Universidad de las Ciencias Informáticas

Facultad1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Subsistema de búsqueda de música para la plataforma c.u.b.a.

Autor:

Antonio Miguel Abreu Jovelar.

Tutores:

MSc. Yanedi Abreu Bartomeo.

Ing. Daneysi Hernández García.

Co-Tutores:

Ing. Angélica Váquez Reinoso.

Ing. Michel L. Frómeta Burey.

La Habana. Año 60 de la Revolución

Declaración de autoría

Declaro que soy el único autor del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de este, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de ___ del año ____.

Antonio M. Abreu Jovelar

_____

Autor

MsC. Yanedi Abreu Bartomeo

_____

Tutor

Ing. Daneysi Hernández García

_____

Tutor

Dedicatoria

A mi familia. A mi madre por apoyar mis sueños. A mi papá Carlitos por ser mi padre. A mis hermanos.

Agradecimientos

Quiero agradecer en primer lugar a mi mamá Irene, por apoyarme en esta travesía…A mi papá por brindarme su apoyo sin condiciones… A Carlitos por ser mi segundo padre, por estar ahí en todo momento, A mis hermanos Maritza y Marycelis que aún en la distancia están a mi lado, a Rolandito por ayudarme en cuánto pudo. A mis tutores Daneysis, por servirme de guía en este año de tanto estrés, a Yanedi por sus kilométricos comentarios… A Angélica que más que mi tutora es una gran amiga, a Michel por sus locuras. Agradecerles a todos los profesores que me brindaron su apoyo Wendy, Abdel, Abdiel, Dunia, Erick, Yaima… en fin a todos. A Graciela por su inglés y gran paciencia.

"La arcilla fundamental de nuestra obra es la juventud, en ella depositamos nuestra esperanza y la preparamos para tomar de nuestras manos la bandera."

Ernesto Che Guevara

RESUMEN

Para localizar y procesar grandes cúmulos de información de forma rápida y automática en la *web* son utilizados los Sistemas de Recuperación de Información basados en contenidos. Estos son capaces de buscar cualquier contenido existente en la *web* como imágenes, videos y música. Actualmente la plataforma de Contenido Unificado para la Búsqueda Avanzada (c.u.b.a.) entre varios de los servicios que ofrece, garantiza la búsqueda simple y avanzada de imágenes, pero prescinde de una personalización para la búsqueda de archivos musicales en la *web* nacional. Por tal motivo, la presente investigación propone desarrollar el subsistema de búsqueda de música para la plataforma c.u.b.a., guiada por la metodología de desarrollo de *software* AUP-UCI, y compuesta por tres componentes: Aplicación *web*, Rastreador e Indexador. Se seleccionaron como principales tecnologías para su desarrollo: *Nutch* como mecanismo de rastreo de la *web*, *Solr* como componente de indexación, *Symfony* como marco de trabajo PHP, *Java* para la implementación del *plugin* de *Nutch* y *Visual Paradigm* como herramienta para el modelado. El subsistema implementado posee un conjunto de características y funcionalidades que permiten a los usuarios realizar búsquedas simples y avanzadas de archivos musicales publicados en la *web* nacional.

Palabras clave: archivos musicales, búsqueda de música, indexación, rastreo.

ABSTRACT

To locate and to process big heaps of information in a quick and automatic way in the web the Systems of Recovery of Information based on contents are used. These are able to look for any existent content in the web as images, videos and music. However, actually the platform of Contend United for the Advanced Quest (c.u.b.a.) it is focused in the simple and advanced search of images, but it does without of a personalization for the search of musical files in the national web. For such reason, present investigation proposes developing the subsystem of music quest for the platform c.u.b.a., guided under the methodology of development of software AUP-ICU, and composed by three components: Application Web, Tracking and Indexation. Were selected as principal technologies: Nutch as the Web's mechanism of trawling, Solr as component of indexation, Symfony as framework PHP, Java for the implementation of Nutch and Visual Paradigm as tool for the modeling. The implemented subsystem has a set of characteristics and functionalities that contribute to allowing the user to accomplish simple quests and advanced quests of musical files published in the national Web.

***Keywords*:** *musical files, search of music, indexation, tracking.*

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los estudios realizados por el sitio de análisis estadísticos *We Are Social 2018*[1]*,* declara que más de la mitad de la población mundial utiliza *Internet* con más de 4.021 millones de usuarios *online* **(Ver Anexo 1)**. En un contexto más profundo, a cada minuto surgen nuevas necesidades de información por parte del usuario debido al avance de la Tecnologías de la Información y las Comunicaciones (TIC) (Kujawski, 2018). Con el fin de satisfacer dicha necesidad, se comienza a trabajar con una disciplina relacionada con la búsqueda, conocida como "Recuperación de Información" (RI). La RI estudia y propone soluciones planteando modelos, algoritmos y heurísticas en el área de la informática, por lo que constituye una representación del almacenamiento y búsqueda de informaciones en el sistema (Baeza, 1999). El proceso de RI por sí solo no ordena la información, es de vital importancia la disponibilidad de sistemas que auxilien a los usuarios a la hora de buscar lo que necesitan en la *web*. Por ende, cuando se habla de RI se da por hecho que interviene un "Sistema de Recuperación de Información" (SRI). Estos permiten localizar y procesar la información de forma automática; siendo capaces de buscar cualquier contenido existente en la *web*, ya sean imágenes, contenidos de sitios o archivos musicales (Giordanino, 2015).

La difusión de archivos musicales se ha convertido en una de las principales vías de comunicación hoy en día. Desde los discos de pizarra al vinilo hasta los sistemas de recuperación de música en *Internet*, los archivos musicales siempre han contado con una gran experiencia en la organización y recuperación de información asociada a su contenido. Sin embargo, el aumento del ancho de banda y el incremento de los volúmenes de información del contenido de música, ha cambiado por completo su modo de difusión, resultando que en la *web* exista un sin número de archivos musicales ubicados de manera desordenada. Donde la creación de las llamadas "Bibliotecas Digitales de Música" (BDM) logra una mayor organización de los archivos musicales a partir del rastreo e indexación (Galdón, 2016).

Sistemas como *Spotify y MusicAll* son algunas de las BDM que destacan en la recopilación de archivos musicales, muchas funcionan con una sola plataforma tecnológica en la que resaltan *Android* e *iPhone*, con un determinado público para el consumo de sus servicios. Según *GooglePlay*, el número de sistemas de reproducción y descarga de música en la *web*, crece cada año en más de un 50 % y la mayoría de estos son utilizados desde telefonía móvil; datos del sitio de análisis estadísticos *We Are Social 2018* **(Ver Anexo 2)** afirman que el 51 % de la población mundial utilizan móviles y el 48 % son usuarios de *Internet* (Kujawski, 2018). Sin embargo, el índice de desigualdad social en el

[1] **http://www.mikekujawski.ca/wp-content/uploads/2017/02/We-Are-Social-Digital-Yearkbook-2017.pdf**

planeta prolifera significativamente, donde no existe homogeneidad en el desarrollo de las TIC entre países. Muchos de los sistemas para su funcionamiento establecen diversos requisitos a cumplir como: la mayoría obligan que los usuarios utilicen sus servicios de manera *online* con suscripciones pagadas; es importante señalar que varios de estos sistemas no permiten el acceso desde Cuba a algunos servicios, debido a restricciones impuestas por el bloqueo norteamericano: *Spotify*, permite la búsqueda de archivos musicales, pero la reproducción no la realiza con la calidad necesaria y no permite la descarga del archivo.

Cuba, se encuentra en pleno avance de difusión de su repertorio musical en la *web*. Casas discográficas como Egrem, Bis Music, y Sello Colibrí en el 2017 aumentaron el índice de distribución de música a través de algunas tiendas virtuales como deCuba.com[2]. No obstante, en el Foro Internacional de Música en su Segunda Edición, se valoró la necesidad de que el país amplíe los sistemas de búsquedas de música en la *web* cubana, convocando a empresas y proyectos nacionales para el desarrollo de varios sistemas que permitan la difusión de archivos musicales bajo el dominio .cu (Agencia Informativa Latinoamericana Prensa Latina, 2017).

La nación cubana, está llevando a cabo un profundo proceso de informatización de la sociedad, que persigue como objetivo, elevar el desarrollo económico, tecnológico y social de la misma. Según los Lineamientos de la política económica y social del Partido y la Revolución se define:

"…que el sistema económico que prevalecerá, continuará basándose en la propiedad socialista de todo el pueblo sobre los medios fundamentales de producción. Donde plantea que los proyectos y empresas deben contribuir a elevar la eficiencia, deben ser sustentables y permitir desarrollar la economía del país" (PCC, 2011).

Enmarcado en este proceso, en el año 2015 el Centro de Ideoinformática (CIDI), de la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI) comenzó a implementar la plataforma de Contenido Unificado para la Búsqueda Avanzada (c.u.b.a.). Esta nueva plataforma tiene como propósito fundamental proveer a la *web* nacional de una herramienta principal para la búsqueda y análisis de contenidos en páginas cubanas mediante una interfaz amigable e intuitiva. A partir de ese mismo año, se registraron más 6 423 subdominios bajo el Dominio.cu (CUBANIC, 2015), por lo que la utilización de esta plataforma resultaría de gran importancia para Cuba; representando un paso más hacia la soberanía tecnológica.

La plataforma c.u.b.a. tiene como principio fundamental brindar gran parte de sus servicios de manera

[2] **www.decuba.com.**

gratuita a todos los usuarios que lo utilicen, con el fin de cumplir con las estrategias económicas del país. Es una herramienta que cuenta con tres mecanismos principales: una interfaz *web* de consulta para realizar las búsquedas, *Solr* como mecanismo de indexación y *Nutch* como servidor de rastreo. Para un mejor funcionamiento de la plataforma, se tiene como objetivo desarrollar varios subsistemas, con el propósito de realizar la búsqueda de todo tipo de contenidos y así mejorar el rendimiento y servicios que pueda brindar en la *web* cubana. Está enfocada en la actualidad en la búsqueda simple y avanzada de imágenes y documentos, así como la búsqueda de contenidos generales en la *web* **(Ver Anexo 3)**. Si el usuario realiza una búsqueda de un archivo musical, la plataforma lista un número de páginas que tengan alguna información similar a la consulta dada o alguna información referente a dicha búsqueda **(Ver Anexo 4)**. Donde el usuario tiene que buscar entre las páginas hasta que encuentre el archivo deseado. Siendo de esta forma ineficiente y en la mayoría de los casos nula la búsqueda y recuperación de archivos musicales. Además, la plataforma no está actualmente preparada para realizar búsquedas específicas a un determinado archivo musical que se le incluya la extensión, por ejemplo: *canción.*mp3. Por lo que resulta una desventaja para la plataforma c.u.b.a. que los contenidos publicados en la *web* nacional en formato de audio permanezcan prácticamente invisibles; viéndose limitada la capacidad de divulgación de los archivos musicales y que el usuario no pueda encontrar este contenido en la *web* cubana.

Por todo lo antes expuesto se plantea como **problema de investigación:** ¿Cómo mejorar la precisión y exhaustividad de la recuperación de archivos musicales en la plataforma c.u.b.a.?

El **objeto de estudio** comprende: Proceso de recuperación de información en la *web* nacional.

Se define como **campo de acción:** el proceso de recuperación de información de archivos musicales.

Para darle solución al problema descrito, se define el siguiente **objetivo general**: Desarrollar un subsistema de búsqueda de archivos musicales publicados en la *web* cubana, utilizando técnicas de recuperación de información para mejorar la precisión y exhaustividad del proceso de recuperación de archivos musicales en la plataforma c.u.b.a.

Para cumplir la meta propuesta se han trazado los siguientes **objetivos específicos:**

- Sistematizar los referentes teóricos en cuanto a la forma de recuperar información de los sistemas de búsqueda de archivos musicales.
- Diseñar el subsistema de archivos musicales en la plataforma c.u.b.a. para realizar la búsqueda simple y avanzada de archivos musicales.

- Implementar un subsistema de archivos musicales en la plataforma c.u.b.a. que permita mejorar la búsqueda y recuperación de archivos musicales.
- Validar el correcto funcionamiento del subsistema de archivos musicales en la plataforma c.u.b.a.

Luego de haber realizado una revisión bibliográfica y desarrollado el marco teórico, se formula la siguiente **hipótesis de investigación**: El desarrollo del subsistema de búsqueda de archivos musicales publicados en la *web* cubana, utilizando técnicas de recuperación de información, mejora la **precisión** y **exhaustividad** del proceso de recuperación de archivos musicales en la plataforma c.u.b.a.

Se define como **variable independiente**: el subsistema de búsqueda de archivos musicales de la plataforma c.u.b.a.

Se define como **variable dependiente**: la recuperación de archivos musicales en la *web* cubana realizada con la plataforma c.u.b.a.

Para dar cumplimiento a los objetivos trazados, se hace necesario desarrollar las siguientes **tareas de investigación:**

1. Estudio de los conceptos asociados a los referentes teóricos de los sistemas de búsqueda de archivos musicales, así como caracterización de las tecnologías y herramientas a utilizar en el desarrollo del subsistema de búsqueda de archivos musicales.
2. Análisis y diseño la propuesta de solución en correspondencia a la arquitectura y patrones utilizados.
3. Desarrollo de las funcionalidades para la búsqueda simple y búsqueda avanzada de archivos musicales.
4. Validación y documentación del subsistema teniendo en cuenta su funcionamiento, seguridad, rendimiento e integración con la plataforma c.u.b.a.
5. Validación la hipótesis de investigación.

Entre los **métodos** de trabajo científico empleados en esta investigación se encuentran:

- **Métodos Teóricos**

Histórico - Lógico: Para el presente trabajo se realizó un análisis histórico del desarrollo del problema objeto de estudio, siguiendo una valoración lógica sobre los principales postulados del tema expresados por diferentes autores en distintos años, partiendo desde el surgimiento, el análisis de los

antecedentes hasta las características principales, los cuales son tratados en la introducción de la investigación. Además, este método se utilizó para estudiar y determinar la evolución, comportamiento y tendencias actuales de las tecnologías y herramientas utilizadas en el desarrollo del subsistema de búsqueda de archivos musicales para la plataforma c.u.b.a.

Analítico - Sintético: Para el presente trabajo se realizó el análisis de los conceptos relacionados, lenguajes, tecnologías y herramientas utilizadas en el desarrollo del subsistema de búsqueda de música. Además, se empleó para examinar los documentos consultados durante el desarrollo de la investigación.

Inductivo-Deductivo: Para el presente trabajo se utilizó el método para desarrollar razonamientos lógicos que permitió arribar a conclusiones generales a partir de premisas particulares vinculadas con el proceso de recuperación de archivos musicales.

Sistémico-Estructural: Para el presente trabajo se utilizó este método para abordar sistemáticamente todos los procesos involucrados en las temáticas estudiadas, proporcionando una visión general integral y sistémica del fenómeno objeto de estudio, sus componentes, estructura y relaciones fundamentales que servirán de base para el desarrollo del subsistema.

- **Métodos Empíricos**

Modelación: Para el presente trabajo se utilizó para lograr una mejor comprensión de los procesos asociados a la recuperación de archivos musicales, a partir de la representación en el modelo de dominio de entidades y actores.

Entrevista: Para el presente trabajo se realizó entrevistas con la intención de obtener información referente a los procesos de búsqueda.

Para mostrar el desarrollo de la investigación y los resultados, el trabajo se ha estructurado en tres capítulos, además de las Conclusiones, Recomendaciones, Bibliografía y Anexos.

Capítulo 1. Fundamentación teórica del Subsistema de búsqueda de música para la plataforma c.u.b.a.: En este capítulo se exponen los conceptos asociados al dominio de la investigación y se realiza un estudio de soluciones existentes a nivel internacional. Se estudiaron las herramientas, metodologías, lenguajes de programación y tecnologías utilizadas para dar solución al problema planteado, así como las librerías que permiten el procesamiento de los archivos musicales.

Capítulo 2. Características del Subsistema para la búsqueda de música para la plataforma c.u.b.a.: En este capítulo se exponen las características del subsistema realizándose un análisis y

diseño del subsistema de búsqueda de archivos musicales para la plataforma c.u.b.a. Se define los requisitos funcionales y no funcionales, patrones de diseño y arquitectura utilizados; también se definen los correspondientes artefactos que requiere la metodología de desarrollo utilizada.

Capítulo 3. Implementación y prueba del Subsistema de búsqueda de música para la plataforma c.u.b.a.: En este capítulo se evidencian las actividades que se realizan durante las fases de implementación y pruebas para el subsistema de búsqueda de archivos musicales para la plataforma c.u.b.a. En estas etapas, se generan los artefactos pertenecientes a las mismas, como es el caso del diagrama de componentes, los estándares de codificación y los casos de prueba. Se incluyen en el capítulo una explicación de los *plugin* que se van a implementar en el subsistema.

