Capítulo III: Descripción del Sistema

En éste capítulo, explicaremos a detalle el funcionamiento total del sistema. Describiremos las distintas técnicas utilizadas para resolver los problemas mencionados en el capítulo anterior. Entraremos a detalle al análisis de requerimientos y al análisis del diseño del sistema. Es fin de éste capítulo no dejar ninguna duda acerca del funcionamiento total del sistema.

3.1 Descripción General del Sistema

Es finalidad de este trabajo de tesis, desarrollar un sistema con el cual podamos administrar fácilmente y con total transparencia un sitio web, el cual nos permita realizar algunas tareas como: buscar archivos de audio MP3, agregar archivos de audio MP3 a la colección, bajar dichos archivos, administrar proveedores (altas y modificaciones), realizar actualizaciones del sitio, etc. Todo esto con total transparencia para el usuario final, los proveedores y el administrador del sitio. Pero entonces, ¿qué queremos decir con transparencia? Este, es uno de los términos computacionales más sobrecargados que existen. En realidad, dependiendo del contexto, el término transparencia puede tener varios significados. En contexto de este trabajo de tesis, nosotros definimos transparencia, como la característica que tiene un sistema computacional para ser utilizado y/o administrado con un conocimiento básico de computación. Es decir, que el sistema puede ser utilizado por cualquier usuario que tenga un conocimiento básico de computación. Decidimos utilizar el término transparencia, ya que los usuarios que utilicen y/o administren dicho sistema, no tendrán que escribir una sola línea de código, todo se realiza a través de interfases gráficas sencillas las cuales esconden en funcionamiento total del sistema haciéndolo parecer una “caja negra”; de manera que todos los procesos internos son transparentes e invisibles para cualquier tipo de usuario.

La aplicación a desarrollar le dará al usuario final del sistema (personas que buscan archivos MP3) la oportunidad de obtener este tipo de archivos de una fuente cien por ciento confiable, donde no existan problema de derechos de autor y donde el usuario va a estar seguro de que el archivo que está bajando es justamente lo que desea, sin problemas de incoherencia con la información de los archivos, y sabiendo que dichos archivos se encuentran completos y perfecto estado. Se partirá de la hipótesis de que existen una serie de grupos musicales, artistas y/o disqueras los cuales desean publicar parte o todo su trabajo gratuitamente a manera de promoción en la red y que ellos utilizan esta aplicación para que sus archivos puedan ser bajados por cualquier persona. De esta manera la gente que baje archivos desde este "Portal Musical", sabrá que toda la música contenida dentro del sitio es confiable y se encuentra respaldada por el artista mismo. Cada artista o disquera que esté registrada dentro del sistema contará con un pequeño espacio en el cual podrá dar una breve descripción de su trabajo, mostrar alguna fotografía y especificar una liga a su página web.

El sistema podrá ser accedido desde cualquier computadora con servicio de Internet; y para cada tipo distinto de usuario habrá una interfaz personalizada de acuerdo a los diferentes servicios a los que tenga acceso. Se manejarán tres tipos de usuarios:

- **Administrador**

  Lleva todas las tareas administrativas del sitio. Se encarga de las actualizaciones del sitio, altas de proveedores, etc.

- **Proveedores**

  Son los usuarios (compañías disqueras o artistas) que se encargan de subir archivos de audio al sistema. Para poder hacerlo necesitarán estar registrados dentro del mismo. Gozarán de un perfil propio y un pequeño espacio en el cual podrá dar una breve descripción de su trabajo, mostrar alguna fotografía y especificar una liga a su página web.

- **Usuarios Finales**

  Cualquier persona que desee realizar una búsqueda y bajar archivos de audio MP3 dentro del sistema. Pueden también navegar por todo el sitio visitando los

diferentes artículos, la descripción de los distintos proveedores, buscar artistas por género, etc.

En lo que al buscador se refiere, como mencionamos en el capítulo anterior, no se realizan búsquedas sobre patrones de cadenas de bits dentro de los archivos de audio, más bien, nos apoyamos en técnicas existentes de manejo de datos no estructurados; en específico se utilizan metadatos para etiquetar la información más importante sobre la que se realizan las búsquedas; de manera que las búsquedas se realizan sobre datos semiestructurados y no sobre los datos no estructurados.
Se utiliza un índice generado dinámicamente sobre el cuales se realizan las búsquedas. Esto agiliza el tiempo de respuesta del buscador ya que el índice se encuentra completamente cargado en memoria principal, y las búsquedas no se realizan sobre la base de datos relacional ni sobre los archivos en si, sino sobre el índice.
Se utilizan técnicas sencillas recuperación de información para poder presentarle al usuario los resultados ordenados de manera descendente con respecto a su grado de relevancia hacia la consulta introducida por dicho usuario.

La diferencia entre este sistema y otros sistemas tipo P2P, (*Peer-to-Peer*: Kazaa, Ares, Overnet, etc) es que todos los archivos de audio MP3 se encuentran concentrados dentro del sistema y no repartidos por todo el mundo en computadoras personales donde existe gran incertidumbre hacia la calidad de dichos archivos, los derechos de autor, la posibilidad de encontrar virus dentro de ellos, entre otras cosas. Todos los archivos registrados dentro sistema, fueron generados por los propios artistas, se encuentran en perfecto estado y gozan de muy buena calidad.

Hay que clarificar que esto es una hipótesis, para desarrollar este trabajo de tesis se utilizaron archivos los cuales se obtuvieron de los discos originales comprados anteriormente. El sistema sí funcionará pero no se pondrá a acceso público, es decir, no se subirá a Internet. Los archivos MP3 no se pondrán a disposición del público, solamente se hará uso personal de ellos de manera de que no se estará violando ninguna ley de derechos de autor. No se contactará a ningún artista ni disquera para que nos proporcione archivos ni para que hagan uso del sistema. Será una simulación de cómo funcionaría el sistema si éste estuviera publicado en Internet funcionando al cien por ciento.

Por fines prácticos, se decidió darle un nombre a nuestro sistema. El sistema se llama **wAudio**, referente a lo que sería un portal web de audio.

3.2 Análisis de Requerimientos

Dentro de este apartado, definiremos las funciones que el sistema será capaz de realizar, es decir, contestaremos la pregunta: ¿qué es lo que el sistema puede hacer? Se definieron y analizaron los diferentes tipos de usuarios que podrán hacer uso de dicho sistema. Como describimos anteriormente, existen tres tipos de usuarios: usuario final, proveedor y el administrador del sistema. Para cada uno, se definieron y analizaron las distintas funciones que serán capaces de llevar a cabo.

Además de definir y analizar dichas funciones, se analizó la manera en que los datos van a fluir dentro del sistema. Se definieron los diferentes tipos de datos que se necesitarán, y como esos datos fluirán dentro del sistema. Prácticamente, se definieron las entradas y las salidas que el sistema tendrá, y se definieron los procesos que recibirán tales entradas y los procesos que generarán dichas salidas.

3.2.1 Casos de Uso

Los casos de uso, nos ayudan a definir cada una de las funciones que el sistema será capaz de realizar. Cada tarea que el sistema lleve a cabo, necesita quedar definida como un caso de uso y necesitamos analizar su funcionamiento.

Dentro de los casos de uso, hemos definido seis actores los cuales interactúan con cada una de las funciones descritas. Estos actores son los que inician cualquier proceso y reciben la salida dichos procesos. El sistema cuenta con seis actores principales. Tres de ellos (el administrador, el usuario final y el proveedor) ya han sido explicados anteriormente; los tres actores restantes definidos dentro del sistema son:

- **Base de Datos**

  Es el lugar donde se almacena toda la información referente a los archivos de audio. También se guarda la información de los distintos usuarios junto con su

descripción. Toda la información referente al contenido del portal, o sea los distintos artículos que se presenten dentro del sitio, también será almacenada dentro de la base de datos.

- **Índice**

  Almacena los valores de los atributos principales de los archivos de audio. Es justo aquí donde se realizan todas las búsquedas dentro del sistema. Dicho actor se encuentra cargado en memoria principal todo el tiempo que el sistema se encuentra funcionando.

- **Disco Duro**

  Se encarga de almacenar los archivos de audio físicamente. La información de los archivos es almacenada en la base de datos y en el índice, pero los archivos son almacenados dentro de éste actor.

