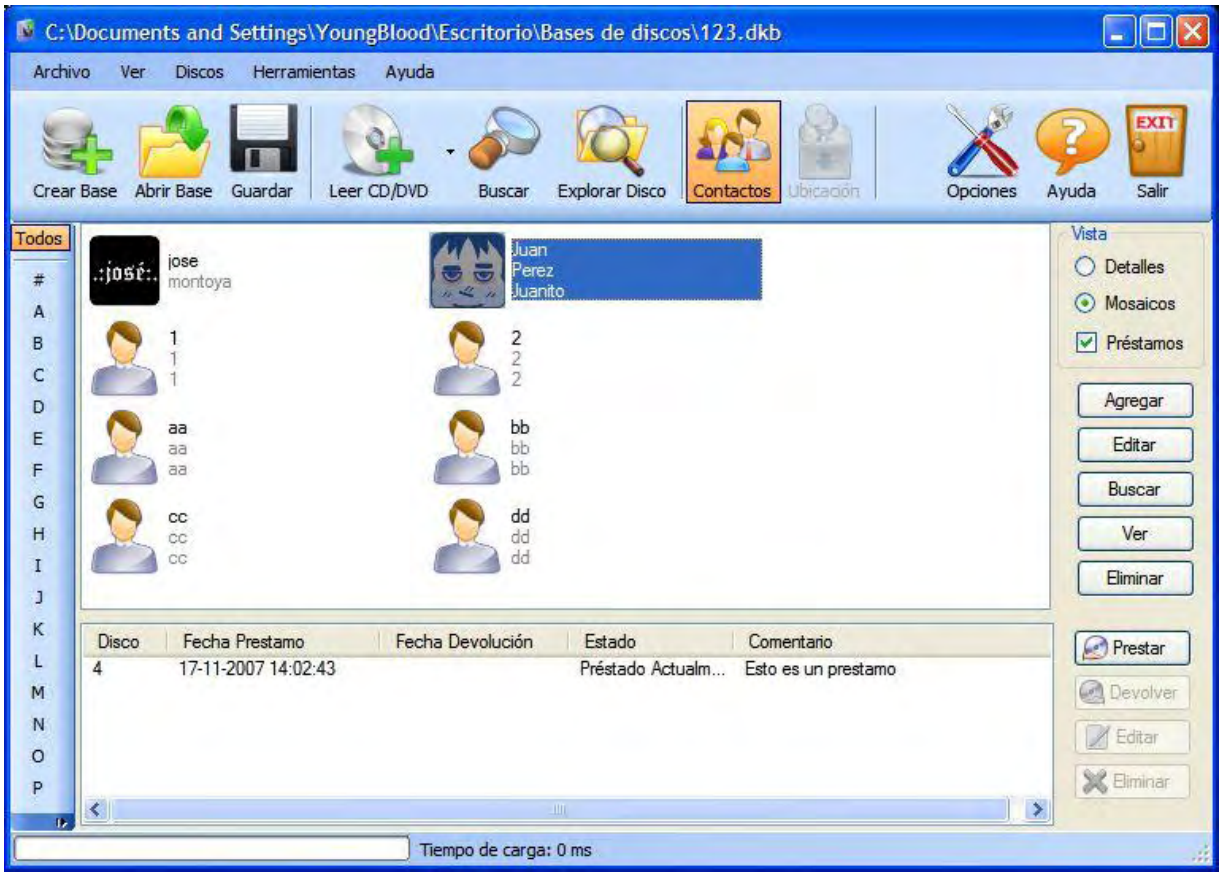


Ilustración 12.19 Contactos

Ilustración 12.20 Modificar contacto

Ilustración 12.21 Datos de ubicación

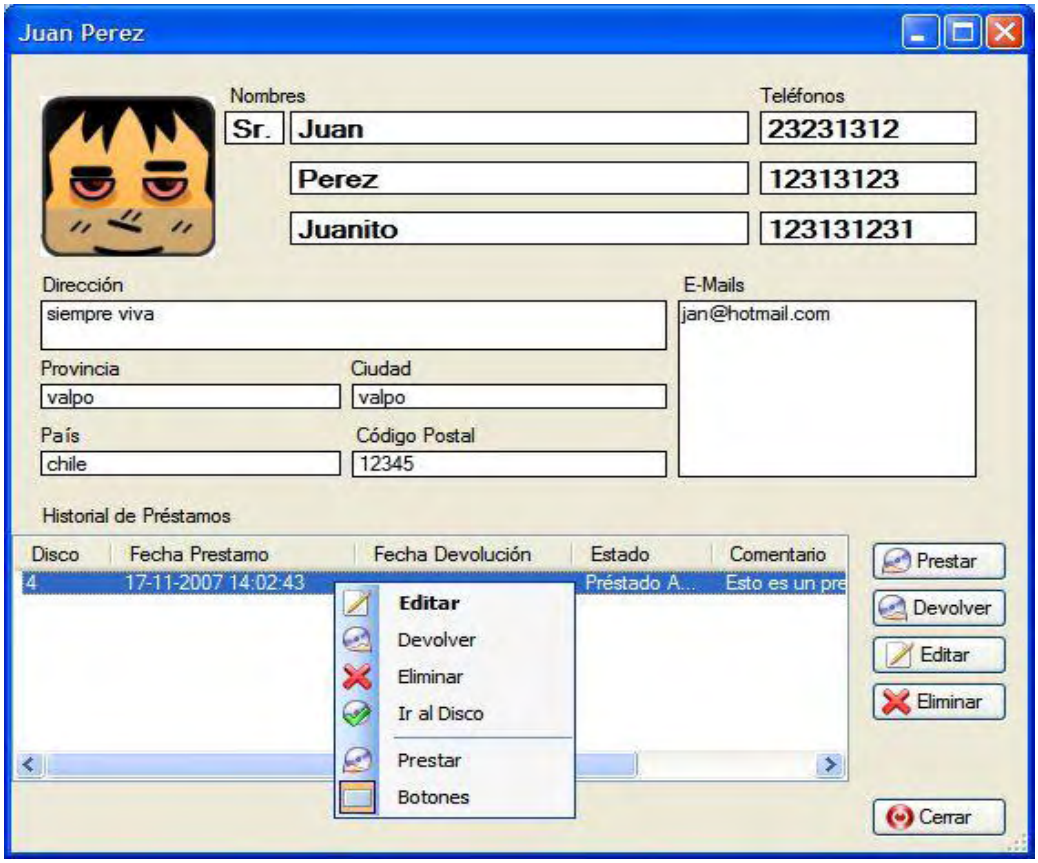


Ilustración 12.22 Ficha de contacto