Se espera como posible resultado un subsistema que permita la búsqueda simple y avanzada de archivos musicales publicada en la *web* cubana. Se obtendrá el documento de la investigación que podrá ser consultado por las personas que lo deseen, teniendo en cuenta sus intereses.

CAPÍTULO 1. Fundamentación teórica del subsistema de búsqueda de música para la plataforma c.u.b.a.

En el presente capítulo se exponen los conceptos asociados al dominio de la investigación y se realiza un análisis de los referentes teóricos de los sistemas de búsqueda de archivos musicales. Se incluye un estudio de la metodología, las herramientas, lenguajes y tecnologías escogidas para el desarrollo del subsistema de búsqueda de archivos musicales para la plataforma c.u.b.a.

1.1. Información

La información es un conjunto organizado de datos procesados con una determinada importancia y significado. Permite un nuevo o mayor conocimiento al individuo sobre un tema específico con la finalidad de tomar decisiones frente a una determinada cuestión (González, 2018). Varios investigadores han planteado que existe un fenómeno denominado "sobrecarga de información" representada en diferentes formatos y con diversos propósitos para el usuario (Guías Prácticas., 2014). En la actualidad, la información puede representarse digitalmente, almacenarse y distribuirse masivamente en forma simple y rápida a través de redes de computadoras. De ahí se define que un archivo musical constituye un tipo de información, de igual manera, el volumen de información de música crece permanentemente, desde estar almacenada en una computadora personal, hasta sitios digitales y espacios mucho más grandes y complejos en la *web*. Por tanto, la necesidad de la utilización de herramientas para mostrar y organizar esta información se hace esencial. De esta manera es necesario la RI para facilitar a los usuarios su búsqueda y utilización.

1.2. Recuperación de Información

La RI se define como el problema de la selección de documentos en respuesta a consultas o demandas de información por parte del usuario. Según Baeza-Yates y Ribeiro-Neto *"la recuperación de la información tiene que ver con la representación, almacenamiento, organización, acceso y procesamiento al ítem de información"* (Baeza Yates & Ribeiro Nieto, 2005). Con el desarrollo de los sistemas digitales de procesamiento de datos y tratamiento de la información, las técnicas de RI han desarrollado un conjunto de teorías, modelos y aplicaciones prácticas que subyacen en la actualidad a cualquier búsqueda que tiene lugar en la *web* (Osorio, 2013)

1.2.1. Modelos de recuperación de información

Para facilitar el proceso de RI, existen diferentes herramientas como los SRI basados en varios modelos de RI. En la presente investigación se analiza las características del modelo híbrido entre el modelo Booleano y el modelo Vectorial (Modelos, 2014):

- Debido a las particularidades de los modelos que lo originan constituye un modelo teórico sólido y su estrategia de recuperación está basada en predecir si un documento es relevante o no.
- Ordena todos los documentos de la colección en orden decreciente de su grado de similitud con la consulta, incorporando a los resultados aquellos documentos que satisfacen solo parcialmente los términos de la consulta. Obteniendo la equiparación exacta.

En general la RI representa un conjunto de actividades orientadas a facilitar la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros. Está vinculada directamente con toda la información disponible en la *web*, así como la necesidad del uso de materiales o equipos que permitan su gestión, recuperación y filtrado.

1.3. Sistema de Recuperación de Información

Los SRI se definen como: *"sistemas que emplean procesos donde se accede a una información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsquedas específicas. Dicha información ha debido ser estructurada previamente a su almacenamiento"* (Baeza, 1999).

1.3.1. Arquitectura de los sistemas de recuperación de información

Ricardo A. Baeza Yates, como uno de los autores más destacados dentro de los estudios de los SRI aborda que estos sistemas comparten una misma arquitectura, se detalla a continuación (Baeza Yates, y otros, 2005)

Interfaz: el usuario con necesidades de información bien definidas, interactúa con la interfaz del sistema, mediante la cual introduce las consultas al mismo. La interfaz puede estar basada en una página *web*, una interfaz de escritorio o ambas.

Sistema de Formulación de Consultas: realiza un preprocesamiento de las consultas transformando las consultas realizadas en el lenguaje natural a consultas entendibles por los sistemas de información.

Mecanismo de evaluación de consultas: compara los documentos representados en el sistema de información con la consulta previamente procesada, para obtener un subconjunto de documentos ordenados de acuerdo a un criterio de relevancia, y que satisfagan la consulta realizada por el usuario.

Mecanismo de rastreo: componente que se encarga de rastrear la *web* siguiendo la estructura hipertextual de la misma para almacenarlos en un lugar para su posterior análisis, en muchas ocasiones es llamado también araña o araña *web.*

A continuación, se muestra la arquitectura básica de los SRI (Ver figura 1.) (Baeza Yates, y otros, 2005).

Figura 1. Arquitectura de un SRI (Baeza Yates y otros, 2005).

1.3.2. Clasificación de los sistemas de recuperación de información

Existen diferentes SRI: los sistemas basados en texto y los sistemas basados en contenidos. Los sistemas de recuperación de información textual se han desarrollado con gran éxito durante décadas, mientras que los basados en contenidos, ejemplo los de música tienen una mayor complejidad, especialmente cuando tratan con documentos de gran tamaño o con grandes cantidades de datos (Olvera, 2016). También los SRI pueden clasificarse atendiendo a disímiles criterios de acuerdo a la función que realizan o al alcance que poseen. Entre las clasificaciones que más destacan están: los motores de búsquedas, los metabuscadores y los directorios. La presente investigación se centra en los Motores de Búsquedas (MB) como SRI basado en contenido. Para una mejor compresión ver tabla comparativa de los diferentes SRI **(Ver Anexo 5.)** (Parrilla, 2011).

1.4. Motores de búsquedas

Los MB o buscador según Baeza Yates y Ribeiro-Neto son *"Sistemas de Recuperación de Información que permiten obtener aquellos documentos de mayor relevancia, a partir de un criterio de búsqueda introducido por el usuario. Desde la perspectiva del usuario, los MB deben cumplir dos requisitos fundamentales: "un tiempo corto de respuesta y una gran colección de documentos web disponibles en su índice"* (Baeza Yates & Ribeiro Nieto, 2005). A partir de los conceptos investigados sobre los MB se expresa que son sistemas distribuidos que cuentan con un programa informático denominado *spider* que recorre la *Web* de forma automática y almacena la información indexándola en una base de datos.

1.4.1. Diseño de un motor de búsqueda

Según varios autores el diseño típico de un buscador *web* es en cascada **(Ver Anexo 6.)** Donde las operaciones son ejecutadas en el siguiente orden (Franco, 2014):

1. Recolección de contenido *web.*
2. Indexación del contenido recolectado.
3. Búsqueda de información por parte de los usuarios.

A partir del estudio de estas tres etapas se considera que el proceso de recolección de contenido *web* es la etapa más importante dentro de la búsqueda porque se extraen de los documentos las palabras o términos más importante de la consulta **(Ver Anexo 7.)** (Franco, 2014).

1.4.2. Ventajas y desventajas de los motores de búsquedas

Los MB presentan las siguientes ventajas (Gutiérrez, 2013):

- Realizan el descubrimiento del contenido de la *web* mediante un proceso automático, lo que permite analizar una mayor cantidad de información.
- Mantienen las bases de datos más actualizadas que los directorios.
- Asignan a la información recolectada un orden de acuerdo con la calidad de su contenido.
- Permiten la realización de búsquedas avanzadas.

Sin embargo, cuando los usuarios efectúan una consulta, el motor busca la información y devuelve como respuesta, una lista con las *URLs*[3] de los documentos que se ajustan a los criterios establecidos en dicha consulta. Esta búsqueda no se efectúa directamente sobre la *web*, debido a su inmenso volumen y al tiempo requerido para dar respuesta a los usuarios. Estos sistemas consideran la *web* como una extensa base de datos lo que provoca desventajas en su funcionamiento, como que la información sea redundante o que se encuentre desordenada (Ibarra, 2014).

1.5. Rastreador

El rastreador (en inglés, *Crawlers*) se encarga de realizar peticiones a servidores distantes en busca de la información contenida en los mismo a través de distintos protocolos como los de la familia TCP/IP (González, 2018). A partir de estudios relacionados con el rastreador el autor lo define como un programa que cruza la *www* moviéndose de un documento a otro, decidiendo progresivamente que rastrear, con el fin de obtener resultados satisfactorios para el usuario a la hora de introducir una

[3]**Localizador de recursos uniforme. Es una cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet.**

consulta en los MB. Los procesos de un rastreador se pueden clasificar generalmente en 5 tipos. A continuación, se resume cómo funcionan **(Ver Anexo 8)**:

1. El rastreador visita la página *web*, entrando por el *root,* lee todo el contenido y crea una lista de lo que ha encontrado.

2. La información es indexada según los algoritmos internos usados por el buscador.

3. Esta información es llevada a una central donde se almacena.

4. Cuando el usuario realiza una búsqueda, el sistema muestra todas las *webs* que contienen la palabra o frase buscada.

5. El orden en que muestra los resultados depende de los algoritmos internos en los que se tiene en cuenta la importancia de una página *web*.

Los recorridos del rastreador en la *web* pueden ser realizados en profundidad o a lo ancho. Donde cada uno de los documentos encontrados son analizados y almacenados por el indexador (Rodríguez, 2012).

1.6. Indexación

La indexación se expresa como el proceso de almacenar la información ordenadamente para acelerar su búsqueda y seleccionarla con mayor exhaustividad y precisión, independientemente de que se solicite o no. Varios autores describen al proceso como una operación destinada a representar los resultados del análisis del contenido de documentos, de una parte del documento o de ficheros, mediante elementos (términos de indexación) de un lenguaje documental o natural, orientados a facilitar la posterior recuperación de los documentos indexados y acceso al contenido de los materiales (Rodríguez, 2012). De acuerdo con lo planteado, se precisa que la indexación como la creación de índices para posteriormente acceder a los archivos, mediante elementos descriptivos encontrados.

1.6.1. Indexación para audios

Durante años se han desarrollado un conjunto de técnicas que permiten la indexación de audios. Debido al incremento de los volúmenes de información de audio, se ha tenido que mejorar las técnicas mediante el uso de metadatos con el objetivo de hacer más efectiva la recuperación. Los metadatos son los que contienen la información concisa del objeto al que representan. Tienen un tamaño mucho menor que el objeto en sí, facilita a una computadora su localización, identificación y descripción y hace posible la búsqueda de información en múltiples lugares a la vez. Lo que permite establecer restricciones de uso, condiciones de licenciamiento e información sobre los derechos de

autor. Estos pueden ser el título del audio, nombre del autor, el tipo de formato, la duración, etc. De forma general las técnicas de indexación se pueden representar de diversas maneras, las más típicas son: secuencial, multinivel y árbol, en esta última se basan casi todas las técnicas de indexación actuales porque viene a mejorar el problema del crecimiento de entradas en un nivel (Caldera, 2016). Los metadatos descriptores[4], es el tipo de metadato más utilizado para la recuperación de archivos de audio, que permite la búsqueda avanzada de los materiales (Caldera, 2016).

1.7. Análisis de otras soluciones existentes

En la actualidad, existen varias compañías que se han dedicado al desarrollo de sistemas para la recuperación de materiales audiovisuales, contado con un subsistema que le permite la búsqueda de archivos musicales. En este epígrafe se analizan algunas de las soluciones existentes con gran auge en el mercado internacional, con el propósito de identificar sus ventajas y desventajas.

1.7.1. Buscadores existentes internacionales

Refiriéndose a los avances en el desarrollo de la informática a nivel internacional, según investigaciones realizadas se encontraron varios sistemas de recuperación y reproducción de archivos musicales:

Yahoo! Music[5]: Es un sistema de recuperación de archivos musicales, bajo la plataforma *Android* y sistema operativo *Windows* que permite buscar, comprar y reproducir el contenido de la música brindándole a los usuarios la *URL* dónde encontrar los archivos. Se alimenta del *robot* rastreador de *Yahoo! Search* que permite recolectar de manera ilimitada los enlaces donde se ubican los archivos musicales, teniendo en cuenta diferentes metadatos como: artista, álbum, género o etiquetas de grabación. (Yahoo! Inc. The History of Yahoo! - How It All Started, 2017).

Spotify[6]: Es un sistema que entre sus servicios ofrece la recuperación de archivos musicales. Actualmente es el más popular en la reproducción y descarga de música. Cuenta con más 140 millones usuarios en todo el mundo y ofrece servicios gratuitos como: la búsqueda y reproducción de archivos musicales de diferentes géneros. Está disponible para la mayoría de los servicios modernos, incluidos los equipos como *Windows, MacOs*, y *Linux*, así como los teléfonos inteligentes y tabletas. Registra un catálogo de más de 30 millones de canciones donde la música puede buscarse por su nombre o mediante el uso de metadatos como: artista, álbum, género, lista de reproducción, o etiquetas de grabación (Spotify, 2017).

[4] **Detallan las principales características de un archivo dígase en este caso título, autor, formato, álbum, duración, etc.**
[5] **www.yahoo.com.**
[6] **www.spotify.com**

Apple Music[7]: Es un sistema de recuperación y reproducción de música que tiene la ventaja de funcionar en modo *offline* con un robot rastreador que indexa música a su base de datos diariamente, por lo que constituye la aplicación con más cúmulo de música en internet, permitiendo al usuario crear diferentes bibliotecas en el equipo, con el fin de organizar los archivos musicales según los metadatos asociado a su contenido. Está disponible para sistemas operativos *MacOs* y *Windows,* así como para dispositivos de *Apple*. Tiene una base de datos de más 37 millones de canciones donde la música puede buscarse por su nombre o mediante el uso de metadatos como: artista, álbum o género (Apple Music, 2017).

MusicAll[8]: Es un sistema *online* que permite la recuperación, reproducción y descarga de archivos musicales desde un dispositivo portátil o cualquier equipo de manera gratuita, revolucionando los métodos y vías de obtención de ganancia de los demás sitios. Está disponible para *iOS* y *Android*, así como para cualquier equipo con sistema operativo *Windows*. Comprende en su base de datos más de 25 millones de archivos musicales tomadas de la biblioteca de *YouTube* donde se puede establecer la búsqueda de música por el nombre o por varias etiquetas definidas como: artista, álbum, género, lista de reproducción, o etiquetas de grabación (MusicAll, 2016).

1.7.2. Contenido Unificado para la Búsqueda Avanzada (c.u.b.a.)

La plataforma c.u.b.a. creada a partir del año 2015 como continuación del motor de búsqueda Orión, es un sistema cuyo propósito fundamental es proveer a la red nacional de una herramienta principal para la búsqueda y análisis de contenidos cubanos. Implementada con varias herramientas libres como: *Nutch* en su versión 1.9 para el proceso de rastreo y almacenamiento de la información, *Solr* en su versión 6.3 para el componente de indexación, *Symfony* en su versión 2.8 para la interfaz *web* y la interacción con el componente de indexación, MongoDB como gestor de base de datos no relacional para almacenar estadísticas, Java versión 8.0 para el manejo de *plugins* en *Nutch* y *Solr*, HTML 5 y CSS 3 para la interfaz *web*, PHP en su versión 8.0 y Bootstrap en su versión 4.0. Está enfocada en la actualidad en la búsqueda simple y avanzada de imágenes, así como la búsqueda de contenidos generales en la *web*. Enlaza a servicios como: la enciclopedia EcuRed, el sitio Cubadebate, la cartelera cultural La Papeleta, la plataforma de blogs Reflejos, entre otros.

1.7.3. Comparación y Resultados de soluciones existentes

Luego del estudio realizado en el epígrafe sobre las soluciones existentes a nivel internacional y de la plataforma c.u.b.a se establece la siguiente comparación:

[7] **www.applemusic.com**
[8] **www.musicall.com**

Tabla 1. Comparación de los Sistemas en cuanto a la búsqueda de música (Elaboración propia).

Funcionalidades	Sistemas				
	Yahoo! Music	Spotify	Apple Music	MusicAll	c.u.b.a
Búsqueda simple de archivos musicales	Sí	Sí	Sí	Sí	No
Búsqueda avanzada de archivos musicales	Sí	Sí	Sí	Sí	No
Mostrar la URL donde se encuentra del archivo musical	Sí	No	No	No	No
Mostrar información asociada al archivo musical	Sí	Sí	Sí	Sí	Sí
Recuperación del archivo musical	Sí	Sí	Sí	Sí	No

En la tabla anterior (Ver tabla 1.) se aborda que todos los sistemas internacionales estudiados, realizan las búsquedas simple y avanzada de archivos musicales a diferencia de la plataforma c.u.b.a. que actualmente prescinde de esta función. Otra de las funciones que se observa en la tabla, es que los sistemas internacionales muestran alguna información referente al contenido del archivo buscado al igual que la plataforma c.u.b.a.