  El disco duro también se encarga de almacenar los archivos de las imágenes las cuales son parte de la descripción de los proveedores. La descripción de dichos proveedores es almacenada en archivos de texto dentro del disco duro; de igual manera el texto de los artículos presentados dentro del sitio así como los archivos de las imágenes que los acompañan.

Una vez definidos los actores, podemos entonces comenzar el análisis de la interacción entre las distintas funciones dentro del sistema y los actores. Cada proceso o función, queda definido dentro de un caso de uso. A continuación presentaremos los distintos casos de uso:

La figura **3.1** describe los casos de uso que se definieron para el **usuario final**. Cada una de las burbujas representa cada una de las funciones que dicho usuario podrá llevar a cabo.

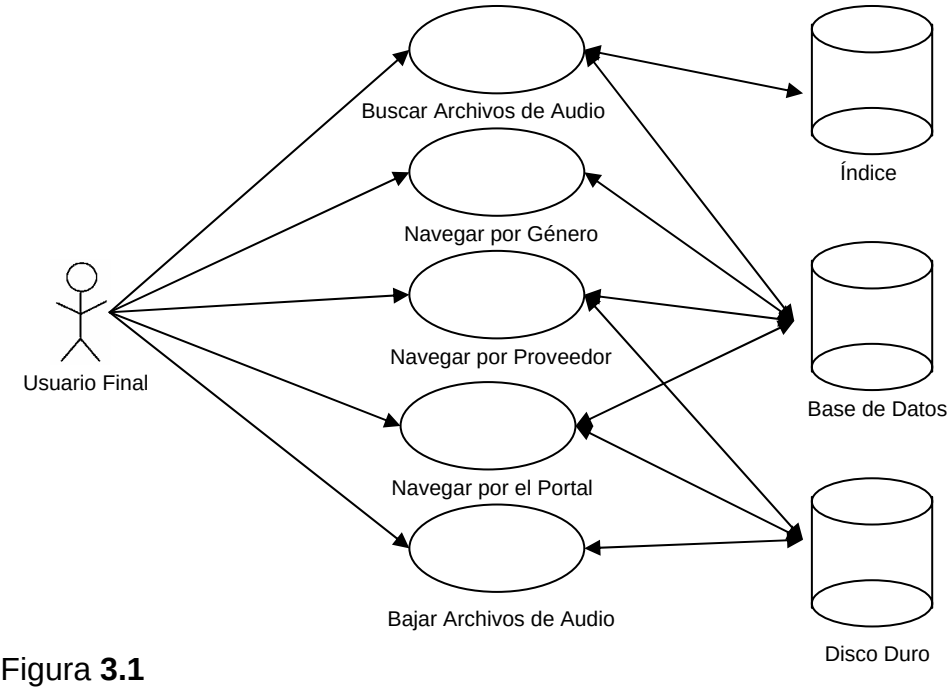

Figura **3.1**
Casos de Uso – **Usuario Final**

Cada uno de los casos de uso del usuario final, es decir, cada una de las burbujas descritas dentro de la figura **3.1**, se encuentran detalladas y analizadas dentro de las **tablas de casos de uso** correspondientes. Las tablas **3.2**, **3.3**, **3.4**, **3.5** y **3.6**, contenidas dentro del **Apéndice B**, son las **tablas de casos de uso** correspondientes para el usuario final.

La figura **3.7**, describe los casos definidos para el **administrador**. Igualmente, cada burbuja representada en dicha figura, representa cada una de las funciones que dicho usuario será capaz de realizar. Las tablas **3.8**, **3.9**, **3.10** y **3.11**, contenidas dentro del **Apéndice B**, describen detalladamente cada caso de uso.

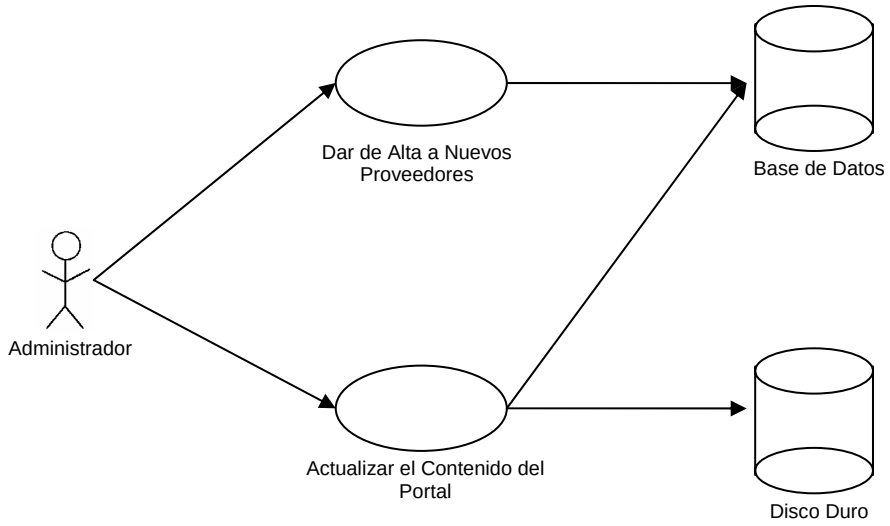

Figura **3.7**
Casos de Uso – **Administrador**

Finalmente, la figura **3.12**, describe los casos de uso para el **proveedor**. Cada uno de sus casos de uso, son analizados y definidos en las tablas **3.13**, **3.14**, **3.15** y **3.16**. Dichas tablas se encuentran contenidas dentro del **Apéndice B**.

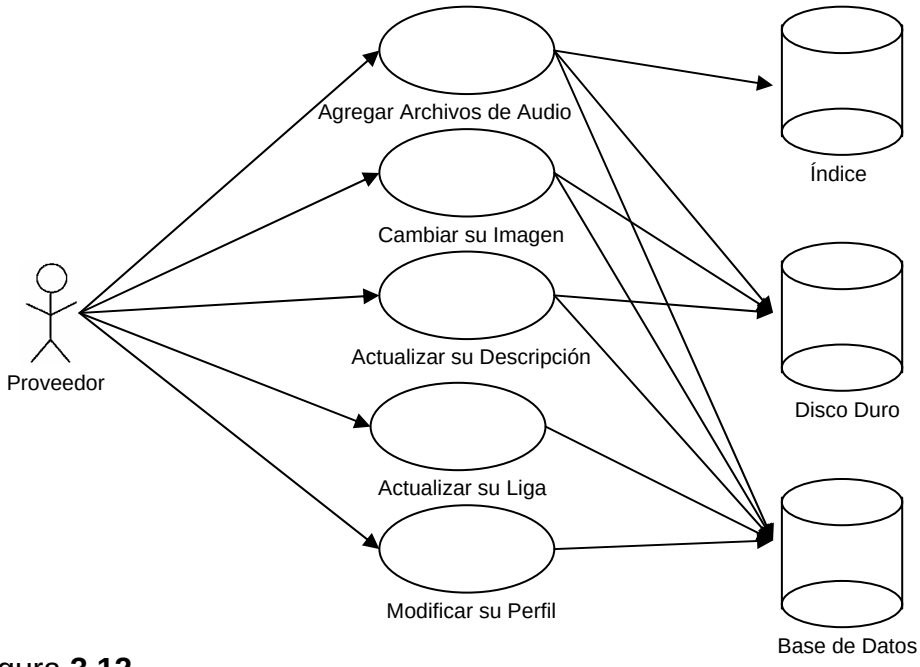

Figura **3.12**
Casos de Uso – **Proveedor**

3.2.2 Diagramas de Flujo de Datos

Una de las partes fundamentales del análisis de requerimientos, es el desarrollo de los diagramas de flujo de datos. Dichos diagramas nos ayudan a definir la manera en la que los datos fluirán dentro del sistema. A demás, nos ayuda a definir el sistema en procesos y ya no verlo como una gran "caja negra". El primer diagrama de flujo de datos (figura **3.17** contenida en el **Apéndice B**) se refiere al nivel contextual del sistema, es decir, tenemos a todos los actores dentro del diagrama del sistema, y todos los procesos se definen en un primer nivel como "caja negra". A partir de éste nivel, comenzamos a profundizar dependiendo el grado de complejidad que cada uno de los procesos presente. Gracias a estos diagramas, podemos comenzar a obtener un pseudocódigo muy primitivo del funcionamiento general del sistema. Las figuras **3.17**, **3.18**, **3.19**, **3.20**, **3.21**, **3.22**, **3.23** y **3.24**, contenidas dentro del **Apéndice B**, contienen los diagramas de flujo de datos para el sistema completo. Estudiando dichos diagramas lograremos un entendimiento más específico y detallado del sistema.