Ilustración 12.23 Prestar un disco

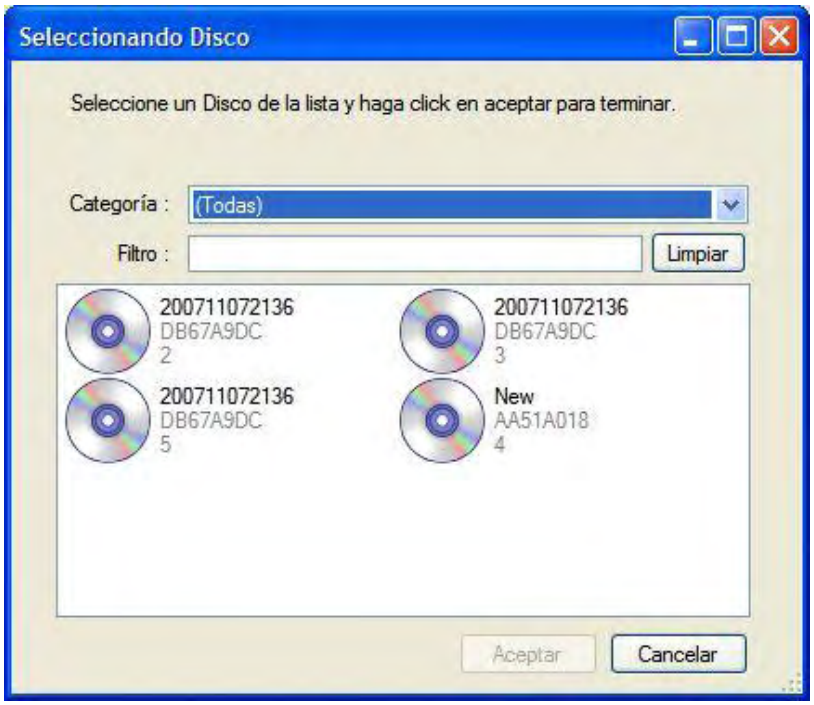


Ilustración 12.24 Filtro de discos

Menú Principal

Ilustración 12.25 Menú archivo

Ilustración 12.26 Menú ver

Ilustración 12.27 Menú discos