Tabla 2. Comparación de los Sistemas en cuanto sus características (Elaboración propia).

Características	Sistemas				
	Yahoo! Music	Spotify	Apple Music	MusicAll	c.u.b.a.
Sistemas Operativos	Windows	Windows, MacOs, Linux.	MacOS, Windows.	Windows MacOs.	Linux y Windows.
Funcionamiento	Online.	Online y Offline.	Online y Offline.	Online y Offline.	Online.
Tipos de servicios	Suscripción paga.	Gratis y Suscripción paga.	Suscripción paga.	Gratis.	Gratis
Código Fuente	Privado	Privado	Privado	Privado	Libre

En la tabla anterior (Ver tabla 2.) se aborda que no todos los sistemas internacionales trabajan con *Linux* y las políticas por las cuales se rigen son privativa, por lo que no se tiene acceso al código

fuente. A diferencia de la plataforma c.u.b.a. que no se rige por políticas extranjeras, y su código fuente es libre.

Análisis de los Resultados:

En general los sistemas internacionales no pueden dar solución a la problemática de la presente investigación debido:

1. Las políticas internas a las cuales se rigen son totalmente diferentes a las políticas internas definidas en nuestro país.
2. El código fuente que utilizan los sistemas internacionales son privativos.
3. Se requiere de una alta infraestructura tecnológica para su utilización.
4. La mayoría de estos sistemas no están disponibles para *Software* Libre.

Por lo que se decide desarrollar un subsistema de búsqueda de música para la plataforma c.u.b.a., como propuesta de solución al problema planteado. Para el desarrollo del subsistema, se tomará en cuenta algunas ventajas que ofrecen los sistemas internacionales analizados anteriormente:

1. Procesan la música mediante el uso de metadatos como artista, género, tamaño, etc.
2. Los diseños de la interfaz *web* son bastante intuitivos para el usuario.
3. Realizan la búsqueda avanzada de música filtrando el contenido de los metadatos del archivo.
4. Realizan el proceso de rastreo del contenido de música periódicamente.

1.8. Entorno de desarrollo

Para el desarrollo de un *software* se deben tener en cuenta ciertos elementos, como son la metodología de desarrollo, los lenguajes de programación y las tecnologías y herramientas básicas para su construcción. A continuación, se describe la estructura del entorno de desarrollo de la propuesta de solución:

1.8.1. Metodología de desarrollo de software

AUP-UCI

La metodología Proceso Unificado Ágil (por sus siglas en inglés AUP) representa una versión simplificada de la metodología ágil RUP[9], donde se describe de una manera simple y fácil la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se

[9]**Proceso Unificado de Rational (en inglés Rational Unified Process) es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos**.

mantienen válidos en RUP. AUP aplica técnicas ágiles incluyendo el modelado ágil, el desarrollo dirigido por pruebas y la gestión de cambios ágil. La metodología establece cuatro fases que transcurren de manera consecutiva:

1. **Inicio**: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. **Elaboración**: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. **Construcción**: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. **Transición**: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega.

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos): modelo, implementación, prueba, despliegue, gestión de proyectos y entornos. Cuenta con 9 roles: administrador de proyecto, ingeniero de procesos, desarrollador, administrador, modelador ágil, administrador de configuración, administrador de pruebas y probador. Además, se apoya en el Modelo CMMI-DEV v1.3., el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Es importante resaltar que AUP es la metodología que se utiliza en los proyectos productivos de la UCI debido a las ventajas que ofrece. Funciona para equipos de desarrollo pequeño y está más orientada a la generación de código con ciclos muy cortos de desarrollo, haciendo especial hincapié en aspectos humanos asociados al trabajo en equipo e involucrando activamente al cliente (Sánchez, 2015).

1.8.2. Apache Nutch

Nutch es un rastreador *web* de código abierto, desarrollado totalmente en Java por *The Apache Software Foundation*. Ha sido implementado sobre la base de *Apache Lucene*, una librería de alto rendimiento para la búsqueda basada en texto y que utiliza una modificación del algoritmo *"Vector Space Model"* (Modelo de Espacio Vectorial en español), con un enfoque booleano que restringe las estimaciones de los resultados obtenidos (ApacheNutch, 2018). La arquitectura de *Nutch* es muy flexible, permitiendo realizarle mejoras por parte de los usuarios a través de *plugins*. Es independiente del servidor de indexación permitiendo la integración con *Solr*, encargándose este último de las tareas de realización del índice incremental, o sea se actualiza en cada rastreo con los nuevos documentos encontrados. *Nutch* permite la recuperación de muchos tipos de documentos utilizando componentes implementados por separado. Otras de sus características es que es multiplataforma, donde recolecta

la información en formato PPT, DOC, PDF, HTML, XML, TXT, JPG, MP3, MP4, entre otros. Actualmente es el rastreador que utiliza la plataforma c.u.b.a. por las ventajas que ofrece. A continuación, se describe las fases del rastreo de *Nutch*:

Fases del Rastreo de *Nutch* (Ver Anexo 9.) (Nioche, 2013):

1. Inyectar *URLs (injection)*.
2. Generar *URLs* a visitar *(generating).*
3. Visitar *URLs* (*fetching*).
4. Analizar Documentos *(parsing).*
5. Actualizar BDs *(CrawlDB, linkDB) (merging DB).*
6. Inversión de enlaces *(link invertion).*
7. Eliminar *URLs* duplicadas *(deduplicate).*

1.8.3. Apache Solr

Es una popular y rápida plataforma de búsquedas de código abierto basada en *Apache Lucene*. Sus principales características incluyen búsquedas de texto completo, resaltado de resultados y manejo de documentos (como *Word* y PDF). *Solr* es escalable, realiza búsquedas distribuidas y replicación de índices, actúa como una base de datos no *SQL* manejando un gran volumen de información, y utiliza la biblioteca Java de búsqueda en su base para la indexación de texto completo. Este servidor de índice empresarial tiene como REST HTTP / XML y JSON APIs que hacen que sea fácil de configurar y de utilizar desde prácticamente cualquier lenguaje de programación. La potente configuración externa de *Solr* permite que sea adaptado a casi cualquier tipo de aplicación Java sin codificación, simplemente hay que utilizarlo con peticiones *GET* para realizar las búsquedas en el índice, y *POST* para agregar documentos **(Ver Anexo 10.)** En la actualidad es bastante usado por importantes sitios en Internet como: el sitio *web* de la Casa Blanca, *Instagram*, *Jobreez* y otros, gracias a las ventajas que ofrece (The Apache Software Foundation-Solr, 2014):

- **Búsqueda distribuida**: La búsqueda puede realizarse en varios fragmentos/índices y al concluir la misma los resultados serán agregados.
- **Búsquedas Facetadas**: Cuando se realiza la búsqueda se muestra un contador para cada categoría en los resultados de búsqueda.
- **Búsqueda Geo-espacial**: La búsqueda puede realizarse por localización y por la distancia. (Ejemplo: Buscar dentro de 5 km de la posición actual).

Su arquitectura se divide en dos partes:

1. **Índice:** Sistema de ficheros que almacenan la información. Contiene la configuración de *Solr* y la definición de la estructura de datos.
2. **Servidor**: Proporciona el acceso a los índices y las características adicionales. Admite *plugins* para añadir funcionalidades.

Este mecanismo de indexación utiliza un modelo de recuperación de información híbrido basándose al igual que *Nutch* en el modelo booleano y en el modelo de espacio vectorial para establecer el subconjunto de documentos relevantes, siendo el servidor de índice utilizado por *Nutch*, por lo que la utilización de *Solr* conjunto a *Nutch* ayuda en el desarrollo del proyecto. Además, que *Solr* es el mecanismo de indexación que utiliza c.u.b.a. por lo que facilita la integración con la plataforma.

1.8.4. Lenguajes de programación

Java

Es un lenguaje de programación libre, que recoge los elementos de programación que típicamente se encuentran en todos los lenguajes de programación. Creado por *Sun Microsystems* en 1995, con la ventaja de poder ser utilizado en cualquier plataforma que cuente con la máquina virtual de Java instalada. Cuenta con una amplia documentación disponible para la comunidad de manera gratuita y una gran variedad de API que facilitan el trabajo de los desarrolladores. Con Java se pueden publicar servicios *web* sin necesidad de utilizar servidores externos. Brinda una API completa que permite crear aplicaciones basadas en *plugins* con gran facilidad. Es uno de los lenguajes de programación más empleados en el mundo por los desarrolladores de *software*, lo que da una prueba fiable de la popularidad y utilidad de este lenguaje para el desarrollo de aplicaciones informáticas que abarcan un número significativo de escenarios El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (Oracle, 2014).

Entre sus principales ventajas se encuentra:

1. Es independiente de la plataforma de desarrollo.
2. Existen dentro de su librería, clases gráficas que permiten crear objetos visuales comunes altamente configurables y con una arquitectura independiente de la plataforma.
3. Permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.

Este lenguaje de programación está integrado con *Nutch* y *Solr*, además, es el lenguaje que se usa en la implementación de los *plugins* de *Nutch*.

PHP

PHP (en inglés *Hypertext Preprocessor*) es un lenguaje de programación de código abierto muy popular, especialmente utilizado para crear aplicaciones para servidores o generar contenido dinámico para sitios *web*, permitiendo la creación de aplicaciones *web* muy robustas, al posibilitar la conexión a diferentes tipos de servidores de base de datos como: *MySQL, PostgreSQL*, *Oracle*, *Adabas D, dBase*, *Empress*, *Ingress*, InterBase, FrontBase, *DB2, Informix*, *mSQL*, *ODBC*, *Oracle, Sybase* y otras. Es un lenguaje multiplataforma que puede ser incrustado en HTML, teniendo la capacidad de ser ejecutado en la mayoría de los sistemas operativos e interactuar con varios servidores *web*. Entre sus principales ventajas se encuentra (Bakken, 2013):

1. **Sintaxis cómoda**: PHP cuenta con una sintaxis similar a la de C, C++ o *Perl*. Lo más destacado ocurre a nivel semántico: el tipado es muy poco estricto. Es decir, cuando creamos una variable no tenemos que indicar de qué tipo es, pudiendo guardar en ella datos de cualquier tipo.
2. **Soporta objetos y herencia**: Tiene soporte para la programación orientada a objetos, es decir, es posible crear clases para la construcción de objetos, con sus constructores, etc. Además, soporta herencia, aunque no múltiple.
3. **Licencia de software libre**: Es un lenguaje basado en herramientas con licencia de software libre, es decir, no hay que pagar licencias, ni existen límites en su distribución y es posible ampliarlo con nuevas funcionalidades.

HTML5

HTML5 es la última evolución de la norma que define el Lenguaje de Marcas de Hipertexto (HTML por sus siglas en inglés). Se trata de una nueva versión del lenguaje, con nuevos elementos, atributos y comportamientos, y un conjunto amplio de tecnologías que permite crear sitios *web* y aplicaciones diversas y de gran alcance. El nuevo estándar de HTML permite que el formato de la *web* (formado por elementos como cabecera, pie y navegadores) se agrupe en nuevas etiquetas que representan cada una de las partes típicas de una página. Se utilizará el lenguaje HTML5 para el diseño de las interfaces del subsistema, aprovechando las características del soporte para CSS3 y el manejo mejorado de formularios en el navegado (Tortajada, 2014).

Entre las ventajas que ofrece HTML 5 se encuentran:

1. **Sin conexión y almacenamiento**: Permite a las páginas *web* almacenar datos localmente en el lado del cliente y operar sin conexión de manera más eficiente.
2. **Multimedia**: Otorga un excelente soporte para utilizar contenido multimedia como lo son audio y video nativamente.
3. **Rendimiento e Integración**: Proporciona una mayor optimización de la velocidad y un mejor uso del hardware.

Hojas de estilo de cascada (CSS3)

CSS es un lenguaje de hojas de estilos en cascada (en inglés *Cascading Style Sheets)* creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Este lenguaje ofrece una nueva gran variedad de opciones para hacer diseños más sofisticados, mejora la accesibilidad de los documentos, reduce la complejidad de su mantenimiento y permite visualizar los mismos documentos en infinidad de diferentes dispositivos. Se utiliza para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otros (Tortajada, 2014).

Bootstrap

Bootstrap es un marco de trabajo de *software* libre, intuitivo y de gran alcance para el diseño de sitios y aplicaciones *web*. En la actualidad se ha convertido en uno de los proyectos de código abierto de gran popularidad en el mundo. Desarrollado por *Twitter*, *Bootstrap* simplifica el proceso de creación de diseños *web* combinando CSS y *JavaScript*. La mayor ventaja es que se puede crear interfaces que se adapten a los distintos navegadores con la propiedad denominada diseño adaptativo. Los diseños creados con *Bootstrap* son simples, limpios e intuitivos, dándole agilidad a la hora de cargar y adaptarse a otros dispositivos como tabletas y móviles a distintas escalas y resoluciones. Integrándose perfectamente con las principales librerías *JavaScript*, como *JQuery.* Otras de las características son (Thornton, 2013):

1. Ofrece un diseño sólido usando LESS y estándares como CSS3 y HTML5.
2. Dispone de distintos diseños predefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos.
3. Todas las clases CSS están contenidos en un solo archivo y permite una mejor personalización mediante mejoras, herencias y especificidad.

1.8.5. Herramientas

Visual Paradigm

Es un tipo de herramienta *CASE* que soporta el ciclo de vida completo en el desarrollo de *software*: análisis y diseño orientado a objetos, construcción, prueba y despliegue. Permite dibujar todo tipo de diagrama de clases, generar código fuente a partir de diagramas y generar documentación. Soporta el intercambio de diagramas *UML* y modelos con otras herramientas, así como la importación y exportación a formatos XMI, XML y archivos *Excel*. Permite la captura de requisitos mediante diagramas de requisitos, modelado de caso de uso y análisis textual. Posee un entorno para la especificación de detalles de casos de uso, incluyendo la especificación del modelo general y las descripciones de los casos de uso. *Visual Paradigm* posibilita la generación de código e ingeniería inversa para diversos lenguajes, entre los que se encuentra PHP y Java por lo que constituye una excelente herramienta para el modelado de los casos de usos y diagramas de la propuesta a desarrollar (Visual Paradigm, 2017).

NetBeans

NetBeans es una herramienta modular de desarrollo para un amplio rango de tecnologías para el desarrollo de aplicaciones, donde los programadores pueden escribir, compilar, depurar y ejecutar programas. Aunque inicialmente fue ideado para Java, puede ser empleado para la codificación de aplicaciones en múltiples lenguajes de programación, incluyendo un avanzado editor para varios lenguajes, editor de perfiles, detector de errores y varias herramientas para el control de versiones y desarrollo colaborativo. Proporciona aplicaciones de muestra en forma de plantillas de proyectos para todas las tecnologías que soporta, así como una amplia variedad de *plugins* para el desarrollo de proyectos (Oracle, 2014). *NetBeans* además de ser gratuito y sin restricciones de uso, posee versiones para los distintos sistemas operativos del mercado, convirtiéndolo en una alternativa con grandes ventajas para los desarrolladores. La estructura modular de *NetBeans* le proporciona estabilidad y grandes posibilidades de ser extendido gradualmente por desarrollos comunitarios, permitiendo agregar continuamente nuevas funcionalidades. Su versatilidad lo ha convertido en el IDE por excelencia entre miles de programadores alrededor del mundo (Oracle, 2014).

Apache Jmeter

Es una herramienta de código abierto diseñada para medir el rendimiento de las aplicaciones *web* a partir de comportamientos funcionales. Originalmente fue diseñada para probar servicios *web*, pero se ha expandido a otras funciones de prueba como: análisis estáticos y dinámicos, simulación de una carga pesada en un servidor, grupo de servidores, en la red y para hacer un análisis gráfico de rendimiento. Entre sus principales ventajas se encuentran (Apache JMeter., 2015):

1. Completamente multi-hilo, permite el muestreo simultáneo de muchas funciones.

2. Portabilidad completa 100% Java.
3. Su diseño permite agilizar el funcionamiento en un tiempo preciso del servicio analizado.
4. Permite el análisis offline.

Acunetix Web Vulnerability

Es una herramienta para el análisis de la seguridad en aplicaciones *web* desarrollado con el lenguaje de programación *JavaScript*. Permite el escaneo de sitios *web* y chequea automáticamente vulnerabilidades que aparentemente no son detectadas, aportando informes de auditoría de seguridad *web*. *Acunetix* permite realizar pruebas para ataques de inyección de código, secuencia de comandos en sitios cruzados y falsificación de peticiones. Posee un componente que facilita la realización de pruebas a formularios y a áreas protegidas por contraseña. Realiza varias peticiones simultáneamente, siendo capaz de explorar cientos de páginas sin interrupciones (Acunetix Web Vulnerability, 2017).