3.3 Análisis del Diseño de Software

En este apartado, estaremos analizando profundamente el diseño del sistema. Una vez que las funciones, los actores y la manera en que se comunican han quedado definidas, podemos comenzar a pensar en el diseño del sistema. A diferencia del análisis de requerimientos, en este apartado contestaremos la pregunta: ¿cómo es que el sistema lo hace? En el análisis de requerimientos nos enfocamos en la pregunta ¿qué?, y ahora nos enfocaremos en el ¿cómo? Para esto, nos ayudaremos de distintos diagramas *UML* así como de diferentes técnicas las cuales nos ayudan a definir el diseño de dicho sistema. Recordemos que una de las herramientas más importantes que utilizaremos para diseñar el sistema, es el patrón de diseño *MVC* descrito a detalle en el **Capítulo II**.

3.3.1 Arquitectura General del Sistema

En esta parte del diseño, nos enfocamos a desarrollar un bosquejo general de las diferentes partes que conforman éste sistema y la manera en que dichos actores se encuentran acomodadas dentro del mismo. Literalmente, definimos la arquitectura del sistema. La figura **3.25**, nos describe gráficamente dicha arquitectura. Como podemos observar, tenemos a los tres tipos de usuarios los cuales se comunican con el sistema a través de un navegador web. Dicho navegador se comunica directamente con el servidor web el cual alberga al contenedor de *servlets* y *JSPs*. En nuestro sistema, el software *Tomcat*, uno de los proyectos de la fundación de software tipo *open source* (código abierto) *Apache* (http://www.apache.org/), se encargará de dicha tarea; será a la vez nuestro servidor web y nuestro contenedor de *servlets* y *JSPs*.

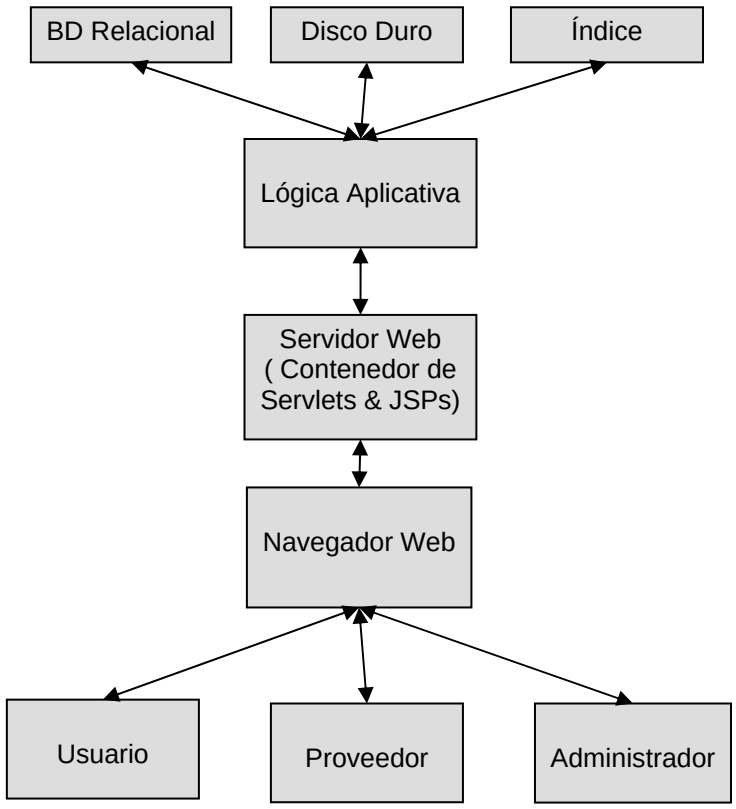

Figura **3.25**
Arquitectura del Sistema

Atrás del servidor web y del contenedor, contamos con la parte más compleja del sistema: la lógica aplicativa. Es justo en esta parte donde se encuentran todos los algoritmos y los procesos principales del sistema. Esta es la parte responsable por todos los cambios de estado dentro del mismo. Evidentemente esta es la parte que representa el Modelo dentro del patrón *MVC*.

Finalmente, detrás de la parte que alberga a la lógica aplicativa, tenemos tres de los actores principales: la base de datos relacional, el disco duro y el índice. Son justamente ellos los que se encargarán de la persistencia de los datos dentro del sistema.

3.3.2 Base de Datos Relacional

Aquí es donde definiremos nuestra base de datos relacional. Recordemos que la base de datos en la que se encargará de almacenar la toda la información de cada uno de los archivos de audio MP3 que se encuentren dentro del sistema. La base de datos también se encargará de almacenar la información de cada uno de los usuarios registrados (administrador y proveedores) dentro del mismo. Finalmente, dentro de dicha base de datos también almacenaremos toda la información referente a los diferentes artículos que se presenten en el portal. La figura **3.26** describe el esquema **Entidad-Relación** para nuestra base de datos relacional.

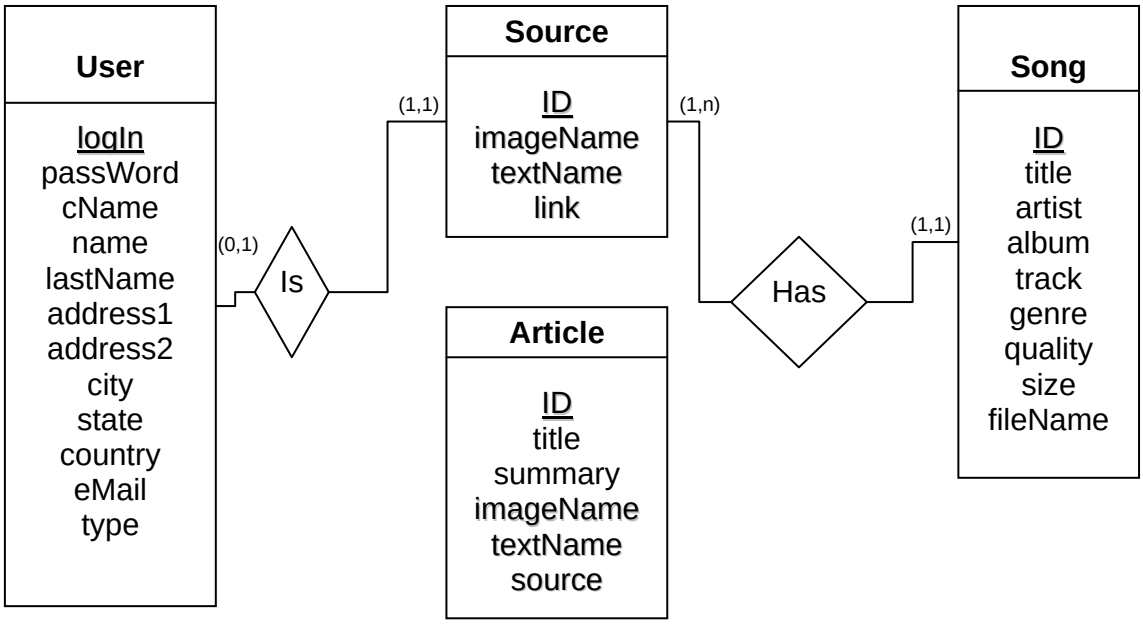

Figura **3.26**
Esquema **Entidad-Relación**

Se utilizaron palabras en inglés para definir dichas entidades y sus atributos ya que el sistema encargado de la gestión de la base de datos (*MySQL*) para nuestro sistema, fue definido originalmente en inglés; y deseamos evitar cualquier conflicto con caracteres especiales como acentos, diéresis, letras 'ñ', etc. De esta manera, a la hora de insertar las tablas dentro de la base de datos, los nombres de las entidades y sus atributos permanecerán igual que en el diseño.