Ilustración 12.28 Menú Herramientas

Ilustración 12.29 Menú Ayuda

Ilustración 12.30 Editar Descripción de un disco

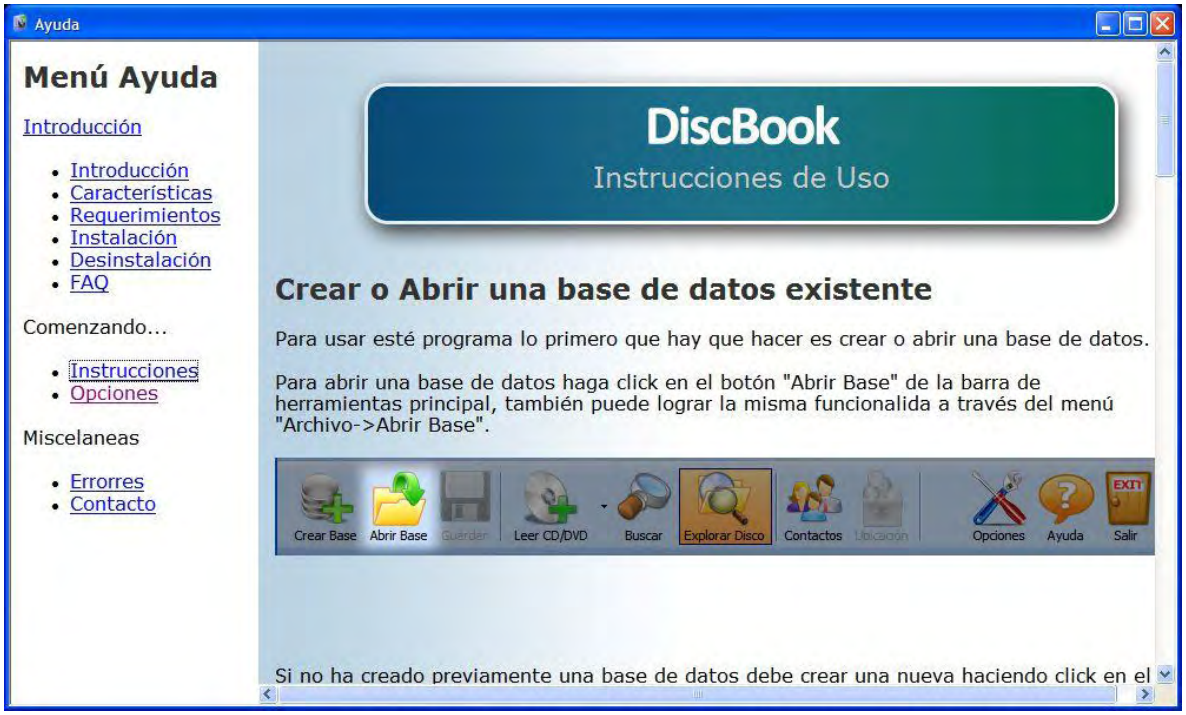


Ilustración 12.31 Manual de ayuda

Módulo: Ubicación.