1.8.6. Marco de trabajo

Symfony

Symfony es un marco de trabajo de PHP para el desarrollo de aplicaciones y sitios *web* rápidos y seguros. Su código, y el de todos los componentes y librerías que incluye se publican bajo licencia MIT de *software* libre. Cuenta con un sitio *online* donde se puede encontrar gran cantidad de documentación de forma gratuita. Este marco de trabajo separa la lógica de negocio, la del servidor y la presentación de la aplicación *web* utilizando el Modelo Vista Controlador como patrón de arquitectura *web*, propicia varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. *Symfony* es compatible con la mayoría de gestores de bases de datos, simbolizando uno de los mejores marcos de trabajo de PHP por su excelente rendimiento, aprovechando al límite las nuevas características de PHP. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellos componentes que no encajan en el proyecto. Para el diseño de aplicaciones *web Symfony* utiliza *Doctrine* como Mapeo Objeto-Relacional (ORM) para generar la capa de acceso a datos. Se puede ejecutar tanto en plataformas *Unix* (*Linux,* etc.) como en plataformas *Windows* (Eguiluz, 2014).

Conclusiones del capítulo

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- El estudio realizado de los conceptos asociados a los referentes teóricos de la investigación, permitió comprender como funcionan los Sistemas de Recuperación de Información.
- El estudio realizado de las soluciones existentes permitió determinar que existe un acceso limitado a los sistemas de búsqueda de archivos de musicales internacional y falta de ausencia de estos en el ámbito nacional.
- El análisis de los diferentes criterios para filtrar los resultados en los sistemas internacionales, permitió identificar diferentes filtros a incorporar en el subsistema de música como el formato, el autor y la duración de archivo musical.
- El estudio realizado de las principales herramientas, tecnologías y lenguajes de programación a utilizar, permitió realizar la selección de los más adecuados para dar solución al problema planteado.

CAPÍTULO 2: Características del subsistema de búsqueda de música para la plataforma c.u.b.a.

En el presente capítulo se abordarán aspectos fundamentales relacionados con la propuesta de solución, entre los que destacan: el diagrama del modelo del dominio, la representación de los principales procesos mediante casos de uso, diagramas de clases del diseño, de colaboración y de despliegue. Para el diseño de la aplicación se definirán los estilos y patrones de arquitectura y diseño. Se establecerán vías para definir las futuras funcionalidades de la aplicación, generando los artefactos relacionados a la especificación de los requisitos funcionales y no funcionales que deberá poseer la propuesta de solución para así satisfacer el objetivo de la presente investigación.

2.1. Modelo de dominio

El modelo de dominio es un artefacto de la disciplina de análisis, el cual constituye una representación de las clases conceptuales del mundo real. Su objetivo principal radica en comprender y describir las clases más importantes dentro del contexto del sistema (Cabrera, 2012).

2.1.1. Descripción de clases del modelo de dominio

La modelación del dominio constituye la herramienta fundamental para garantizar la comprensión y descripción de las clases o conceptos y sus relaciones más importantes dentro del contexto del problema. A continuación, se presenta la descripción de los conceptos identificados en la presente investigación.

Tabla 3. Descripción de las clases del modelo del dominio (Elaboración propia).

Conceptos	Descripción
Usuario	Persona que realiza la búsqueda en la aplicación *web*.
Administrador	Persona que configura el rastreador, el indexador y la aplicación *web* e inicia el rastreo de los archivos musicales en la *web*.
Aplicación *Web*	Componente del MB encargado de recibir las peticiones de los usuarios, consultar los datos en el indexador y mostrarle al usuario los resultados de su búsqueda.
Rastreador	Mecanismo que se encarga de acceder a la información pública en la web y procesa los archivos musicales encontrados.
Indexador	Componente del MB que almacena los resultados del rastreo e forma de índice para una mejor selección de los datos solicitados por la aplicación *web*.
Motores de Búsqueda.	Los MB son sistemas de RI compuesto por la aplicación *web*, y uno o más rastreadores e indexadores.
Documento	Tipo de contenido electrónico que puede o no contener un archivo musical.
Archivo Musical	Tipo de contenido del cual se almacenará sus datos para mostrarlos al usuario.

Diagrama de Clases del Modelo del Dominio

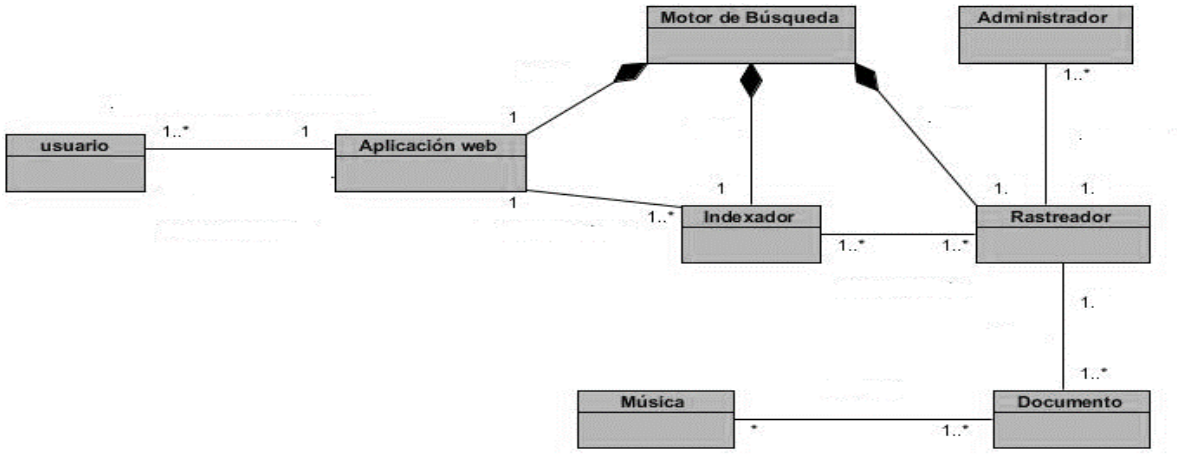

Figura 2. Diagrama de Clases del dominio.

En el modelo de dominio representado (Ver figura 2.), se muestra la relación existente entre todos los conceptos que intervienen en la investigación que se presenta. El modelo parte de las acciones realizadas por un usuario o administrador. Un usuario puede introducir uno o varios criterios de búsqueda en la aplicación web, consulta los datos en el indexador existente, atendiendo a lo que se desea buscar. El indexador le devuelve a la aplicación *web* todos los datos que coinciden con la búsqueda y esta a su vez, se los muestra al usuario. Por otra parte, un administrador inicia el rastreo de los documentos presentes en la *web* por un rastreador. De los documentos, pueden ser analizados el texto contenido en ellos u otro contenido existente. Una vez obtenida la información de los recursos rastreados, es enviada al indexador donde este almacena la información.

2.2. Descripción del subsistema para la búsqueda de música

El subsistema que se propone desarrollar (Ver Anexo 25.), pretende mejorar la calidad del proceso de recuperación de archivos musicales en la plataforma c.u.b.a. y así disminuir el tiempo de búsqueda en el que el usuario busca entre las páginas y encuentra información o la extensión de archivo musical en la *web* nacional. Estará compuesto por tres componentes: una interfaz *web* de consulta para realizar las búsquedas, *Solr* en su versión 6.3 como componente de indexación y *Nutch* en su versión 1.9 como servidor del proceso de rastreo y almacenamiento de la información. El sistema debe permitir a cualquier usuario (sin previa

autenticación) realizar búsquedas simples y avanzadas, donde se podrán filtrar los contenidos atendiendo a los siguientes criterios:

Formato: El sistema deberá permitir a los usuarios buscar archivos musicales atendiendo al tipo de formato (por ejemplo: .mp3). Para esto, el administrador podrá seleccionar los formatos que desea que el sistema rastree, indexe y muestre a los usuarios.

Título: El sistema permitirá a los usuarios buscar archivos musicales atendiendo al título del archivo musical (por ejemplo: unicornio azul).

Autor: Los usuarios podrán filtrar los resultados de sus búsquedas para obtener el archivo musical, de acuerdo al nombre del autor, (por ejemplo: Silvio Rodríguez).

2.3. Especificación de los requisitos de software

Los requisitos o requerimientos de *software* permiten describir lo que el sistema debe hacer, sus características fundamentales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo del *software*, donde expresan las necesidades objetivas o deseos que satisface un producto (Sarmiento, 2010). Luego de haber definido y modelado los conceptos asociados al dominio del problema y su relación entre cada uno de ellos, se presentan los requisitos funcionales y no funcionales de la herramienta a desarrollar.

2.3.1. Requisitos funcionales

Los requisitos funcionales (RF) o funciones del sistema constituyen una descripción de las tareas del sistema (Sarmiento, 2010). A continuación, se muestran los requisitos funcionales del subsistema de búsqueda de música publicadas en la *web* nacional.

Tabla 4. Requisitos Funcionales (Elaboración propia).

Número	Requisitos	Complejidad
RF1	Rastrear los archivos musicales publicados en la red.	**Alta**
RF2	Filtrar contenidos por el título del archivo musical.	**Alta**
RF3	Filtrar contenidos por el autor del archivo musical.	**Alta**
RF4	Filtrar contenidos por el formato del archivo musical.	**Alta**
RF5	Identificar el tamaño del archivo musical.	**Alta**
RF6	Identificar el título del archivo musical.	**Alta**
RF7	Identificar el autor del archivo musical.	**Alta**
RF8	Identificar el formato del archivo musical.	**Alta**
RF9	Identificar la URL donde se encuentra el archivo musical.	**Alta**
RF10	Realizar búsqueda simple de música.	**Alta**

RF11	Mostrar el título del archivo musical.	**Alta**
RF12	Mostrar la URL de la página donde se encuentra el archivo.	**Alta**
RF13	Mostrar el tamaño del archivo musical.	**Alta**
RF14	Mostrar el formato del archivo musical.	**Alta**

2.3.2. Requisitos no funcionales

Los requisitos no funcionales (RNF) son aquellos que definen las características y limitaciones que debe tener el sistema para alcanzar el éxito como: la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. A continuación, se muestran los requisitos no funcionales del subsistema de búsqueda de archivos musicales, agrupados fundamentalmente atendiendo a las categorías: usabilidad, software, hardware, confiabilidad, eficiencia, licencia, diseño e implementación, interfaz y seguridad (Sarmiento, 2010).

Usabilidad

RnF 1. El subsistema deberá ser utilizado por los usuarios que tengan acceso a la *web* nacional.

RnF 2. El subsistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente *web* en sentido general.

RnF 3. El subsistema debe permitir acceder a la información con una profundidad máxima de 3 *clics*

Software

RnF 4. Se requiere una distribución *GNU/Linux* como sistema operativo.

RnF 5. Se requiere la instalación del servidor *web*, de la Máquina Virtual de Java (JVM, por sus siglas en inglés) para el correcto funcionamiento del servidor de *Solr*.

RnF 6. Se requiere la instalación del servidor *web Apache* y PHP para poder visualizar la interfaz *web*.

Hardware

RnF 7. El servidor donde está desplegada la aplicación *web* deben tener como recurso mínimo de *hardware:* 4 GB de RAM.

Confiabilidad

RnF 8. El sistema no debe gestionar ni requerir información de usuarios para su uso.

Licencia

RnF 9. Se requiere el uso de herramientas y recursos de *software* libre, las cuales se podrán usar, modificar y distribuir libremente.

Eficiencia

RnF10. El sistema *web* debe poseer un tiempo de respuesta por transacción de hasta 5 segundos en dependencia de la colección documental, el número de usuarios realizando peticiones y el servidor donde se encuentre instalado**.**

RnF 11. El sistema debe permitir que los usuarios interactúen de manera concurrente.

Diseño e Implementación

RnF 12. Como lenguaje de programación para la interfaz *web* se deberá utilizar PHP en su versión 8.0 o superior.

RnF 13. Como lenguaje de programación para el *plugin* del componente de rastreo se deberá utilizar Java.

RnF 14. Para el desarrollo de la aplicación *web* se deberá utilizar *Symfony* en su versión 2.8 o superior como marco de trabajo PHP.

RnF 15. Para el desarrollo del diseño de las interfaces de la aplicación se utilizará el lenguaje HTML5.

RnF16. Para el diseño de las interfaces de la aplicación se utilizará la hoja de estilo en cascada CSS3 aprovechando las características de soporte que brinda.

RnF 17. Para el componente de rastreo se deberá utilizar la herramienta *Nutch.*

RnF 18. Para el componente de indexación se deberá utilizar la herramienta *Solr.*

RnF 19. En la elaboración del diseño de diagramas y artefactos se deberá utilizar *Visual Paradigm* para *UML* 8.0.

Interfaz

RnF 20. La interfaz gráfica de la aplicación debe de ser agradable a la vista del usuario.

RnF 21. Para la realización del diseño se requiere homogeneidad con la identidad de la plataforma c.u.b.a.

Seguridad

RnF 22. El subsistema debe estar protegido contra ataques de suplantación de peticiones en sitios cruzados (CSRF por sus siglas en inglés) e inyecciones de código.

RnF 23. El subsistema debe de tener la opción de permitir la realización de salvas periódicas de la información.

RnF 24. El administrador debe permitir que el subsistema tenga una instalación distribuida contribuyendo al balanceo de carga y la redundancia.

RnF 25. Los campos de entrada deben ser validados.

RnF 26. El administrador debe de configurar el subsistema para que por defecto tenga activada la búsqueda segura.

2.4. Arquitectura del sistema

El diseño arquitectónico representa la estructura de los datos y de los componentes del programa que se requieren para construir un sistema basado en computadora. Considera el estilo de arquitectura que adoptará el sistema, la estructura y las propiedades de los componentes que lo constituyen y las interrelaciones que ocurren entre sus componentes arquitectónicos (Pressman, 2005). Según Pressman cada estilo describe una categoría de sistemas que incluye:

- Un conjunto de componentes (como una base de datos o módulos de cómputo) que realizan una función requerida por el sistema.
- Un conjunto de conectores que permiten la "comunicación, coordinación y cooperación" entre los componentes.
- Restricciones que definen cómo se integran los componentes para formar el sistema.
- Modelos semánticos que permiten que un diseñador entienda las propiedades generales del sistema al analizar las propiedades conocidas de sus partes constituyentes (Pressman, 2005).

La propuesta de solución muestra varios estilos arquitectónicos en los tres componentes que la estructura: **Interfaz *Web***, **Indexador** y **Rastreador**. La interfaz *web* está desarrollada en *Symfony*, por lo que se basa en el patrón arquitectónico de Modelo Vista Controladores (MVC) **(Ver Anexo 11.)**, donde esa aplicación se despliega en los servidores que son los encargados de implementar la lógica del negocio y gestionar a través del protocolo HTTP las informaciones que se guardan en el Indexador. A su vez, estas informaciones son los resultados de la indexación realizada con el rastreador cuando recorre la *web.* El rastreador implementa un tipo de arquitectura basada en *plugin* **(Ver Anexo 7.)**, esta tiene la ventaja de ser flexible, permitiendo que se realicen mejoras al rastreador a través de los propios *plugins* (Aleman, y otros, 2014).

2.5. Modelo de Diseño

El modelo de diseño se encarga de describir la realización de los casos de uso del sistema, y se utiliza como medio de abstracción del modelo de implementación y el código fuente del *software*. Su objetivo fundamental es transmitir a través de la representación mediante diagramas, una comprensión en

profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones concernientes a los lenguajes de programación (Sparks, 2013).

2.5.1. Modelo de casos de uso del sistema

El modelo de casos de uso (CU) describe la funcionalidad propuesta del nuevo sistema, en el que el usuario final interactúa con el sistema en un conjunto específico de circunstancias (Sparks, 2013). A continuación, se muestra el modelo de casos de uso que describe las funcionalidades propuestas para el subsistema de búsqueda de música para la plataforma c.u.b.a.