Comencemos por analizar la entidad **Usuario** la cual está descrita dentro del esquema Entidad-Relación como *User*. Cada uno de los usuario registrados dentro del sistema contará con los siguientes atributos: un nombre de usuario, una contraseña, una compañía para la que trabaja, el nombre de la persona con la que se contacta a dicha compañía, su apellido, la dirección de la compañía completa de la compañía, el país donde se encuentra dicha compañía, el correo electrónico del contacto y el tipo de usuario. Recordemos que solamente se tendrán dos tipos de usuario registrados: el administrador y los proveedores. Para nuestro sistema solamente tendremos un administrador registrado pero tendremos varios proveedores registrados. Dichos proveedores son las compañías disqueras o los artistas mismos los cuales son los encargados de proveer todos los archivos de audio MP3 del sistema. La entidad **Usuario**, tiene una relación tipo "es un" con la entidad **Proveedor** (*Source*). Dicha

entidad nos describe la información de los distintos proveedores que se encuentren registrados dentro del sistema. Dicha entidad cuenta con los siguientes atributos: un número identificador, el nombre del archivo de la imagen de su descripción, el nombre del archivo de texto en donde se encuentra guardada su descripción y la liga a su página web. Los archivos de las imágenes de las descripciones de los proveedores serán almacenados en el disco duro para facilitar su manejo. La base de datos solamente almacena el nombre de dichos archivos. Igualmente, el texto de la descripción de dichos proveedores se guardará dentro de archivos de texto tipo ".txt" dentro del disco duro. Esto, ya que el campo de texto de nuestra base de datos, admite solamente hasta 255 caracteres. Muchas de las descripciones de nuestros proveedores cuentan con más de 255 caracteres. La relación entre la entidad **Usuario** y la entidad **Proveedor** es de **cero a uno** del lado del **Usuario** ya que cualquier usuario puede o no ser un proveedor. El administrador es un usuario pero no es un proveedor. La restricción es que cada usuario solamente puede ser considerado como proveedor una sola vez, es decir, un usuario no puede estar registrado como dos proveedores al mismo tiempo. Del lado del **Proveedor** la relación es de **uno a uno**. Un proveedor es un y solamente un usuario.

Después contamos con la entidad **Canción** la cual está descrita en el esquema entidad-relación bajo el nombre de *Song*. Dicha entidad describe a los archivos de audio que se encuentran dentro del sistema, y cuenta con los siguientes atributos: un número identificador, el título de la canción, el artista que la interpreta, el álbum al que pertenece, el número de pista, el género, la calidad, el tamaño del archivo y el nombre del archivo. Los archivos son almacenados en el disco duro y no dentro de la base de datos, la base de datos solamente almacena el nombre del archivo correspondiente a dicha canción. Para establecer la calidad de dicho archivo nos basamos en el *bitrate* de dicho archivo y lo calificamos con respecto a la tabla **3.27**. Evidentemente mientras mayor sea el valor del *bitrate*, mayor será la calidad del archivo.

La entidad **Proveedor** tiene una relación tipo "tiene" con la entidad **Canción**. Dicha relación es de **uno a muchos** del lado del **Proveedor** ya que un proveedor puede tener desde una hasta muchas canciones registradas a su nombre. Del lado de la entidad **Canción**, la relación es de **uno a uno** ya que cada canción solamente puede pertenecer a un solo proveedor.

Tabla **3.27**. Índice de calificación de la calidad de los archivos de audio con respecto al *bitrate*.

Bitrate (Kbps)	Calidad
< 64	1
64 ú 80	2
93	3
112	4
≥ 128	5

Por último, contamos con la entidad **Artículo**. Esta entidad describe todos los atributos de los artículos que se presentan dentro del portal. Dicha entidad está descrita dentro de la figura **3.26** bajo el nombre *Article*. Esta entidad se encuentra aislada de las demás ya que no tienen relación ninguna con el resto de las entidades. La entidad **Artículo** cuenta con los siguientes atributos: un número identificador, un título, un resumen, el nombre del archivo de la imagen que irá con dicho artículo, el nombre del archivo de texto donde se encuentra el artículo entero y el nombre de la fuente de donde se obtuvo el artículo. El texto completo de los artículos se guarda en archivos de texto dentro del disco duro por la misma razón que las descripciones de los proveedores. Se almacena un pequeño resumen del artículo ya que cuando el usuario accede a la página que presenta todos los artículos, solamente se le presentará un resumen de cada uno de ellos para no saturar la pantalla con información. Será opción del usuario elegir uno de los artículos y entonces se le presentará el texto completo de dicho artículo.

La figura **3.28** nos muestra las tablas resultantes a partir de transformar el esquema **Entidad-Relación** en tablas. Dichas tablas se encuentran codificadas dentro de la base de datos. Para este sistema, el software que se encarga de la gestión de la base de datos es el sistema tipo *open source* (código abierto) *MySQL* (http://www.mysql.com/).

User

logIn	passW	cName	name	lastName	address1	address2	city	state	country	eMail	type

Source

ID	imageName	textName	link	IdUser

Song

ID	title	artist	album	track	genre	quality	size	location	IdSource

Article

ID	title	summary	imageName	textName	source

Figura **3.28**
Tablas Resultantes del Esquema **Entidad-Relación**

3.3.3 Diagramas de Secuencias

Para el diseño del sistema, nos apoyamos en el lenguaje de diseño *UML*. *UML*, *Unified Modeling Language* por sus siglas en inglés, es un lenguaje de diseño el cual describe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Dicho lenguaje consta de nueve tipos de diagramas distintos los cuales nos ayudan a modelar y a diseñar cualquier sistema orientado a objetos. Uno de los diagramas descritos por *UML* es el diagrama de secuencias. Este diagrama consiste de objetos representados como rectángulos, mensajes representados por flechas las cuales comunican a los diferentes objetos, y por el tiempo representado como una progresión vertical. Este diagrama nos ayuda a representar la manera en que los distintos objetos interactúan, pero principalmente, nos ayuda a representar el tiempo dentro de nuestro sistema.

Contamos con un diagrama de secuencias por cada tipo de usuario dentro de nuestro sistema. La figura **3.29** describe el diagrama de secuencias del **administrador**.

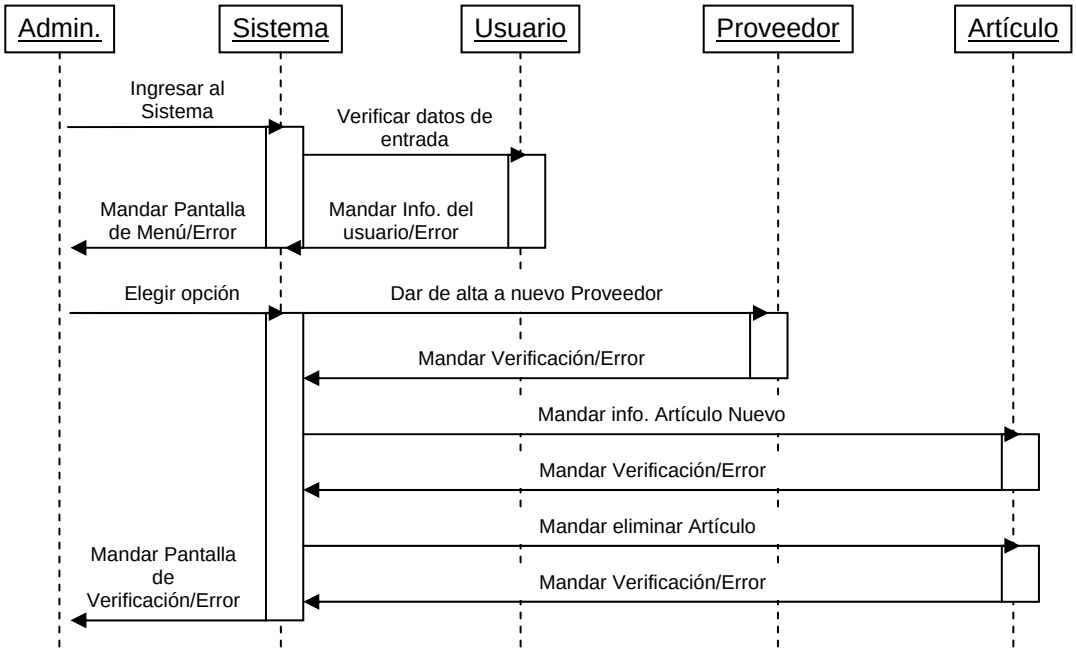

Figura **3.29**
Diagrama de Secuencias – **Administrador**

La figura **3.30** representa el diagrama de secuencias del **proveedor**.