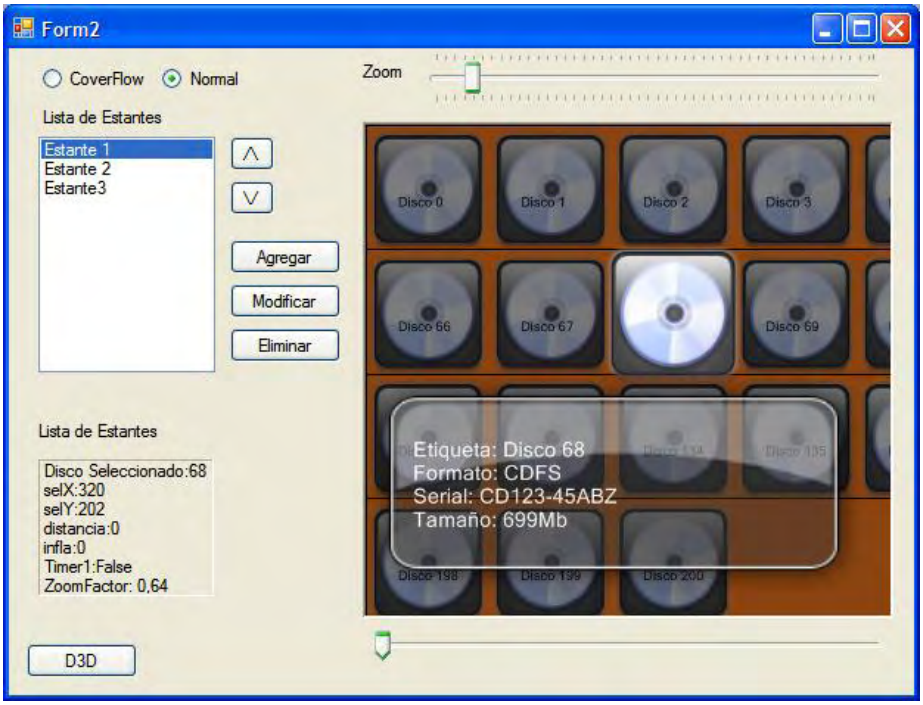


Ilustración 12.32 Prototipo Ubicación

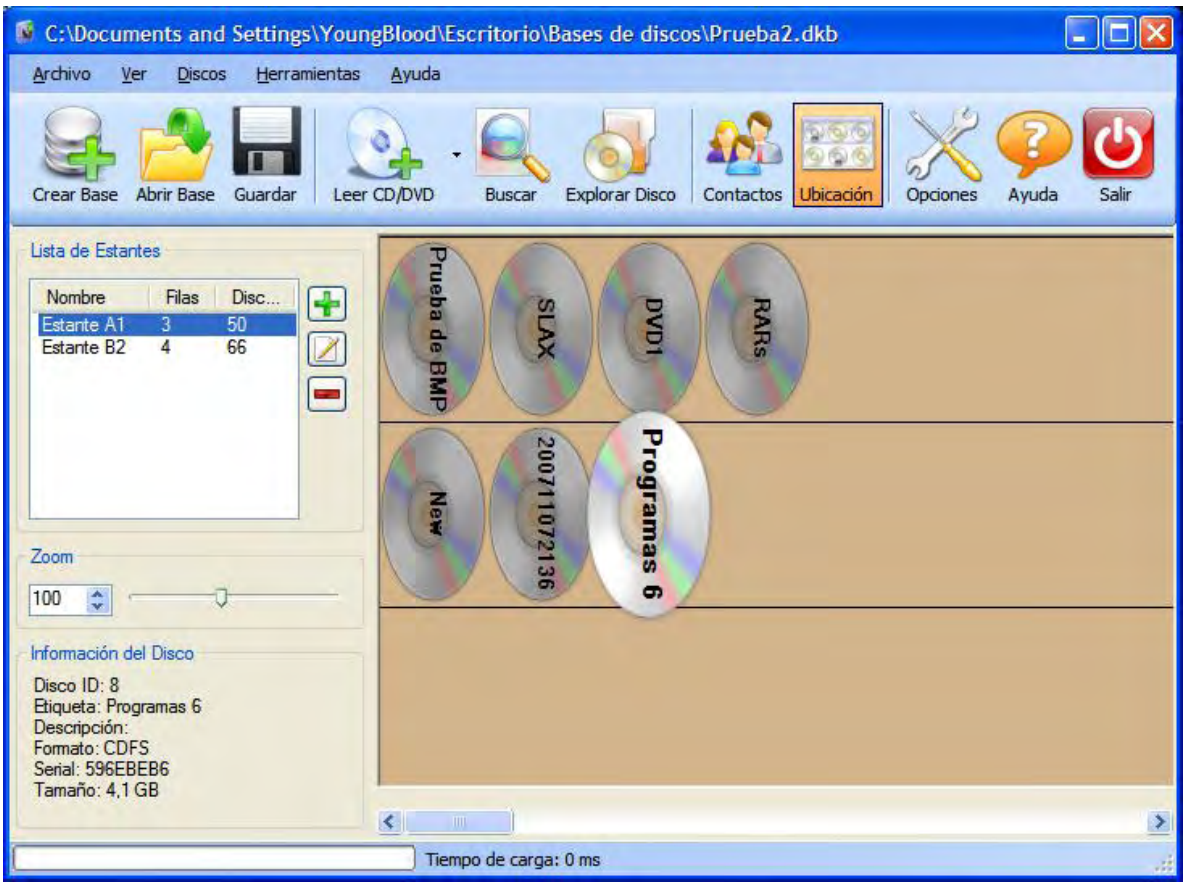


Ilustración 12.33 Ubicación

Módulo: Principal - Opciones.

Ilustración 12.34 Opciones

Ilustración 12.35 Opciones de Plugins

F. ANEXO F : Pruebas de Usabilidad

PRUEBA DE USABILIDAD

Producto a evaluar: DiscBook

Atributo a evaluar: Rendimiento

DESCRIPCIÓN DEL TIPO DE PRUEBA

Este tipo de prueba permitirá obtener datos cuantitativos acerca del rendimiento que usuarios representativos tienen cuando ejecutan ciertas tareas sobre el producto evaluado. Su objetivo principal es verificar si el producto a evaluar satisface ciertas metas de usabilidad previamente establecidas o establecer una comparación con algún producto de la competencia.

RECOMENDACIONES

El evaluador y el usuario no deben tener ningún tipo de interacción DURANTE el desarrollo de la prueba.

El número mínimo de participantes en la prueba debe ser 5 para obtener resultados fiables.

Se recomienda complementar la prueba con entrevistas y/o cuestionarios post-test.

PROCEDIMIENTO

1. Definir los objetivos de la prueba y los factores a medir.

2. Realizar la prueba con usuarios representativos.

3. Analizar los datos para esbozar las conclusiones.

TIPO DE PRODUCTO A EVALUAR

El catalogador electrónico es una herramienta cuyo propósito es permitir la búsqueda y recuperación de información de la manera más rápida y expedita. Cada usuario que posee un computador y dispone de la posibilidad de guardar información en medios externos desea mantener esta información ordenada. En particular, un Catalogador de Discos es un programa que permite a un usuario con muchos discos, acceder a su contenido, sin necesidad de introducirlos a su computador.