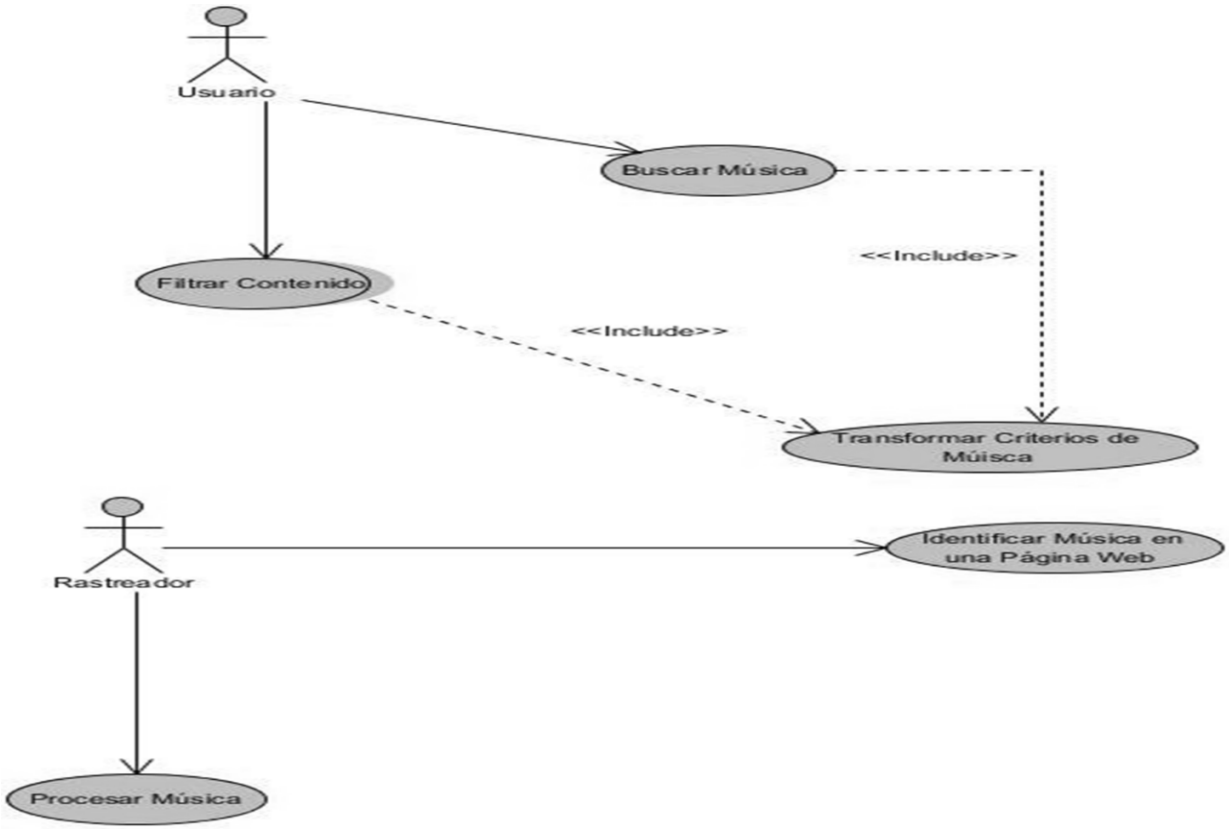

Figura 3. Modelo de caso de uso del sistema.

En la figura 3 se encuentran representados en total 5 casos de uso y los actores que interactúa con los casos. A continuación, se describe cada caso:

CU1 Buscar Música: Engloba la forma de búsqueda simple de música por parte del usuario.

CU2 Filtrar Contenido: Engloba la forma de búsqueda avanzada de música por parte del usuario, dado un criterio.

CU3 Transformar el Criterio de Búsqueda: Se refiere a la trasformación de la consulta introducida por el usuario de un lenguaje natural a un lenguaje documental o entendible para el sistema.

CU4 Identificar Música en página web: El rastreador (*Nutch*) identifica la *URL* en donde se encuentra el archivo musical (ejemplo archivo.mp3).

CU5 Procesar Música: El rastreador (*Nutch*) indexa la *URL* e información del archivo musical en el componente *Solr.*

Especificación de casos de uso.

A continuación, se realiza la descripción de los casos de uso crítico Buscar Música y Procesar Música del subsistema de búsqueda de música para la plataforma c.u.b.a., (Ver Tabla 5. y Tabla 6.) la descripción de los restantes casos se encuentran en los anexos **(Ver Anexos 12, 13 ,14)**:

Tabla 5. Descripción CU Buscar Música (Elaboración propia).

Objetivo		**Identificar la dirección donde se encuentra el archivo musical**
Actores		Administrador (inicia), Rastreador
Resumen		El sistema identifica las URLs de los archivos musicales existentes en una página web.
Complejidad		Alta
Prioridad		Alta
Precondiciones		El administrador ha configurado el sistema y ha iniciado el rastreo. El contenido de la página web ha sido obtenido y el árbol de su estructura ha sido creado.
Postcondiciones		Los archivos musicales fueron identificados.
Flujo de eventos		
Flujo básico Procesar Música		
Actor		**Sistema**
1	Ejecuta el filtro que analiza el contenido.	
2		Realiza un recorrido en profundidad X por el las páginas identificadas en el semillero hasta que alcance su límite máximo de búsqueda de música.
3		Si se alcanza el límite máximo de búsqueda de música por página se vuelve a recorrer el árbol con una nueva profundidad.
4		Analiza cada archivo musical identificado.
5		Devuelve los metadatos de los archivos musicales encontrados.
6	Ejecuta el filtro para almacenar los metadatos.	

7		Guarda los metadatos de cada archivo musical.
8		Termina el CU.
Requisitos no funcionales		

Tabla 6. Descripción CU Procesar Música (Elaboración propia).

Objetivo		**Extraer la dirección donde se encuentra el archivo musical(URL)**
Actores		Administrador (inicia), Indexador
Resumen		El sistema indexa las URL previamente identificada de donde se encuentra la música en la red.
Complejidad		Alta
Prioridad		Alta
Precondiciones		El administrador ha configurado el sistema y ha iniciado el rastreo.
Postcondiciones		Se han extraído las URL de donde está la música y se ha indexado la dirección en Solr.
Flujo de eventos		
Flujo básico Procesar Música		
Actor		**Sistema**
1	Inicia el rastreo	
2		Realiza Búsqueda de sitios que contengan archivos.mp3
3		Encuentra los sitios donde se encuentra el archivo.mp3
4		Guarda las direcciones de los sitios encontrados en Nutch
5		Almacena la información en Solr.
6		Termina el CU
Requisitos no funcionales		

2.5.2. Diagrama de clases del diseño

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de *software* y de las interfaces en una aplicación, conteniendo las definiciones de las entidades del *software*. A continuación, se representan los diagramas de clases del diseño de los CU que se están analizando primeramente CU Identificar Música y luego CU Procesar Música, la descripción de los restantes casos se encuentran en los anexos **(Ver Anexos 19. y 20.)**. El CU Identificar Música se divide en dos subprocesos, debido a los dos

tipos de búsqueda (simple y avanzada) que se realiza (Sarmiento, 2010). Por tanto, se divide la imagen de clase de diseño del CU Identificar Música para un proceso y luego para el otro proceso:

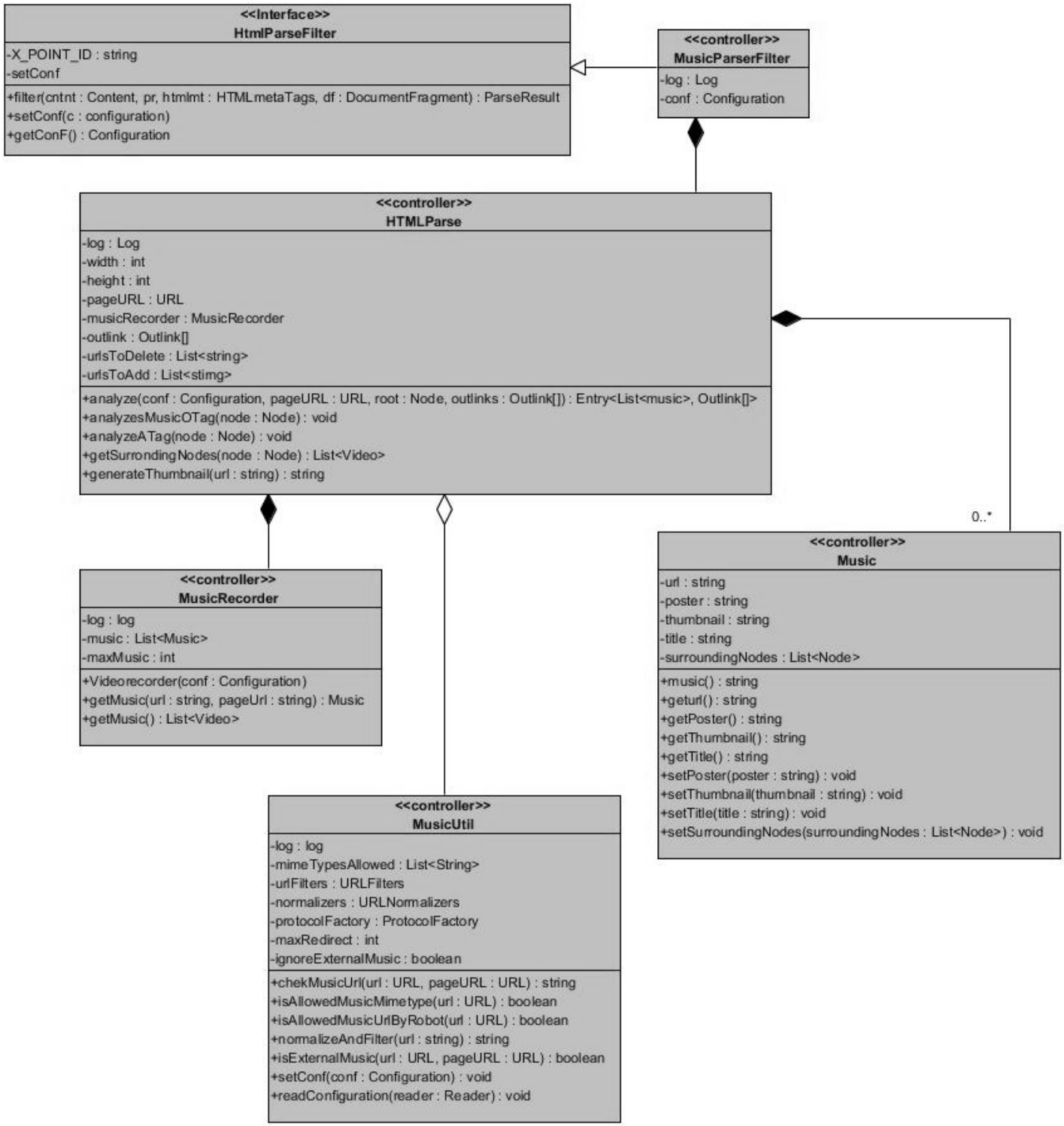

Figura 4. Diagrama de clases del diseño Identificar Música en la página web (Proceso 1) (Elaboración propia).

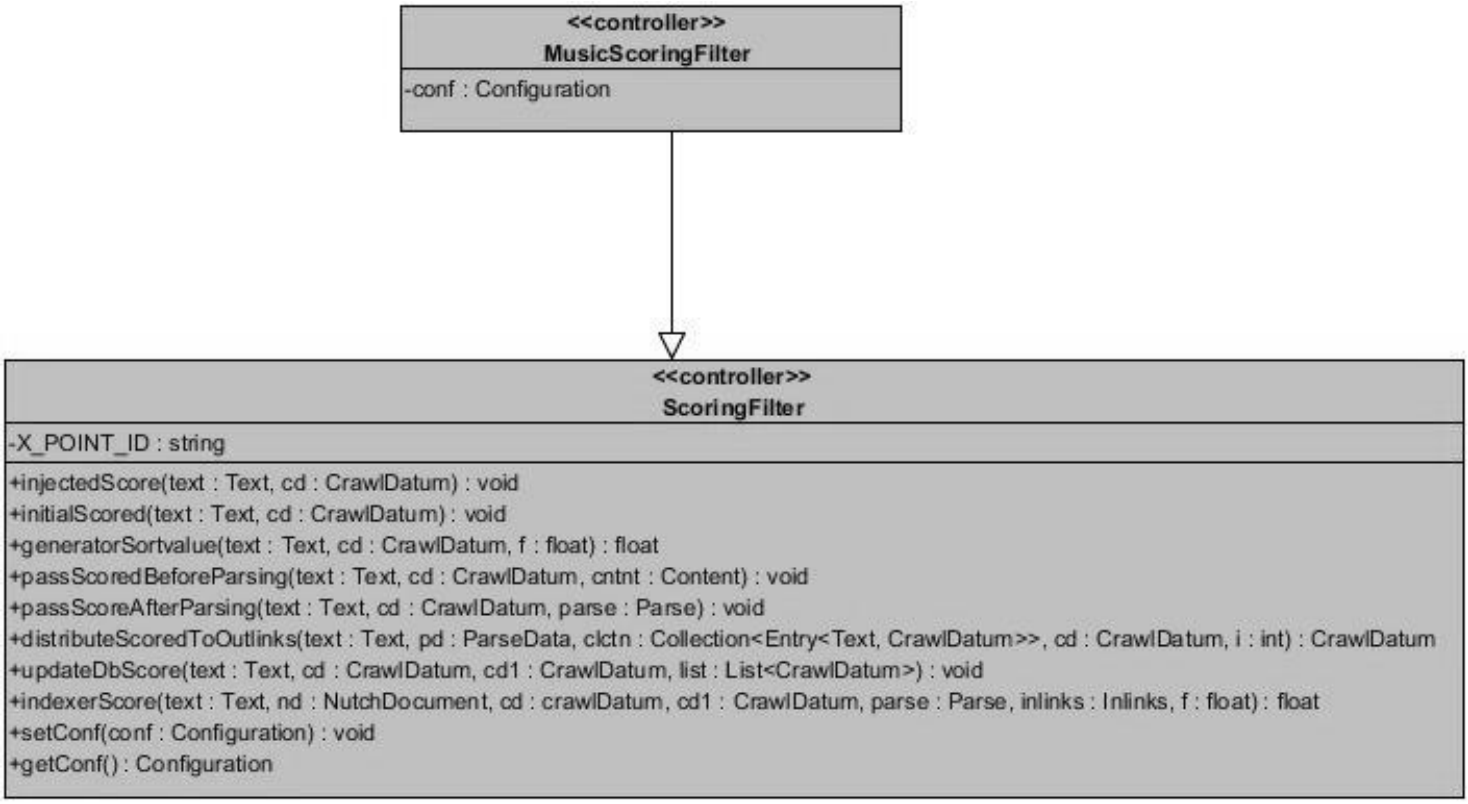

Figura 5. Diagrama de clases del diseño Identificar Música en la página web (Proceso 2) (Elaboración propia).

En el primer diagrama de clases de diseño (Ver figura 4.) se representan las clases que intervienen en el primer proceso de identificar archivos musicales existentes en una página web: La interfaz **HtmlParserFilter** actúa como punto de comunicación entre el rastreador y la clase **MusicParserFilter** que es la encargada de devolver la información de los archivos.mp3 u otros formatos encontrados. **MusicParserFilter** depende de la clase **HTMLParser** que contiene los procedimientos necesarios para identificar los archivos musicales y obtener la información alrededor de ellos. La clase **MusicRecorder** registra todos los archivos musicales encontrados durante el proceso, que en términos de programación serían las instancias de la clase **Music** donde se registra toda la información que se obtiene del archivo musical que representa. **MusicRecorder** y **MusicUtil** depende también **HTMLParser,** para comprobar si se identificó la *URL* de un archivo musical. En el segundo diagrama de clases de diseño (Ver figura 5.) la clase **MusicScoringFilter** es la responsable de asociar la información extraída alrededor de cada archivo musical a su *URL*. La comunicación con el rastreador se lleva a cabo a través de la interfaz **ScoringFilter**.