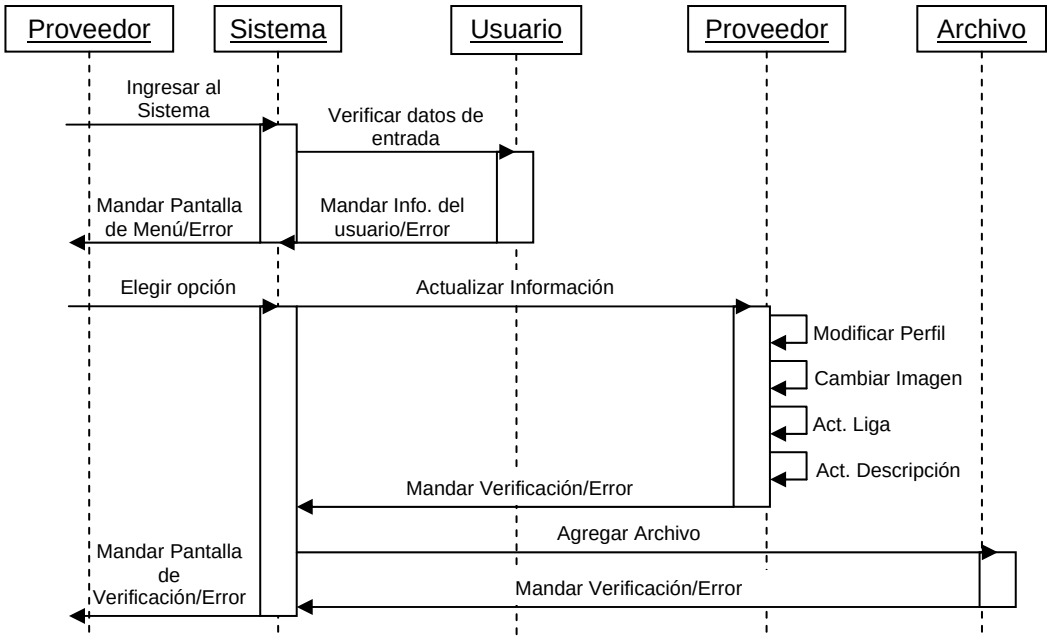

Figura **3.30**
Diagrama de Secuencias – **Proveedor**

La figura **3.31** describe el diagrama de secuencias del **usuario final**.

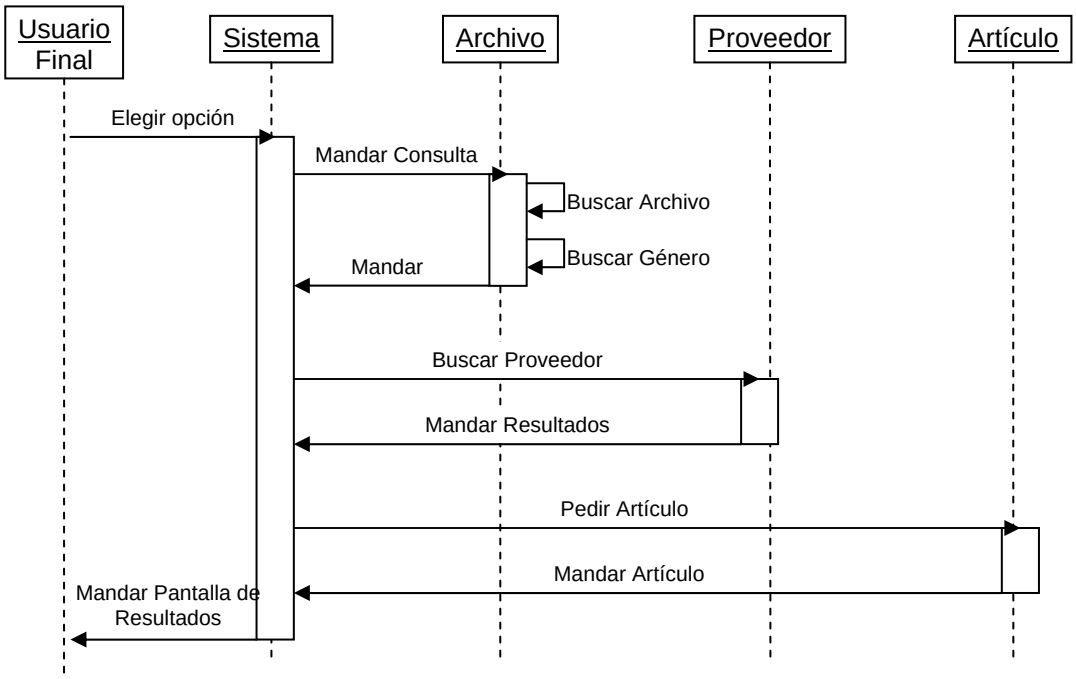

Figura **3.31**
Diagrama de Secuencias – **Usuario Final**

3.3.4 Diagramas de Colaboraciones

Los diagramas de colaboraciones son otro de los nueve tipos de diagramas descritos por el lenguaje *UML*. Este tipo de diagrama nos ayuda a describir la relación entre los objetos dentro de nuestro sistema. El diagrama de colaboraciones también nos ayuda a definir con más claridad cuales son los mensajes que serán enviados de un objeto a otro. Esto es muy útil ya que nos ayuda a definir con más detalle la comunicación interna entre los diferentes objetos protagonistas de nuestro sistema.
La figura **3.32** representa el diagrama de colaboraciones para el **administrador**.

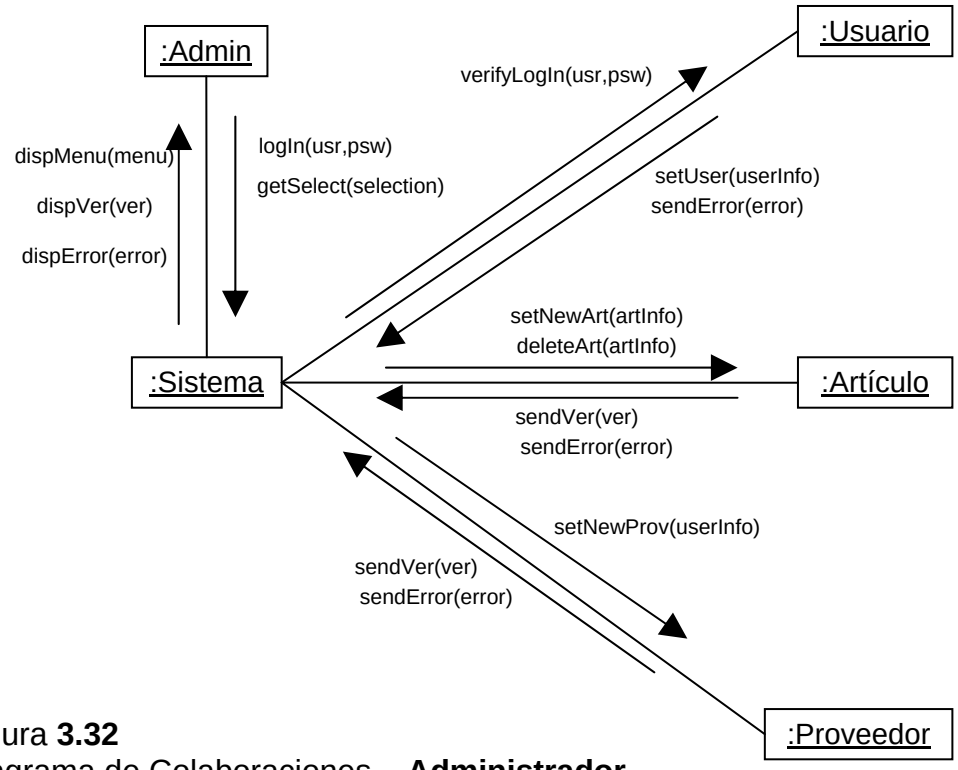

Figura **3.32**
Diagrama de Colaboraciones – **Administrador**

La figura **3.33** describe el diagrama de de colaboraciones para el **proveedor**.