PRODUCTO ESPECÍFICO A EVALUAR

El producto que se debe evaluar es DiscBook un programa de catalogación de discos gratuito, proporcionado por José Montoya, el cual permite leer un Disco, catalogarlo y guardar las propiedades de los archivos en el, como propiedades me refiero a: Nombre, Directorio, Atributos, Tamaño, Fecha, Descripción del usuario.

También guarda la información con respecto al disco, Serial, Sistema de Archivos, Etiqueta de Volumen, Número de Archivos, Tamaño total del disco y Descripción.

El usuario puede buscar un archivo en particular en la base o si prefiere puede navegar a través de todos los discos de forma manual, sin ni siquiera insertar el Disco.

Se podrá crear y administrar una lista de contactos los cuales se asocian a los discos cuando estos son prestados, de los cuales se mantendrá un registro de Nombres, Apellidos, Teléfonos, Mails, Dirección e Historial de préstamos.

El explorador de discos tiene una apariencia muy cercana al de MS Windows, en el cual se puede visualizar los archivos tal cual como si el disco estuviera conectado al PC, respetando los iconos de cada formato e incluso conservando los iconos de los ejecutables.

Tiene una Interfaz de Usuario intuitiva y fácil de usar.

Es capaz de leer las etiquetas ID3v1 e ID3v2 que contienen algunos archivos multimedia.

Es capaz de leer el contenido de archivos comprimidos RAR.

Las bases serán guardadas en un formato comprimido, ahorrando espacio.

Se puede agrupar los disco por categorías al momento de almacenarlos, por ejemplo: Juegos, Música, Películas, Programas, etc., las categorías serán creadas por el usuario.

Muestra el progreso de lectura del disco de manera amigable, mediante varios métodos (Pre-contar Archivos, Por Tamaño de Disco y Archivos, Híbrido) lo que permite el correcto funcionamiento con discos multi-sesion.

OBJETIVOS DE LA PRUEBA

• Obtener medidas sobre la eficiencia de uso del producto.

• Obtener medidas sobre la cantidad de errores que comete el usuario al usar el producto.

• Obtener opiniones acerca de la satisfacción de uso.

FACTORES A MEDIR

• Tiempo que los usuarios toman en completar una tarea específica.

• Cantidad de tareas que el usuario puede realizar en un tiempo límite dado.

• Cantidad de errores que comete el usuario al usar el producto.

• Tasa de interacciones exitosas y erróneas.

• Tiempo utilizado por el usuario en recuperarse de los errores.

• Proporción de usuarios que usan estrategias de trabajo eficientes en caso de que exista más de una forma de realizar una tarea.

PERFILES DE USUARIOS

Se considerará:

• “Usuario Novato” el usuario que graba menos de 2 o 3 discos a la semana DiscBook,

• “Usuario Experto” el usuario que graba más de 2 o 3 discos a la semana DiscBook.

DEFINICIÓN DE TAREAS

Tarea Nº 1

Descripción: Crear una base de Datos.

Tiempo Máximo (Usuario Novato): 2 minutos.

Tiempo Máximo (Usuario Experto): 1 minutos.

Caso Éxito: La base está creada y lista para su uso.

Error: No lograr crear la base.

Tarea Nº 2

Descripción: Agregar un nuevo Disco a la base de datos.

Tiempo Máximo (Usuario Novato): 5 minutos.

Tiempo Máximo (Usuario Experto): 2 minutos.

Caso Éxito: Lograr agregar un nuevo Disco.

Error: No lograr agregar un nuevo Disco.

Tarea Nº 3

Descripción: Agregar una nueva Categoría a la base.

Tiempo Máximo (Usuario Novato): 5 minutos.

Tiempo Máximo (Usuario Experto): 3 minutos.

Caso Éxito: Lograr crear la categoría.

Error: No lograr crear la categoría.

Tarea Nº 4:

Descripción: Agregar un nuevo contacto.