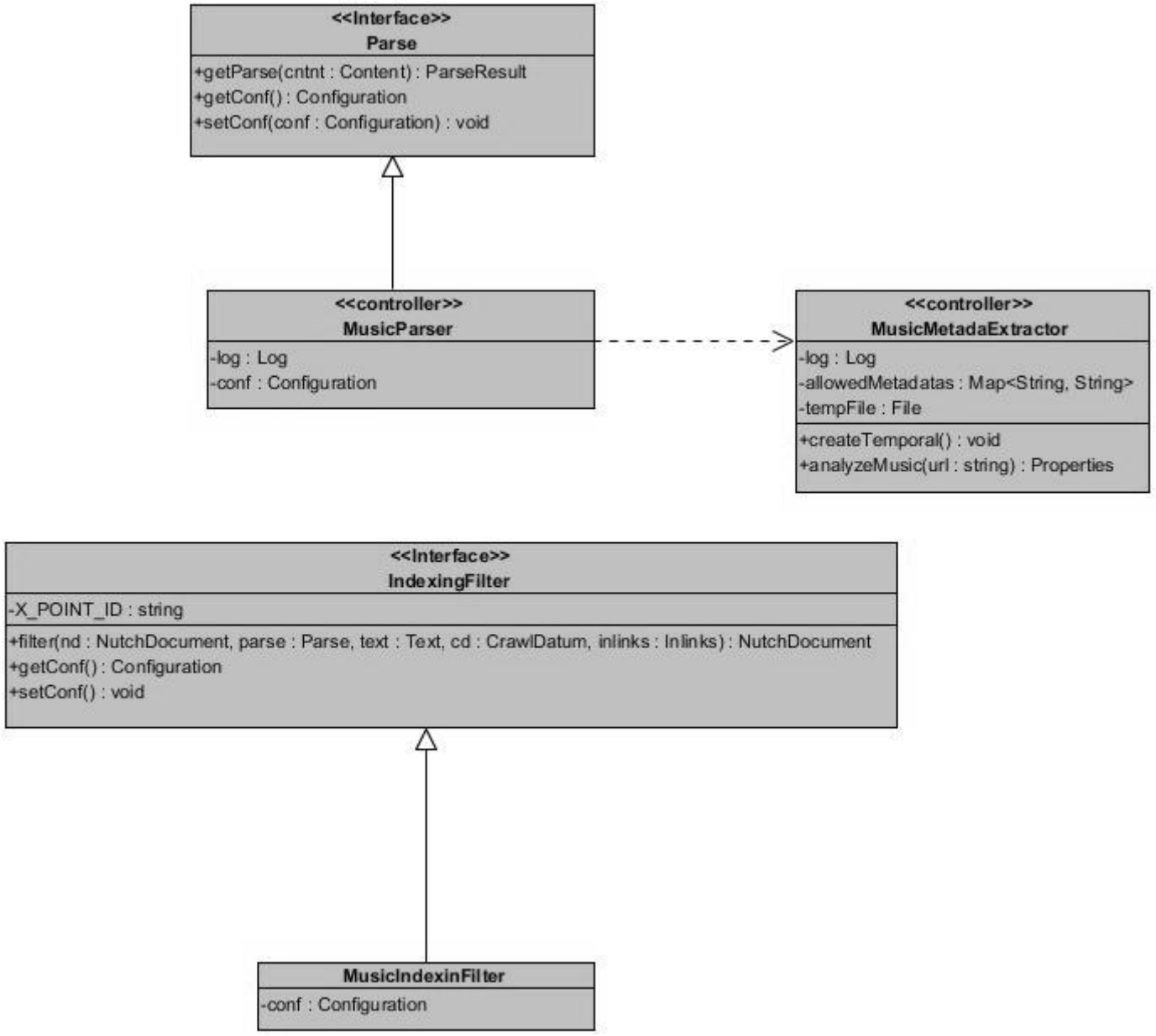

Figura 6. Diagrama de clases del diseño Procesar Música (Elaboración propia).

En el diagrama de clases de diseño para el CU Procesar Música (Ver figura 6.) se representan las clases que intervienen en el proceso de analizar un archivo musical: El rastreador se comunica a través de la interfaz **Parse** con la clase **MusicParser** que es la encargada de devolver los metadatos extraídos del archivo musical. **MusicParser** depende de **MusicMetadataExtractor** para la extracción de los metadatos; los metadatos son procesados en un subproceso independiente por **MusicIndexingFilter** para decidir cuáles metadatos y cómo serán indexados. **MusicIndexingFilter** se comunica con el rastreador mediante la interfaz **IndexingFilter**.

2.6. Diagramas de interacción

Los diagramas de interacción son utilizados para explicar gráficamente las iteraciones existentes entre las instancias y las clases del modelo de las instancias, es decir que modelan los comportamientos que caracterizan un sistema informático. Existes dos diagramas definidos por UML: los diagramas de secuencias y los diagramas de colaboración. Teniendo en cuenta que en los diagramas de colaboración no es necesario representar el tiempo como una dimensión en el intercambio de mensajes, se deciden utilizar los diagramas de colaboración (Sarmiento, 2010).

2.6.1. Diagramas de colaboración

Los diagramas de colaboración describen las interacciones entre los objetos en un formato de grafo o red. A continuación, se muestran los diagramas de colaboración para los CU Identificar Música y Procesar Música:

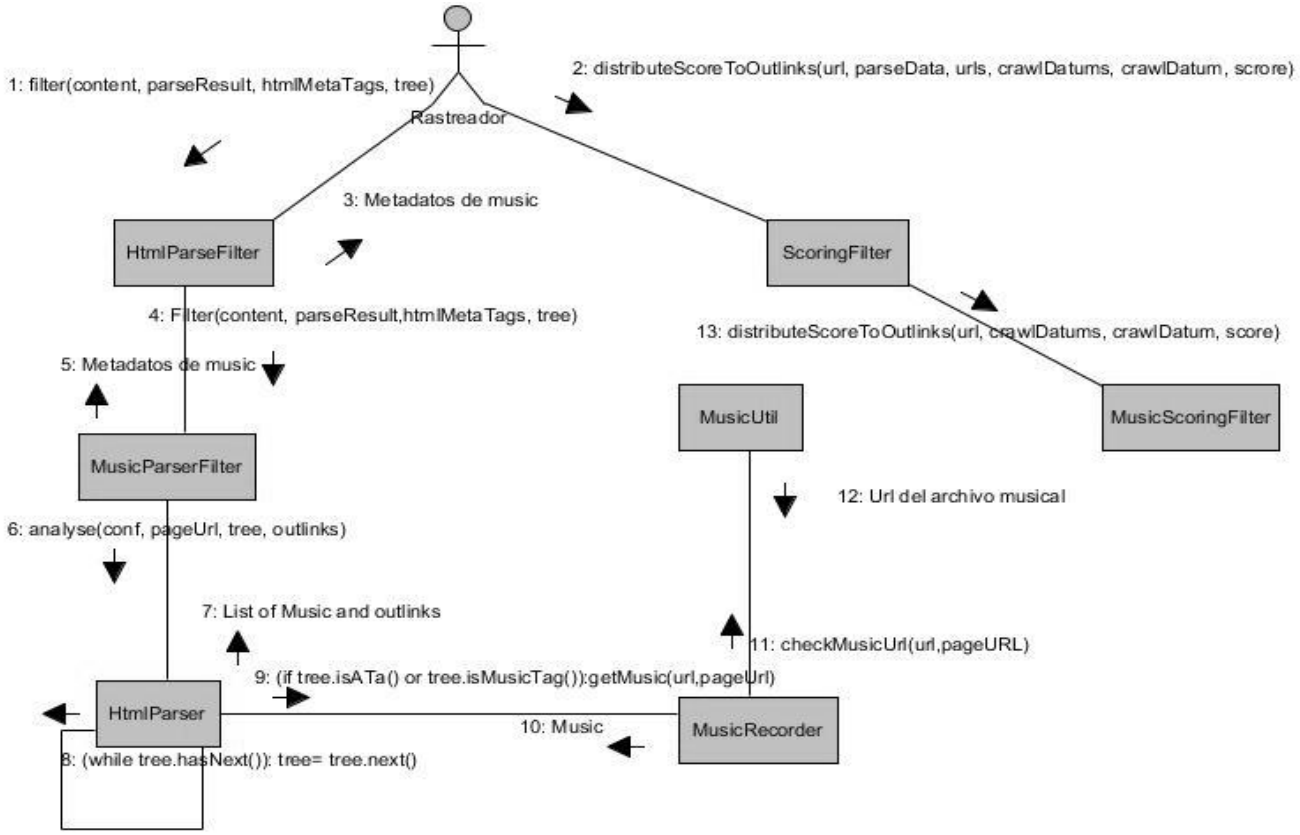

Figura 7. Diagrama de colaboración del CU Identificar Música en la página web.

En el diagrama de colaboración Identificar Música (Ver Figura 7.) se representa el intercambio de mensajes entre las clases que intervienen en el proceso de identificar un archivo musical existente en una página *web*:

El rastreador se comunica con la clase **MusicParserFilte**r enviándole el contenido obtenido, el resultado de los análisis realizados anteriormente y el árbol generado. La comunicación se realiza mediante la interfaz **HtmlParserFilter**, luego **MusicParserFilter** le solicita a la clase **HTMLParser** los archivos musicales y el listado actualizado de las *URLs* que contiene la página *web*. Para que la clase sea capaz de realizar el análisis, se le envía la configuración del sistema, la *URL* de la página, el árbol generado y las *URLs* encontradas hasta el momento. **HTMLParser** realiza un recorrido a lo ancho del árbol y por cada *URL* encontrada le solicita a la clase **MusicRecorder** el archivo.mp3 u otro formato asociado a esta *URL*. Si no existiese alguna *URL* encontrada, la clase **MusicRecorder** se comunica con la clase **MusicUtil** para que compruebe la *URL* que le fue enviada. Si se corresponde con un archivo musical y tiene un formato mp3 u otro establecido, **MusicRecorder** registra un nuevo archivo musical y se lo envía a **HTMLParser** para que continúe el proceso. Una vez obtenida la información que se encuentra alrededor de los archivos musicales, el rastreador se comunica con la clase **MusicScoringFilter** a través de la interfaz **ScoringFilter** para asociarle a cada archivo musical encontrado la información que le corresponde. **ScoringFilter** le envía la *URL* de la página *web* analizada, la información obtenida durante el proceso anterior y la *URL* de cada enlace contenido en la página.

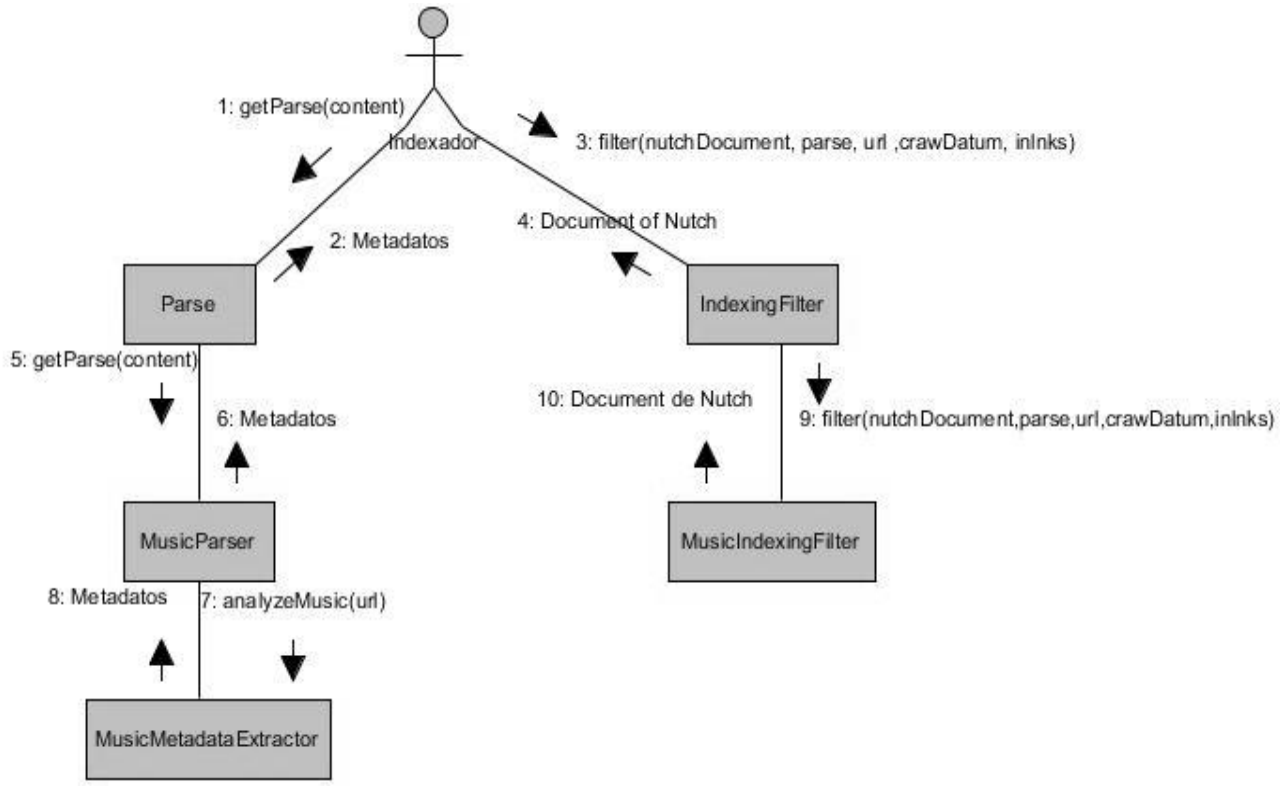

Figura 8. Diagrama de colaboración del CU Procesar Música.

En el diagrama de colaboración Procesar Música (Ver Figura 8.) se representa el intercambio de mensajes entre las clases durante el proceso analizar un archivo musical: El rastreador una vez que ha obtenido el contenido de un archivo.mp3 u otro formato se lo envía a la clase **MusicParser** mediante la interfaz **Parse** para que sea procesado. **MusicParser** se comunica con la clase **MusicMetadataExtractor** para que obtenga los metadatos del archivo musical, facilitándole la *URL* del mismo. Una vez obtenidos los metadatos, el rastreador le solicita a la clase **MusicIndexingFilter** a través de la interfaz **IndexingFilter** que seleccione los metadatos que son necesarios para indexar la información del archivo musical que se está analizando. Luego, le envía la información obtenida del archivo musical y su *URL*.

2.7. Patrones utilizados en el desarrollo del software

Un patrón constituye una descripción de un problema y la solución de este, que recibe un nombre y que puede emplearse en otros contextos (Sarmiento, 2010). A continuación, se explica cada patrón utilizado en el desarrollo del proyecto.

2.7.1. Patrón de casos de uso

Para realizar el modelado de los casos de uso del sistema y su correcta estructuración y organización se utilizan los patrones de casos de usos, estos son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo interactúa con los usuarios (Sarmiento, 2010). En el desarrollo de la investigación se identificaron los siguientes patrones de casos de uso:

Relación Inclusión

Este patrón indica que cuando se tiene algún caso de uso que tiene una parte del fragmento con un comportamiento parcial común a varios casos de uso, se incluye a un nuevo caso de uso y se indica su inclusión para evitar duplicaciones (Sarmiento, 2010).

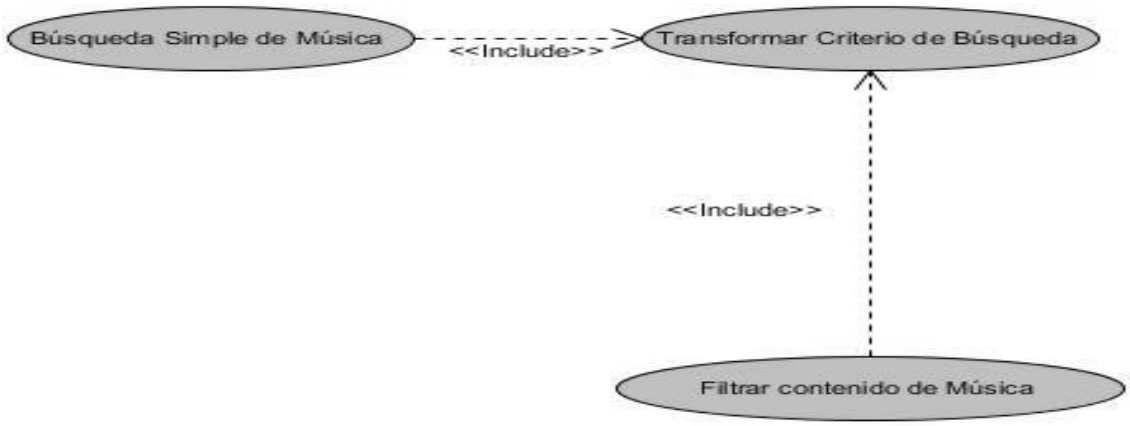

Figura 9. Ejemplo del Patrón Relación Inclusión.

Dentro de diagrama de Modelo de caso de uso del sistema (Ver figura 9.) el patrón Relación Inclusión es utilizado para incluir el CU Transformar criterio de búsqueda en el CU Búsqueda de música y el CU Filtrar contenidos de música. Tanto el CU Búsqueda de música y el CU Filtrar contenidos de música presentan un comportamiento común, que consiste en transformar un criterio de búsqueda en una consulta para el indexador. Por lo que se decide incluir a este comportamiento en un CU independiente.

2.7.2. Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como la forma como estos colaboran entre sí (Sarmiento, 2010). Para el diseño de la propuesta de solución se utilizaron los Patrones Generales de Software para la Asignación de Responsabilidades por sus siglas (GRASP), los cuales describen los principios fundamentales de la asignación de responsabilidades a los objetos y generalmente son asignados en el momento de modelar los diagramas de interacción (Sarmiento, 2010). A continuación, se describe los principales patrones GRASP:

Experto en Información

Este patrón plantea que se debe asignar una responsabilidad a la clase que cuenta con los datos necesarios para cumplir con una determinada responsabilidad, conservándose el encapsulamiento de la información, puesto que los objetos ejecutan las tareas que le corresponden de acuerdo a la información que poseen. Lo que da lugar a sistemas más robustos y fáciles de mantener (Sarmiento, 2010).