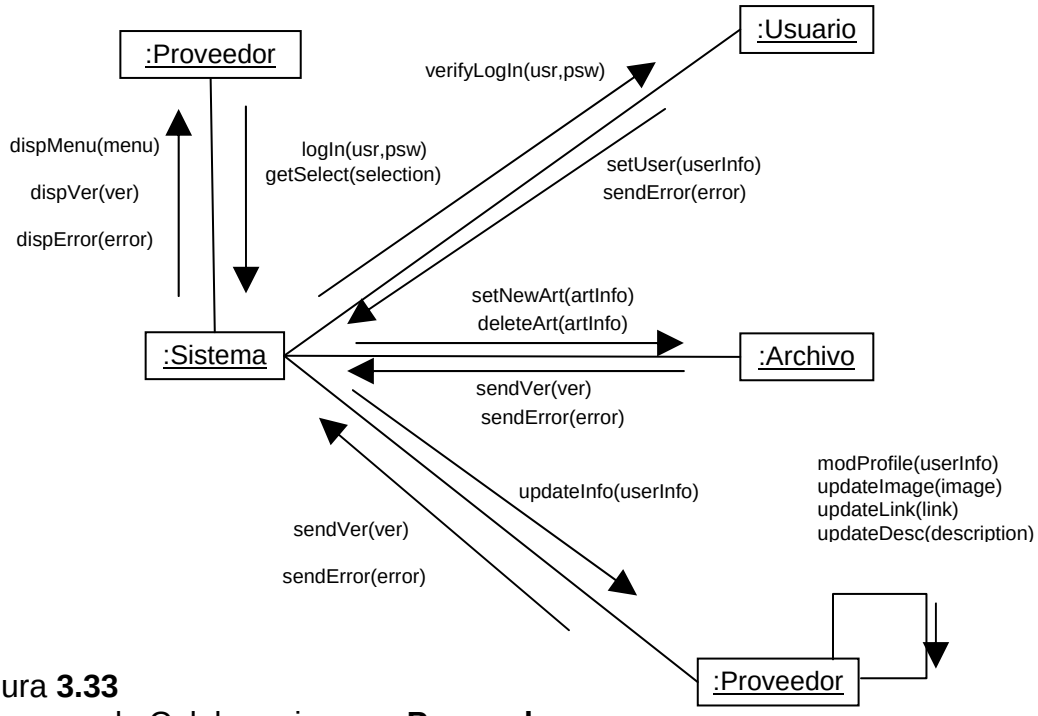

Figura **3.33**
Diagrama de Colaboraciones – **Proveedor**

La figura **3.34** representa el diagrama de colaboraciones para el **usuario final**.

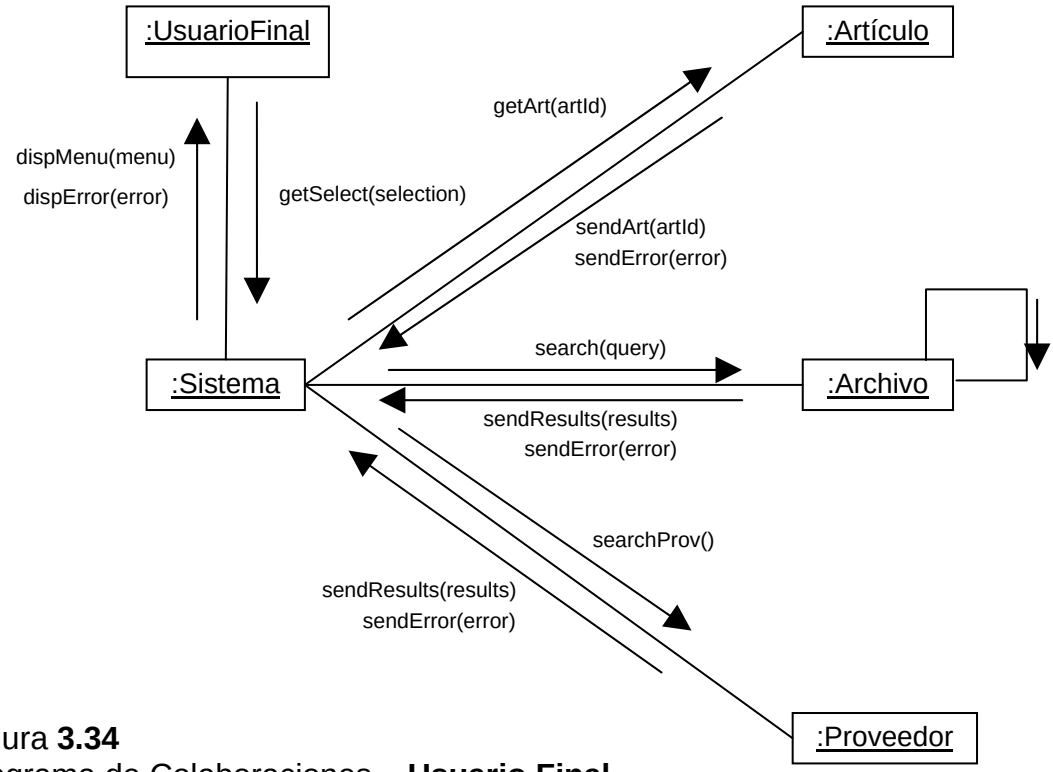

Figura **3.34**
Diagrama de Colaboraciones – **Usuario Final**

3.3.5 Diagramas de Clases

El diagrama de clases, también es otro tipo de diagrama dentro del lenguaje *UML*. Este nos ayuda a diseñar cada uno de los objetos que conforman nuestro sistema. Dichos objetos, representados por rectángulos, incluyen el nombre de la clase (del objeto) y los métodos que dicha clase alberga. Cada método representa una función que dicha clase puede realizar. Las figuras **3.35** a **3.59**, contenidas dentro del **Apéndice B**, representan los diagramas de clases para todo el sistema. Dichas figuras describen la arquitectura completa de los objetos dentro de nuestro sistema.

3.3.6 Índice de Archivos de Audio

Como mencionamos anteriormente, se construyó un índice sobre el cual se realizarán las búsquedas de los archivos de audio MP3. Dicho índice debe de estar cargado en memoria principal todo el tiempo que el sistema esté corriendo para agilizar las búsquedas que realicen los usuarios finales. Dicho índice está conformado por una colección de documentos *XML*. Cada uno de los documentos describe los atributos principales de cada uno de los archivos de audio registrados en el sistema. Los atributos etiquetados en los documentos *XML* son los atributos de las canciones sobre los cuales se realizan las búsquedas. Los atributos son: el título de la canción, el artista que la interpreta, el álbum al que pertenece, el género y la calidad del archivo de audio. Todos los documentos *XML* dentro del sistema que describen a los archivos de audio, cuentan con los mismos atributos y la misma estructura. La estructura de los documentos *XML* es la siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<audioFile id="535">
  <title>Time</title>
  <artist>Pink Floyd</artist>
  <album>Dark Side of the Moon</album>
  <genre>Psychedelic Rock</genre>
  <quality>five</quality>
</audioFile>
```

Para mostrar la estructura de los documentos *XML* se tomó como ejemplo una canción titulada *Time* interpretada por el grupo *Pink Floyd* la cual es parte del album *Dark Side of the Moon*. Dicha canción se encuentra clasificada dentro del género *Psychedelic Rock* y cuenta con una calidad de cinco, es decir, la calidad más alta. El atributo `id="535"` de la etiqueta `<audioFile>`, representa el número identificador de dicho archivo de audio. Ese es el mismo número de identificación del archivo dentro de la base de datos. Existe un documento *XML* para cada uno de los archivos de audio MP3 dentro del sistema. Cada vez que un archivo de audio nuevo es agregado al sistema, primero se construye el archivo *XML* de dicho archivo de audio. Una vez que el documento *XML* se

encuentra almacenado en el disco duro, dicho documento es agregado a la colección. La colección completa de documentos *XML* es lo que forma el índice de nuestro sistema.

El software *eXist* (http://exist.sourceforge.net) es el encargado de cargar dicho índice a memoria principal y de administrarlo. Nuestro sistema agrega automáticamente los documentos *XML* a la colección administrada por dicho software. Una vez dentro de la colección, dicho documento queda accesible para poder realizar consultas. Evidentemente, las consultas se realizan sobre la colección completa a través de la interfase de comunicación del software *eXist*.

Para poder realizar las búsquedas sobre el índice de documentos *XML*, utilizamos el lenguaje *XPath* (http://www.w3.org/TR/xpath). Dicho lenguaje es parte una especificación del consorcio *W3C* (http://www.w3.org) la cual fue diseñada para simplificar la localización de elementos dentro de los documentos *XML*. En realidad es un lenguaje muy sencillo para realizar consultas dentro de documentos *XML*. La interfase de comunicación del software *eXist*, recibe consultas en el formato del lenguaje *XPath*. Para explicar la sintaxis y el funcionamiento general de dicho lenguaje, tomemos como ejemplo el mismo documento *XML* que utilizamos anteriormente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<audioFile id="535">
  <title>Time</title>
  <artist>Pink Floyd</artist>
  <album>Dark Side of the Moon</album>
  <genre>Psychedelic Rock</genre>
  <quality>five</quality>
</audioFile>
```