Tiempo Máximo (Usuario Novato): 10 minutos.

Tiempo Máximo (Usuario Experto): 5 minutos.

Caso Éxito: Lograr crear el contacto.

Error: No lograr crear el contacto.

Tarea Nº 5:

Descripción: Abrir una base previamente creada.

Tiempo Máximo (Usuario Novato): 2 minutos.

Tiempo Máximo (Usuario Experto): 1 minutos.

Caso Éxito: Lograr abrir la base.

Error: No lograr abrir la base.

Tarea Nº 6:

Descripción: Buscar un archivo en la base abierta.

Tiempo Máximo (Usuario Novato): 4 minutos.

Tiempo Máximo (Usuario Experto): 2 minutos.

Caso Éxito: Encontrar el disco al cual pertenece el archivo buscado.

Error: No encontrar el disco al cual pertenece el archivo buscado.

Tarea Nº 7:

Descripción: Prestar el disco X encontrado al contacto Y.

Tiempo Máximo (Usuario Novato): 4 minutos.

Tiempo Máximo (Usuario Experto): 2 minutos.

Caso Éxito: Lograr crear el préstamo.

Error: No lograr crear el préstamo.

Cuestionario Pre-Test

Información Personal del Participante.

1. ¿Qué edad tiene? _____
2. ¿Cuál es su sexo? ☐ **M** ☐ **F**
3. ¿Cuál es su ocupación? _____
4. ¿Tiene experiencia previa en pruebas de este tipo? ☐ **Si** ☐ **No**
5. ¿Qué nivel de estudios posee?
    a. Enseñanza media completa
    b. Técnico
    c. Universitario
    d. Otro (especificar) _____

Información sobre uso de Discos.

1. ¿Con qué frecuencia graba discos?
    a. Todos los días
    b. 2 o 3 veces por semana
    c. Una vez al mes
    d. Nunca
2. ¿Cómo aprendió a grabar discos?
    a. Auto aprendizaje
    b. Cursos de computación
    c. Por medio de la enseñanza de amigos o familiares
    d. Otros medios (especificar) _____
3. De acuerdo a su experiencia en el uso de discos ¿En qué categoría se identifica?
    a. Usuario nivel básico
    b. Usuario nivel medio
    c. Usuario nivel avanzado
4. ¿En qué lugar usa el computador habitualmente?
    a. Casa
    b. Trabajo
    c. Lugar de estudio
    d. Otro (especificar) _____

5. ¿Que programas utiliza con frecuencia? (puede seleccionar más de una opción)
    a. Quemadores
    b. Juegos,peliculas
    c. Sitios o páginas en WWW
    d. Descargas
    e. Otros (especificar)

Información sobre el contexto de uso de Internet

1. ¿Cuál es la velocidad de acceso a Internet en el lugar donde habitualmente se conecta?
    a. Igual o menor que 128 Kbps.
    b. Entre 128 Kbps y 512 Kbps.
    c. Entre 512 Kbps y 1 MB.
    d. Mayor a 1 MB.
    e. No sé.
2. ¿Cuál es el sistema operativo que usa habitualmente?
    a. Microsoft Windows.
    b. Linux/Unix
    c. Apple Mac Os/X
    d. No sé.
    e. Otro (especificar) _____
3. ¿Cuál es el navegador Web que utiliza habitualmente?
    a. Mozilla Firefox
    b. Internet Explorer
    c. Opera
    d. No sé.
    e. Otro (especificar) _____
4. ¿De qué tipo de línea dispone en el lugar habitual de conexión a Internet? (Marque tan solo una opción)
    a. ADSL
    b. Conexión telefónica
    c. Conexión inalámbrica
    d. No sé.
    e. Otra (especificar) _____

Instrucciones

Estimado(a) colaborador(a) a continuación Ud. participará en una prueba para evaluar el grado de usabilidad del programa DiscBook. Esta prueba tiene por objetivo detectar la existencia de problemas en el uso de algunas de las funcionalidades que brinda este programa a sus usuarios, en el marco de un estudio sobre la usabilidad en este proyecto.