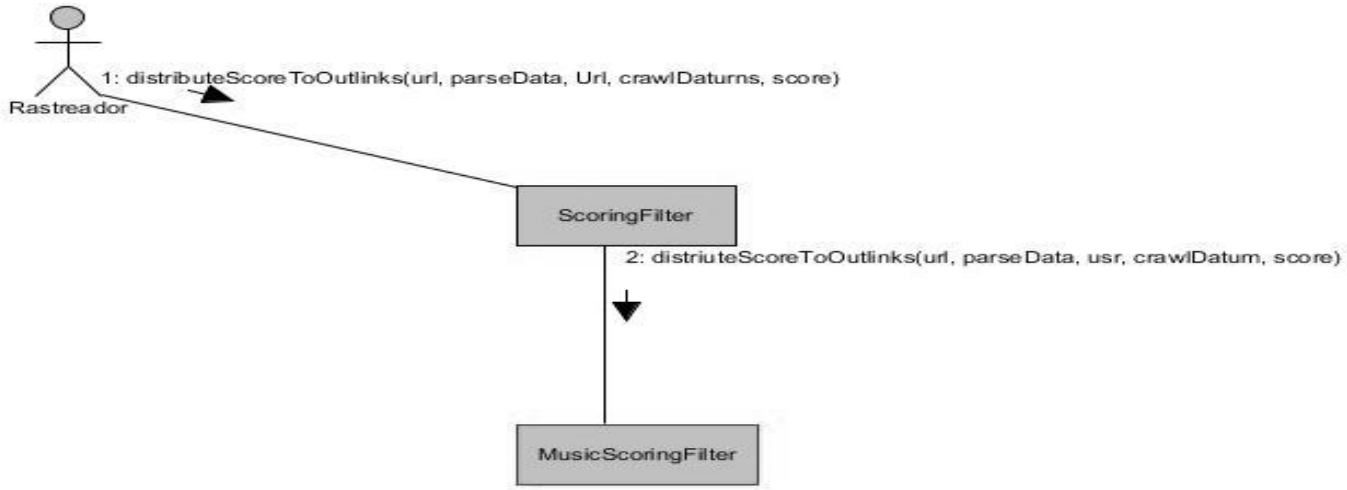

Figura 10. Ejemplo del patrón experto en información.

Dentro de diagrama de colaboración del CU Identificar Música en la página *web* (Ver figura 10.) se utiliza el patrón Experto en Información donde se le asignan responsabilidades determinadas solamente a la clase **MúsicaScoringFilteres**, puesto que conoce la información necesaria y su adecuada manipulación para distribuirla a todos los enlaces salientes de una página *web,* mintiéndose el encapsulamiento de la información que se utiliza para realizar las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener.

Bajo acoplamiento

El patón Bajo Acoplamiento constituye una medición de la fuerza de conexión entre las clases. Es decir, una clase con un fuerte acoplamiento recurre a muchas otras y no es conveniente su existencia ya que son más difíciles de reutilizar porque se requiere la presencia de las clases de las que dependen (Sarmiento, 2010). Dentro de diagrama de colaboración del CU Identificar Música en la página *web* (Ver Figura 7.) se representa el patrón Bajo Acoplamiento en la clase **MusicRecorder** donde se propone asignar las responsabilidades de tal manera que permita lograr un bajo acoplamiento entre las clases involucradas a esta.

Alta cohesión

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y enfocan las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo (Sarmiento, 2010). Dentro de diagrama de colaboración del CU Identificar Música en la página *web* (Ver Figura 7.) se representa el patrón Alta Cohesión en la clase **MusicUtil**, donde la clase asigna responsabilidades a las clases involucradas a esta.

Fabricación pura

El patrón Fabricación Pura tiene como objetivo proponer y asignar un conjunto cohesivo de responsabilidades a una clase artificial que no representa nada en el dominio del problema, pero da soporte a una alta cohesión, bajo acoplamiento y reutilización (Sarmiento, 2010). Dentro de diagrama de colaboración del CU Identificar Música en la página *web* (Ver Figura 7.) se manifiesta el patrón Fabricación Pura en la clase **MusicRecorder**, donde la clase fue creada dentro del Diagrama de Colaboración con el objetivo de controlar los archivos musicales encontrados en una página *web* y suprimirle las responsabilidades a la clase **HTMLParser** logrando un diseño con bajo acoplamiento y alta cohesión.

Creador

El patrón Creador simboliza la creación de una clase con el objetivo de asignarle a esta la responsabilidad de crear instancias cuando a las clases restantes se le agrega responsabilidades, es decir que el patrón

propone asignarle a una clase B la responsabilidad de crear una instancia de la clase A cuando B agrega, contiene, registra o utiliza las instancias de A o tiene los datos de inicialización que serán transmitidos hacia A cuando este objeto sea creado (Sarmiento, 2010). Dentro de diagrama de colaboración del CU Identificar Música en la página *web* (Ver Figura 7.) se manifiesta el patrón Creador en la clase **MusicRecorder**, donde la clase tiene la responsabilidad de registrar todos los archivos musicales encontrados durante el análisis de la página *web*.

2.8. Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama de *UML* que se utiliza para modelar la arquitectura de un sistema por nodos y las relaciones entre ellos. A continuación, se muestra el diagrama de despliegue propuesto para el subsistema de búsqueda de música para la plataforma c.u.b.a.

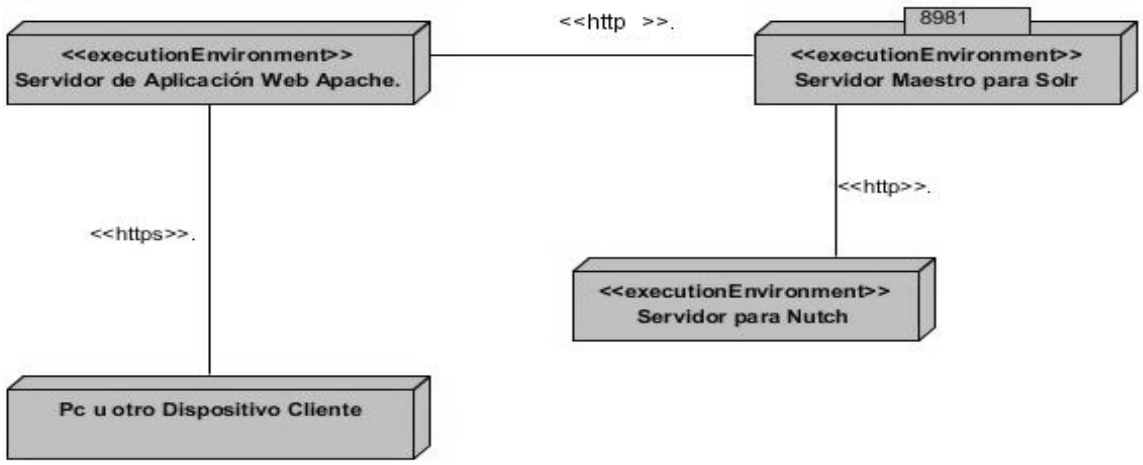

Figura 11. Diagrama de Despliegue (Elaboración propia).

En la figura anterior (ver Figura 11.) el nodo **Servidor de aplicaciones web Apache** es el encargado de atender y ofrecer respuesta a cada una de las solicitudes del cliente. El nodo **Pc cliente u otro dispositivo Cliente** representa el dispositivo utilizado por el usuario desde el cual se podrán realizar las búsquedas de archivos musicales, a través del protocolo HTTPS, haciendo uso de un navegador *web*. El nodo **Servidor**

maestro para Solr y el nodo **Servidor para Nutch** se deben de encontrar en servidores diferentes con el objetivo de utilizar al máximo las características de *hardware* y *software* de estos.

Conclusiones del capítulo

En presente capítulo se abordaron una serie de aspectos correspondientes al análisis y diseño de la búsqueda de archivos musicales para la plataforma c.u.b.a. llegándose a las siguientes conclusiones:

- La elaboración del modelo de dominio y su descripción permitió una mayor comprensión del sistema que se propone desarrollar.
- La representación y descripción de los artefactos generados garantizaron un mejor entendimiento de los flujos de trabajos presentes en el proceso de búsqueda de archivos musicales.
- La especificación de los requisitos funcionales y no funcionales del sistema dieron paso a una mejor comprensión de los resultados que se pretende obtener de una manera precisa y sirvieron de guía para la implementación del sistema.
- La definición de la arquitectura y los patrones de diseño a utilizar, permitieron establecer las bases para fomentar la reutilización y las buenas prácticas de programación durante la fase de implementación, así como disminuir el impacto de los cambios futuros en el código fuente.
- La elaboración de los diagramas de clases del diseño y los diagramas de colaboración propició una mayor comprensión de la distribución y asignación de responsabilidades de las clases involucradas en los casos de uso analizados.
- La elaboración del diagrama de despliegue permitió identificar la disposición física de los artefactos de la herramienta informática a desarrollarse.

CAPÍTULO 3: Implementación y prueba del subsistema de búsqueda de música para la plataforma c.u.b.a.

La fase de implementación en el desarrollo de un producto de *software* es una de las fases imprescindibles dentro del proceso de desarrollo de *software*. Dicha frase comprende la materialización precisa en forma de código de todos los artefactos, descripciones y arquitecturas propuestas en las fases de análisis y diseño. Además, tiene el objetivo de conformar el producto final requerido por el cliente.

La fase de prueba constituye una de las últimas fases del ciclo de vida antes de entregar el proyecto para su explotación, cuyo objetivo es comprobar si el proyecto cumple con los requisitos funcionales y no funcionales identificados en la etapa de análisis y diseño (Larman C, 2004). Dentro de ambas fases (implementación y prueba) se generan los artefactos correspondientes: diagrama de componentes, los estándares de codificación y los casos de prueba. Asimismo, se explica la implementación del *plugin* de *Nutch* y se realiza la validación de la hipótesis de investigación y el correcto funcionamiento del subsistema integrado a la plataforma c.u.b.a.

3.1. Modelo de componentes que integran la solución informática

El modelo de componentes representa la forma en que es estructurado un sistema informático atendiendo a las diferentes partes que lo componen. Partiendo de este punto, Sommerville puntualiza que cada componente debe ser tratado como una unidad de composición independiente e indispensable dentro de un sistema, y que puede contraer relaciones de dependencia con otros componentes. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, librerías, ejecutables, binarios, entre otros (Sommerville, 2005).

3.1.1. Diagrama de componentes

Los diagramas de componentes permiten visualizar con facilidad las partes o componentes físicos y reemplazables de un sistema, así como las relaciones existentes entre ellos (IEEE., 2004). A continuación, se muestra el diagrama de componentes del subsistema de búsqueda de música para la plataforma c.u.b.a., cuya organización se encuentra acorde con la arquitectura MVC que implementa el componente del subsistema aplicación *web*:

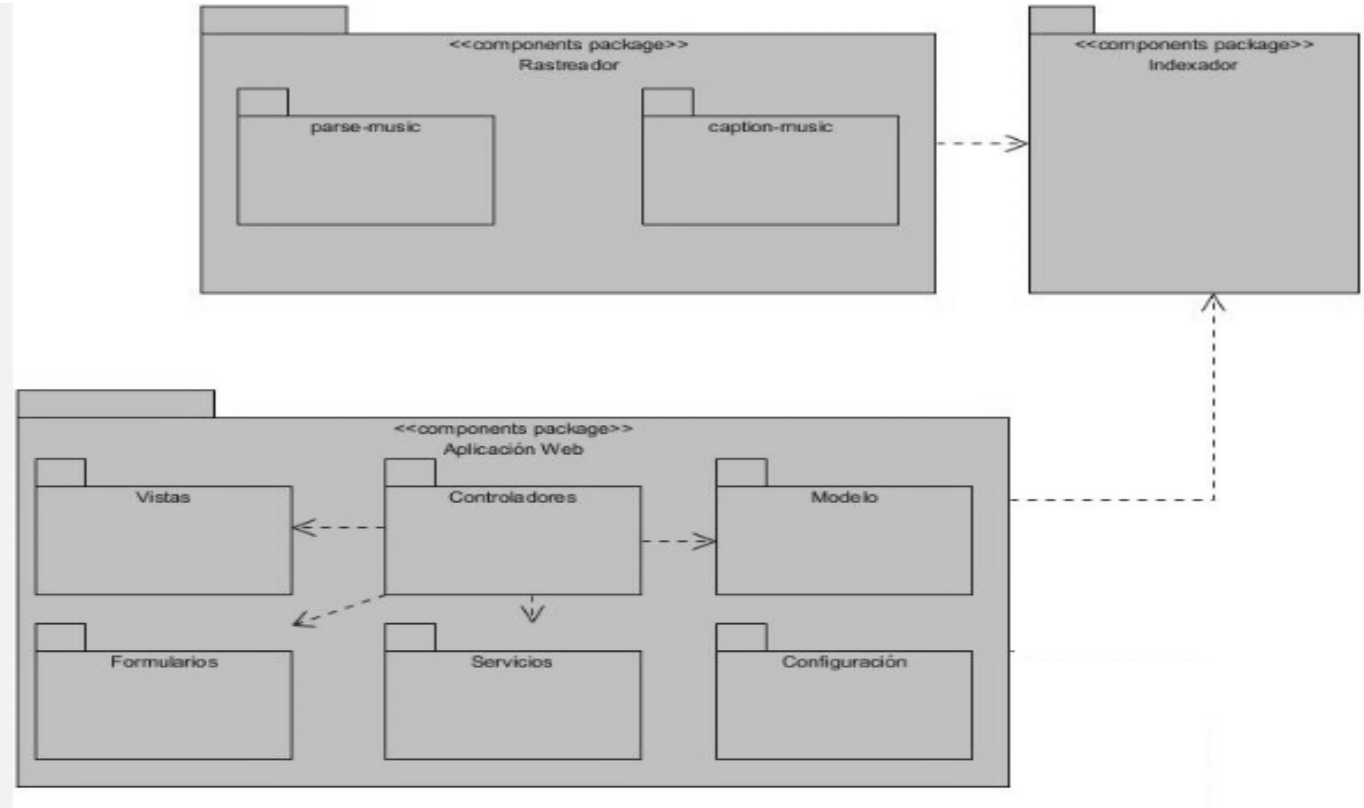

Figura 12. Diagrama de Componentes del Sistema (Elaboración propia).

El Diagrama de Componentes mostrado anteriormente (Ver figura 12.) está compuesto por tres paquetes, los cuales representan los componentes de la arquitectura del subsistema de búsqueda de música para la plataforma c.u.b.a. donde:

- **Rastreador**: Implementa la lógica de la búsqueda de música en la web cubana.
- **Indexador**: Contiene los componentes de configuración que permitirán tanto la indexación de documentos como la comunicación con el rastreador y la aplicación web.
- **Aplicación web**: Incluye los paquetes que contienen los controladores, vistas, modelos, formularios, servicios y ficheros de configuración de la aplicación web.

El paquete **Rastreador** está dividido en dos subpaquetes: **Parse-Music** y **Caption-Music**, los cuales representan los *plugins* desarrollados para el componente de rastreo en *Nutch*. A continuación, se muestra la estructura interna del paquete:

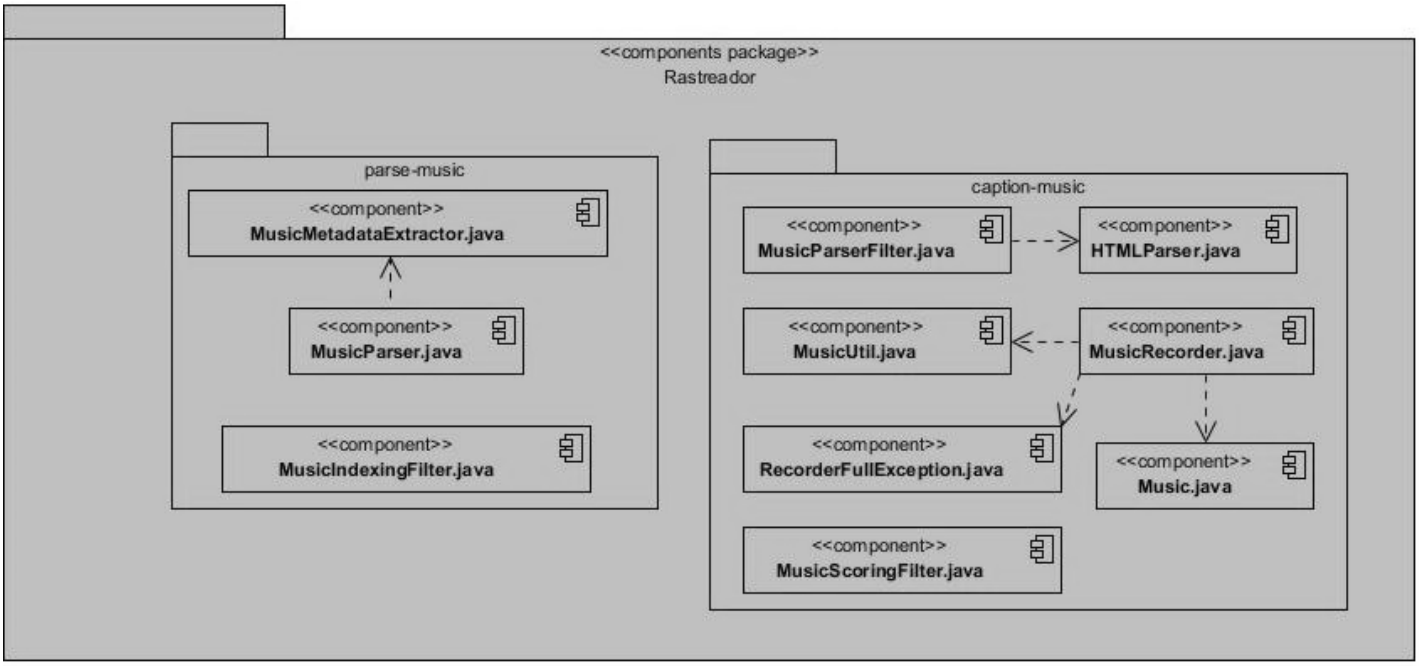

Figura 13. Diagrama de Componentes del paquete Rastreador (Elaboración propia).