Los documentos *XML* se definen por etiquetas las cuales pueden ser hijas o padres de otras etiquetas. Por ejemplo, la etiqueta `<title>` es hija de la etiqueta `<audioFile>`. *XPath* se enfoca en dicha arquitectura (padre-hijo) para simplificar las búsquedas. Tomemos como ejemplo la siguiente expresión escrita en *XPath*:

```
/audioFile/artist
```

Si introducimos tal expresión como una búsqueda, lo que obtendremos serán los datos contenidos dentro de la etiqueta `<artist>` y todas las etiquetas hijas de dicha etiqueta con sus valores para cada uno de los documentos contenidos dentro de la colección. Evidentemente si introducimos la siguiente expresión:

```
/audioFile
```

obtendremos los datos contenidos dentro de la etiqueta `<audioFile>`, así como todas las etiquetas hijas de dicha etiqueta; en este caso todo el documento.
También podemos realizar búsquedas tipo texto abierto utilizando la función `contains()`. Por ejemplo, si deseamos obtener toda la información del documento en el que el nombre de la canción es *Time*, introduciríamos la siguiente línea de código:

```
/audioFile[title[contains(.,"time")]]
```

Tal expresión nos regresaría los datos contenidos dentro de la etiqueta `<audioFile>` así como todas las etiquetas hijas, con sus valores, contenidas dentro del documento que describe una canción la cual tiene como título o parte de él la palabra *Time*. La función `contains()` evalúa si la cadena de caracteres es subconjunto del valor contenido dentro de la etiqueta especificada, en este caso dentro de la etiqueta que describe el título de la canción.
También podemos realizar búsquedas sobre algún atributo de cualquier etiqueta utilizando el carácter '@' para especificarlo. Tomemos la siguiente expresión como ejemplo:

```
/audioFile[@id=""535]
```

Lo que obtendríamos de dicha expresión, serían todos los hijos y los valores de la etiqueta `<audioFile>` pertenecientes al documento que tenga como valor de la etiqueta `id`, el número 535.
Existen otras funciones dentro del lenguaje *XPath*, las cuales no mencionaremos ni explicaremos, con las cuales podemos realizar un sin fin de combinaciones para poder formular cualquier consulta que deseemos.

3.4 Algorítmica del Buscador de Archivos de Audio

En este apartado, explicaremos a detalle lo que quizá es la parte más importante de nuestro trabajo de tesis: la manera en la que el buscador de archivos de audio de nuestro sistema funciona. Es uno de los fines principales de nuestra tesis, construir un buscador de archivos de audio con el cual obtengamos resultados coherentes y reales con respecto a cualquier consulta. No nos interesa saber simplemente si lo que estamos buscando se encuentra dentro del sistema o no, es decir, no solamente estamos buscando una respuesta binaria del sistema. Quizá el usuario introduzca una consulta la cual no coincida exactamente con ninguno de los documentos dentro del sistema. Pero quizá dicha consulta sea una subcadena de alguna palabra contenida dentro de alguno de los documentos del sistema y es de nuestro interés encontrar dichas similitudes o subcadenas de tal manera que nuestro sistema no solamente le dice al usuario si lo que está buscando se encuentra o no, sino que el sistema también puede regresar algunos resultados los cuales tengan un grado de relevancia con lo que el usuario esta buscando. Por ejemplo, si el usuario introduce la palabra *Pink* como su consulta, quizá dentro del sistema no haya ningún documento el cual contenga alguna etiqueta dentro de la cual exista la palabra *Pink* exactamente, pero el sistema se percatará de que la palabra *Pink* es una subcadena de la cadena de caracteres *Pink Floyd*. De tal manera que el sistema le regresará al usuario no solamente los resultados que coincidan exactamente con la palabra *Pink*, sino también regresará los documentos los cuales dentro de sus etiquetas contengan la subcadena *Pink*. Esta quizá es la característica más importante de nuestro buscador.

Nos interesa también conocer el grado de relevancia de cada uno de los resultados generados. Dichos resultados son ordenados descendentemente con respecto al grado de relevancia que cada uno de ellos tiene en comparación con la consulta original, es decir, dentro de la lista de resultados generados por el sistema, primero encontraremos aquellos resultados que coincidan exactamente con la consulta, si es que los hay. Después encontraremos el resto de los resultados los cuales talvez no coincidan exactamente con la consulta, pero que cuentan con un cierto grado de relevancia.

Lo primero es recibir la consulta del usuario final. Dicha consulta es introducida dentro de un campo de texto. A la vez, el usuario puede definir el tipo de búsqueda que desea realizar. Existen cuatro tipos de búsquedas: búsqueda general, búsqueda sobre el título de una canción, búsqueda sobre el nombre de un artista, y búsqueda sobre el nombre del álbum. El usuario define el tipo de búsqueda, introduce la consulta a manera de texto y el sistema se encarga de lo demás.

La interfase `HttpServletRequest` perteneciente al paquete `javax.servlet.http` es la que se encarga de recibir dichos parámetros. Una vez recibidos son traducidos en una expresión del lenguaje *XPath* la cual el software *eXist* pueda interpretar. Recordemos que todas las búsquedas se realizan sobre el índice y no sobre la base de datos relacional. El software *eXist* es el encargado de administrar el índice del sistema. *eXist* recibe expresiones del lenguaje de consulta *XPath* para realizar las consultas dentro del índice. Nuestro sistema utiliza los parámetros introducidos por el usuario (la consulta y el tipo de consulta) para construir la expresión *XPath* con la cual realizaremos la búsqueda sobre el índice. Evidentemente al existir cuatro tipos de consultas, contamos con cuatro diferentes expresiones *XPath* para realizar las búsquedas:

- `/audioFile[contains(.,"query")]` – Realiza la búsqueda de la palabra `query` sobre el contenido de todas las etiquetas del documento.

- `/audioFile[artist[contains(.,"query")]]` – Realiza la búsqueda de la palabra `query` sobre el contenido de la etiqueta que define el nombre del artista.

- `/audioFile[title[contains(.,"query")]]` – Realiza la búsqueda de la palabra `query` sobre el contenido de la etiqueta que define el título de la canción.

- `/audioFile[album[contains(.,"query")]]` – Realiza la búsqueda de la palabra `query` sobre el contenido de la etiqueta que define el nombre del álbum.

La función `contains()`, perteneciente a la especificación *XPath*, es la encargada de comparar la consulta con el contenido de las etiquetas. Dicha función recibe dos parámetros (dos cadenas de caracteres), y evalúa si la segunda cadena de caracteres se encuentra contenida dentro de la primera, de ser así, dicha función nos regresa un valor boleano verdadero; de lo contrario nos regresa un valor boleano falso.