La prueba tiene 3 etapas:

1. En la primera etapa Ud. deberá completar un breve cuestionario con preguntas relativas a su experiencia y contexto habitual de uso de Internet.
2. En la segunda etapa se le proporcionarán un conjunto de tareas las que deberá tratar de realizar en el programa discbook.
3. En la tercera etapa se le entregará otro breve cuestionario que tiene por objetivo obtener la impresión general que Ud. tuvo luego de su experiencia de uso del programa.

¡No se preocupe si comete algún error, es normal, no existen malos o buenos desempeños!
¡Recuerde que no lo estamos evaluando a Ud. sino al programa DiscBook!
Toda la información que Ud., nos proporciona es absolutamente confidencial y muy relevante para nuestro estudio, por lo cual le agradecemos su desinteresada cooperación.

SI TIENE ALGUNA DUDA DURANTE EL DESARROLLO DEL TEST, POR FAVOR CONTACTESE CON EL EVALUADOR.

TAREA 1:

Crear una nueva base de datos.

Descripción:

Abra el programa DiscBook, y haga clic en el botón “Crear Base”, guarde la Base en “Mis Documentos” con el nombre de “Prueba.dkb”.

TAREA 2:

Agregar un nuevo Disco a la base de datos.

Descripción:

Haga click en el botón “Leer CD/DVD”, luego haga click en el botón que contiene el DVD a leer.

TAREA 3:

Agregar una nueva Categoría a la base.

Descripción:

Haga click en el menú principal “Herramientas->Categorías”, haga click en el botón “+ Agregar” Introduzca el nombre de la categoría “Programas”.

TAREA 4:

Agregar un nuevo contacto.

Descripción:

Haga click en el botón “Contactos” de la barra principal, agregue sus datos como un nuevo contacto.

TAREA 5:

Abrir una base previamente creada.

Descripción:

Haga click en “Abrir base”, navegue hasta “Mis Documentos” y seleccione la base llamada “nueva.dkb”.

TAREA 6:

Buscar un archivo en la base abierta.

Descripción:

Haga click en “Buscar”, ingrese el término a buscar “data”, presione enter, anote en esta hoja el Disco ID y el Disco Etiqueta de este.

TAREA 7:

Prestar el disco 1 encontrado al contacto “Juan Pérez”.

Descripción:

Haga click “Contactos”, Preste el Disco con etiqueta “JMontoya” (ID: 1).

Cuestionario Post-Test

Para cada una de las siguientes preguntas, marque la nota más apropiada, en una escala de 1 a 7. Para cada pregunta el cuadro presenta el significado de la nota máxima y mínima.

1.	Encontrar información específica en este programa es:	Muy fácil	7 6 5 4 3 2 1	Muy difícil
2.	La función de búsqueda de archivos que proporciona este programa es:	Muy buena	7 6 5 4 3 2 1	Muy pobre
3.	La apariencia del programa (colores y gráfica) es:	Muy buena	7 6 5 4 3 2 1	Muy pobre
4.	El producto se preocupa de mi satisfacción como usuario:	Sí, mucho.	7 6 5 4 3 2 1	No, para nada.
5.	¿Cuán relevante es la funcionalidad entregada por el programa catalogador?	Muy relevante	7 6 5 4 3 2 1	Para nada relevante
6.	Este programa DiscBook es:	Muy útil	7 6 5 4 3 2 1	Inútil
7.	¿Estaría dispuesto a usar nuevamente éste programa DiscBook?	Definitivamente	7 6 5 4 3 2 1	Nunca
8.	En relación a otros programas de catalogación existentes, este es:	Mucho Mejor	7 6 5 4 3 2 1	Mucho Peor