Como se mencionó anteriormente, todas las búsquedas se realizan sobre las etiquetas que definen los atributos de cada uno de los archivos de audio. Dichas etiquetas se encuentran definidas dentro de cada uno de los documentos *XML* generados por el sistema. Cada archivo de audio cuenta con un documento *XML* el cual define sus atributos. La colección de todos los documentos *XML* forma el índice sobre el cual realizamos las consultas. Dependiendo el tipo de búsqueda que el usuario haya definido, es el tipo de búsqueda que se realiza, es decir, si el usuario define el tipo de búsqueda por artista, entonces la consulta solamente se realiza dentro del contenido de la etiqueta que define el nombre del artista. Si el usuario define el tipo de búsqueda como general, entonces la consulta se realiza dentro del contenido de todas las etiquetas de cada uno de los documentos contenidos dentro del índice. Recordemos que la estructura de los documentos *XML* es la siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<audioFile id="535">
  <title>Time</title>
  <artist>Pink Floyd</artist>
  <album>Dark Side of the Moon</album>
  <genre>Psychedelic Rock</genre>
  <quality>five</quality>
</audioFile>
```

Una vez realizada la búsqueda, el software *eXist* nos regresa una lista con todos los documentos *XML* pertenecientes a los resultados. Dicha lista contiene cada uno de los documentos en el formato `org.w3c.dom.Element`. Posteriormente, cada documento es transformado en el formato `org.jdom.Element` para facilitar su manejo. Una vez transformados, el sistema obtiene el atributo `id` perteneciente a la

etiqueta `<audioFile>` para entonces obtener, de la base de datos relacional, todos los atributos de dicho archivo de audio. Recordemos que el número de identificación de cada archivo de audio es el mismo dentro del índice y dentro de la base de datos relacional. De esta manera agilizamos las búsquedas y la obtención de los atributos de cada archivo de audio.

Una vez obtenidos los atributos, el sistema entonces ordena los resultados de manera descendente con respecto al grado de relevancia que cada archivo de audio tiene en comparación con la consulta introducida por el usuario. Para realizar dicha tarea, se diseñó un algoritmo el cual en realidad es una variante del modelo de espacios vectoriales el cual es una de las principales técnicas utilizadas hoy en día para la recuperación eficiente de información. Dicho modelo fue descrito en la sección **2.3.4** del **Capítulo II**. Lo consideramos una variante ya que nuestro algoritmo parte de la idea principal del método de espacios vectoriales, pero sin considerar los pesos asignados por la frecuencia que presenta cada uno de los términos dentro del documento. Esto, ya que los términos contenidos dentro de cada una de las etiquetas de los documentos *XML* nunca se repetirán, y por lo tanto cada término contará con la misma frecuencia para cada documento dentro de la colección. De tal manera que no podremos distinguir el grado de relevancia de cada término en comparación con el resto de la colección. Por ejemplo, si un documento *XML* describe una canción interpretada por el artista *Bob Dylan*, dichas palabras no se repetirán jamás dentro de la etiqueta que describe el nombre del artista y por lo tanto, cada una de las palabras que describe el nombre del artista contará con una frecuencia de uno; y cada documento que describa una canción interpretada por el mismo artista, contará con el mismo valor de frecuencia. Hacemos esta distinción para los valores contenidos dentro de cada etiqueta por separado ya que recordemos que las búsquedas se realizan sobre el contenido de cada etiqueta por separado y no sobre el contenido completo del documento. Esto surge a partir de que el modelo de espacios vectoriales fue diseñado para recuperar información de documentos sobre los cuales se realizan búsquedas de texto completo. Nuestro sistema realiza búsquedas sobre atributos bien definidos los cuales son descritos por pocas palabras las cuales no se repiten, es decir, nuestro sistema realiza búsquedas sobre etiquetas las cuales describen datos muy específicos. Por ejemplo, nuestro sistema busca el nombre de un artista dentro del contenido de una etiqueta que describe exactamente eso, y no sobre un texto completo (un artículo por ejemplo), dentro del cual el nombre del artista pueda ser mencionado más de una vez.

Entonces, ¿cómo es que nuestro sistema resuelve dicho problema? Lo que nuestro sistema evalúa, es un porcentaje el cual representa la cantidad de palabras de la consulta original, que coinciden con cada uno de los resultados. Se toma la lista completa de resultados generados con sus atributos, y dependiendo del tipo de búsqueda definido por el usuario, es el atributo sobre el cual se obtendrá dicho porcentaje. Por ejemplo, si el usuario define el tipo de búsqueda sobre el título de una canción, entonces para cada uno de los resultados generados, se obtiene el título de la canción el cual es separado palabra por palabra. Cada una de las palabras contenidas dentro del título de la canción es comparada con cada una de las palabras que conforman la consulta original y se cuentan el número de palabras que coinciden exactamente. Dicho número es entonces comparado con el número total de palabras contenidas dentro del título de la canción y de esa manera se obtiene un porcentaje el cual representa el grado de relevancia que cada uno de los resultados tiene en comparación con la consulta original. Por ejemplo, si la consulta cuenta con dos palabras y el título de una de las canciones de los resultados cuenta con cuatro palabras, y las dos palabras de la consulta coinciden con dos de las palabras del título de la canción, entonces dicho archivo de audio tendrá un grado de relevancia del 50%. Por el contrario, si comparamos la misma consulta de dos palabras con otro de los resultados que cuente con dos palabras, y ambas coinciden, entonces dicho archivo contará con un 100% de grado de relevancia.
Este mismo algoritmo se aplica para cualquier tipo de búsqueda, evidentemente si el tipo de búsqueda es general, entonces el número de palabras por comparar será mayor pero el algoritmo no cambiará.

Una vez ordenados los resultados, son presentados al usuario con todos sus atributos para que el usuario escoja el archivo de audio que desee bajar.

3.5 Hardware

En esta sección, especificaremos el hardware sobre el cual el sistema corre. También daremos algunas recomendaciones de hardware las cuales son necesarias para que el sistema funcione correctamente.
Escoger el hardware correcto es indispensable para el buen funcionamiento del sistema descrito dentro de este documento de tesis. Dicho hardware si es un factor importante el cual puede llegar a determinar la estabilidad y la velocidad de dicho sistema. Todo el sistema fue programado y corre en una sola computadora personal, dicha computadora cuenta con el siguiente hardware:

- Procesador Mobile Intel(R) Pentium(R) 4 a 3.06 GHz.
- 512 MB de memoria RAM.
- 60 GB en disco duro.
- Tarjeta de red LAN 10/100

Evidentemente este no es el hardware mínimo o crítico para que el sistema funcione correctamente, solamente es mencionado ya que es el hardware que fue utilizado para programar y correr el sistema. Consideramos, para el correcto funcionamiento del sistema, el siguiente hardware:

- Procesador de 1 GHz.
- 256 MB de memoria RAM.
- 20 GB en disco duro.
- Tarjeta de red LAN 10/100

Es necesario contar con suficiente espacio en el disco duro ya que en promedio cada archivo de audio MP3 mide alrededor de 4 MB. Evidentemente mientras más archivos MP3 se encuentren dentro del sistema, mayor será el espacio necesario dentro del disco duro.

En cuento a memoria principal, se consideró un mínimo de 256 MB ya que el sistema utiliza varios programas los cuales necesitan estar cargados en memoria principal para su funcionamiento. También debemos de tomar en cuenta que mientras más archivos de audio MP3 se encuentren registrados dentro del sistema, mayor será el tamaño del índice sobre el cual se realizan las búsquedas, y dicho índice se encuentra cargado en memoria principal. Favorablemente, el software encargado de la administración del índice, realiza un excelente trabajo en la administración de los recursos de memoria.

Al igual que el índice, la base de datos relacional crece directamente proporcional a la cantidad de archivos MP3 registrados dentro del sistema, y dicha base de datos también necesita estar cargada en memoria principal.

En cuanto al procesador, es necesario contar con un procesador de alta velocidad, ya que el sistema utiliza algunos algoritmos complejos los cuales pueden ser considerados como “pesados” para el procesador. La velocidad del procesador es crítica para determinar el tiempo de corrida del sistema.

3.6 Software

En este apartado mencionaremos el software que fue utilizado para la programación del sistema, para la generación del documento escrito de tesis, para la generación de tablas y diagramas, y para el funcionamiento del sistema. El software utilizado es el siguiente:

- Sistema Operativo Windows XP Home Edition
- JCreator (Editor de Java)
- J2SDK

- Apache Tomcat (Servidor Web y Servlet Container)
- eXist (Administrador de índices XML)
- MySQL (Sistema de Gestión de Base de Datos Relacional)
- Microsoft FrontPage (Editor de HTML)
- Microsoft PowerPoint (Editor de Gráficas y Figuras)
- Macromedia Dreamweaver MX (Editor de HTML, JSP y XML)
- Adobe Photoshop (Editor de Imágenes)
- JUDE Community (Herramienta CASE)
- mp3ID3Handler.jar (Paquete tipo código abierto escrito en Java para manipular archivos de audio MP3)
- commons-fileupload-1.0.jar (Paquete tipo código abierto escrito en Java para subir archivos a través del protocolo HTTP)
- jdom.jar (Paquete tipo código abierto escrito en Java para manipular documentos XML)