En el Diagrama de Componentes mostrado anteriormente (Ver Figura 13.) se muestran los dos subpaquetes del paquete **Rastreador**. Dentro de **Parse-Music** se incluyen tres componentes y dentro de **Caption-Music** se incluyen siete componentes. A continuación, se explican los componentes:

Parse-Music

MusicMetadataExtractor.java: Es el fichero que contiene la clase **MusicMetadataExtractor**, la cual es utilizada para la extracción de los metadatos de los archivos musicales.

MusicParser.java: Representa al fichero que contiene la clase **MusicParser**, cuya función principal es interactuar con el rastreador y enviarle los metadatos extraídos de un archivo musical.

MusicIndexingFilter.java: Componente físico que incluye la clase **MusicIndexingFilter**, la cual tiene la responsabilidad de decidir cuáles metadatos indexar y cómo serán indexados los metadatos.

Caption-Music

MusicParserFilter.java: Representa al fichero que contiene la clase **MusicParserFilter**, la cual contiene los procedimientos necesarios para gestionar la información de los archivos musicales encontrados en una página *web.*

HTMLParser.java: Es el fichero que contiene a la clase **HTMLParser**, utilizada para identificar los archivos musicales en una página web y obtener la información alrededor de ellos.

MusicRecorder.java: Componente que contiene la clase **MusicRecorder**, la cual registra todos los archivos musicales encontrados durante el proceso de análisis de una página *web*.

Music.java: Componente que contiene la clase **Music**, la cual es utilizada para intercambiar la información de los archivos musicales encontrados en la *web*.

MusicUtil.java: Representa al fichero que contiene la clase **MusicUtil**, la cual es utilizada para comprobar si una *URL* representa a un archivo musical en la red.

RecorderFullException.java: Componente físico que contiene la clase **RecorderFullException**, utilizado para indicar que no se pueden registrar más archivos musicales nuevos, ya que la configuración del sistema no lo permite.

MusicScoringFilter.java: Indica el fichero que contiene la clase **MusicScoringFilter,** la cual tiene la responsabilidad de asociar la información extraída del archivo musical a su *URL*.

El paquete **Indexador** contiene tres componentes que garantizan el correcto almacenamiento de los datos y la comunicación con los restantes componentes del sistema. A continuación, se muestra la estructura interna del paquete:

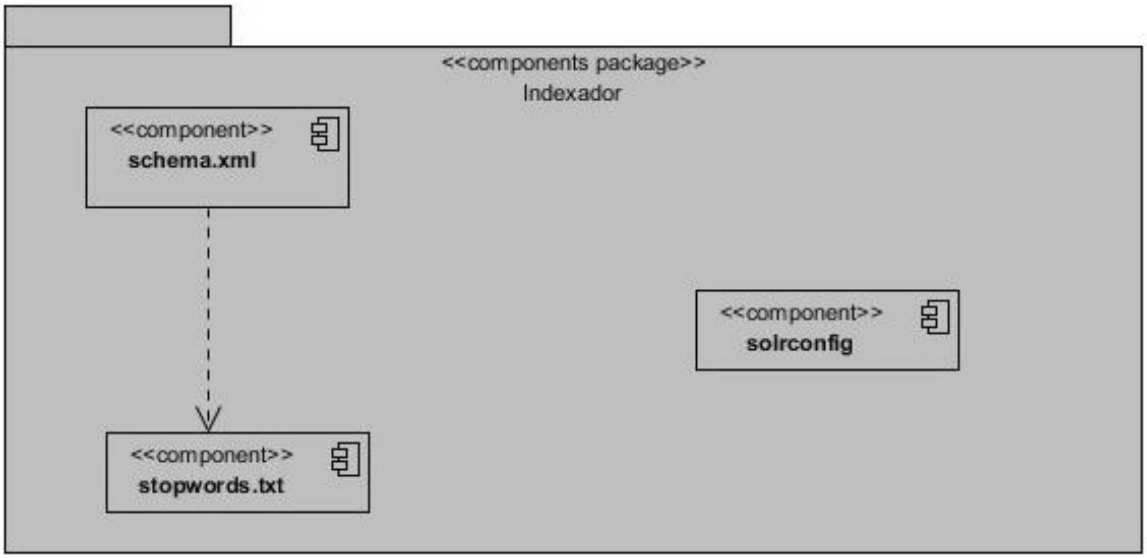

Figura 14. Diagrama de Componentes del paquete Indexador (Elaboración propia).

En el Diagrama de Componentes mostrado anteriormente (Ver Figura 14.) se muestran los componentes:

schema.xml: Contiene los campos y los tipos de campos de un documento, así como los filtros que se le aplican a los tipos de documentos, que ayudan a facilitar la búsqueda y la indexación de los documentos.

stopwords.txt: Contiene palabras por las que no se realizan búsquedas de documentos.

solrconfig.xml: Fichero de configuración principal de *Solr.*

El paquete **Aplicación web** está dividido en seis subpaquetes: **Vistas**, **Controladores**, **Formularios**, **Servicios**, **Modelo** y **Configuración**, debido al tipo de arquitectura descrita en el capítulo 2 del componente. A continuación, se muestra la estructura interna del paquete:

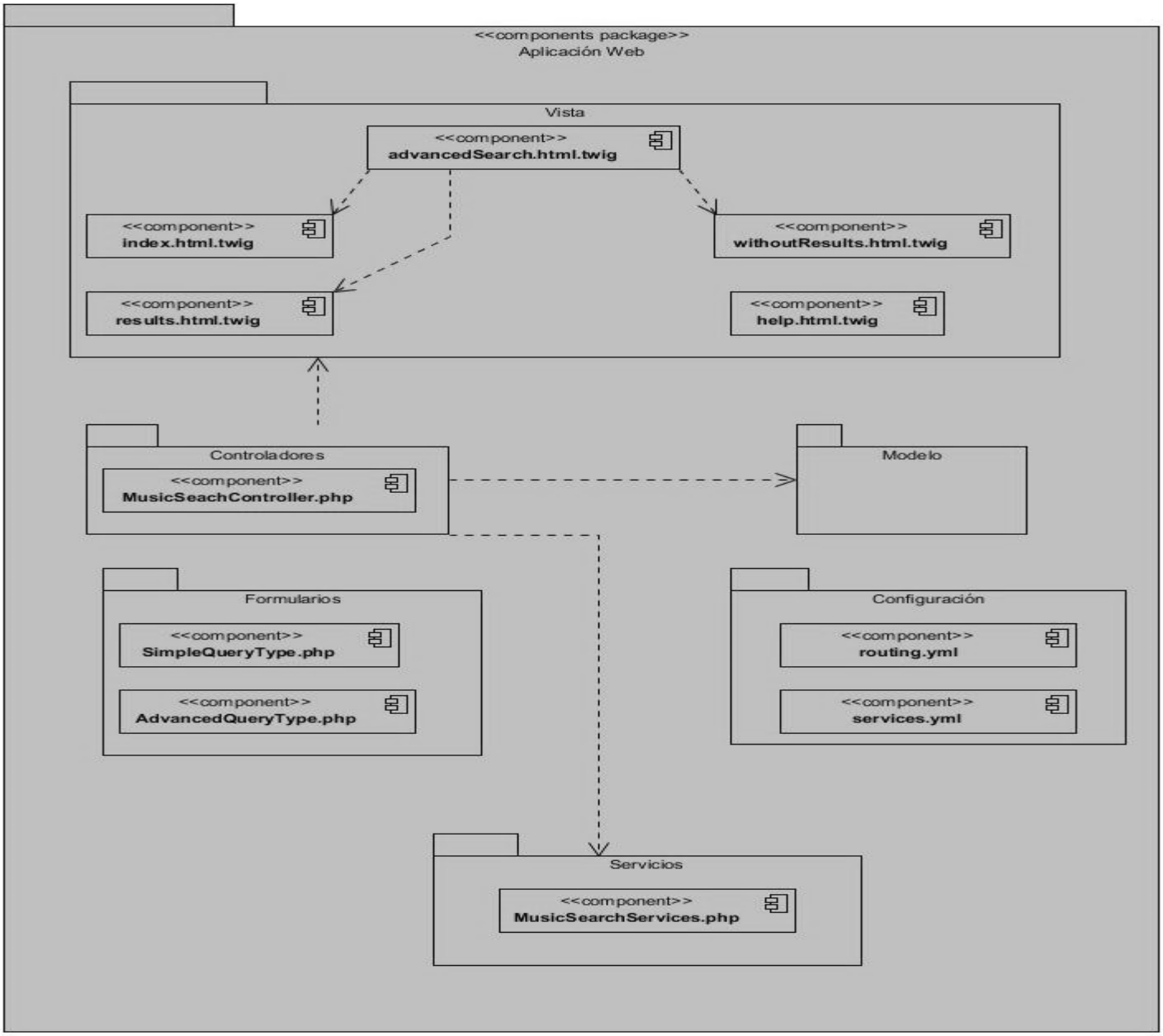

Figura 15. Diagrama de Componentes del paquete Aplicación web (Elaboración propia).

En el Diagrama de Componentes mostrado anteriormente (Ver Figura 16.) se muestran los seis subpaquetes del paquete **Aplicación Web**. Dentro de **Vistas** se incluyen seis componentes, dentro **Formularios** y **Configuración** se incluyen dos componentes respectivamente, dentro **Servicios** y **Controladores** se incluyen un componente respectivamente y dentro de **Modelo** no se incluye ningún componente. A continuación, se explican:

El paquete **Vista** incluye los componentes que el sistema le muestra al usuario.

index.html.twig: Es la vista principal del sistema, la cual muestra los formularios de búsqueda simple y búsqueda avanzada.

results.html.twig: Es la vista donde se muestran los resultados, así como los formularios de búsqueda simple y de búsqueda avanzada.

advancedSearch.html.twig: Es la vista que contiene el formulario de búsqueda avanzada, la cual es incluida en la vista principal y la vista de resultados del sistema.

withoutResults.html.twig: Es la vista que se muestra cuando el usuario no ha introducido criterios de búsqueda mostrándole en la misma un mensaje de error, así como los formularios de búsqueda simple y de búsqueda avanzada

help.html.twig: Es la vista de ayuda del sistema.

El paquete **Formulario** incluye los componentes para que el usuario pueda realizar búsqueda (simple y avanzada) en el sistema.

SimpleQueryType.php: Contiene la clase **SimpleQueryType**, la cual representa la estructura del formulario de búsqueda simple.

AdvancedQueryType.php: Contiene la clase **AdvancedQueryType**, la cual representa la estructura del formulario de búsqueda avanzada.

El paquete **Configuración** incluye los componentes correspondientes a la aplicación web.

routing.yml: Contiene las rutas que utilizará el sistema.

services.yml: Contiene la configuración de los servicios del sistema.

El paquete **Servicios** incluye el componente donde se va a realizar las peticiones de música.

MusicSearchServices.php: Contiene la clase **MusicSearchServices**, encargada de realizar las peticiones a Solr a partir de la consulta dada por el usuario.

El paquete **Controladores** incluye el componente donde se va a controlar el sistema.

MusicSearchController.php: Contiene la clase **MusicSearchController**, encargada de controlar las peticiones de los usuarios, crear las vistas correspondientes a cada una de ellas y manejar los servicios del sistema, así como los formularios de búsqueda.

3.2. Configuración de Nutch

Para realizar las operaciones pertinentes a la implementación de los nuevos *plugins* en *Nutch* se debe de tener en cuenta los directorios principales que contienen los archivos de configuración en el rastreador: **Conf**, **Plugins** y **Urls**, estos archivos de configuración, distribuidos por toda su estructura de directorios se encuentran por lo general en formato XML, pero puede existir el caso que algunos *plugin* ya desarrollados en *Nutch* estén en diferentes formatos (XSD, XLS, DTD o TXT), lo que se recomienda que para la implementación de los *plugins* tomar uno ya desarrollado en XML e irlo configurarlo. *Nutch* tiene consigo algunos metadatos asociados al contenido del archivo a la hora de hacer el rastreo de música e indexarlo en *Solr* como: la *URL* donde se encuentra, artista, título y formato **(Ver Anexo 15.)**. Se tiene que saber ¿qué metadatos reconoce la librería **JavaMusic** para configurar los nuevos *plugins?* **(Ver Anexos 16., 21. y 22.)**. En este caso se implementará las etiquetas para filtrar por título, formato y autor del archivo *musical*.

3.3. Estándares de codificación utilizados

Los estándares de codificación son especificaciones o estilos que establecen la forma de generar el código funcional de las aplicaciones informáticas. Puesto que, en muchas ocasiones, los sistemas de cómputo son implementados por varios programadores, la adopción inicial de un único estilo de codificación constituye uno de los factores de mayor peso en la calidad, rendimiento, legibilidad y capacidad de mantenimiento del producto final. Para facilitar el entendimiento del código, se adoptaron determinados estándares de codificación que a continuación se describen, agrupados por los aspectos en los que fueron utilizados:

- En el caso de la aplicación web se utiliza para el lenguaje PHP el que establece el marco de trabajo *Symfony2* que sigue los estándares definidos en los documentos PSR-1 (PHP Framework Interop Group (PSR-1)., 2014), PSR-2 (PHP Framework Interop Group (PSR-2)., 2014), PSR-3 (SensioLabsNetework., 2014), PSR-4 (PHP Framework Interop Group (PSR-4)., 2014).
- Para la implementación de los plugins para Nutch se emplea el estándar definido para el lenguaje Java entre los elementos más relevantes se encuentran:

    1. **Identificadores:** Para la definición del nombre de las clases, funciones y variables se tiene en cuenta el estilo ***lowerCamelCase***. Este estilo establece que la separación entre palabras internas de los identificadores deberá realizarse escribiendo la letra inicial en mayúscula, a excepción de la primera palabra. Además, no deberá colocarse ningún carácter especial entre palabras de los identificadores. Ejemplo Clase: ***class*** archivoController.

2. **Indentación:** Es uno de los estándares más recomendados para la implementación. Este enfatiza en comenzar a escribir cada línea de código a diferentes distancias desde el borde izquierdo del área de edición. La distancia deberá regirse por la jerarquía que se forma al introducir sentencias dentro de bloques de estructuras. Gracias al uso de *NetBeans* como IDE de desarrollo, los espacios de indentación son ajustados automáticamente.
3. **Llaves:** En la presente investigación las llaves de apertura se colocarán inmediatamente al final de la línea de cabecera del bloque, así como en las estructuras *if, for, while, else, switch, foreach.* Las llaves de cierre se colocarán solitarias en la línea siguiente a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque.
4. **Comentarios:** Los comentarios en el código representan la documentación interna más precisa de un *software.* Estos garantizan el entendimiento de lo que realmente realiza un determinado bloque de código, evitando confusiones y agilizando considerablemente las tareas de revisión y mantenimiento. Para la inclusión de comentarios es necesario respetar algunas reglas básicas como: abreviar el contenido de los comentarios e incluirse un espacio simple entre los caracteres "//" y el texto del comentario.

3.4. Validación del subsistema de búsqueda de música para la plataforma c.u.b.a.

La validación del sistema incluye un conjunto de actividades para asegurar que el *software* desarrollado se corresponde con los requisitos del cliente. A continuación, se detallan los tipos de pruebas de *software* aplicados al subsistema de búsqueda de archivos musicales. Las mismas, persiguen como objetivo fundamental, la detección de las no conformidades respecto a las funcionalidades de la aplicación. Además de las vulnerabilidades que atentan contra la seguridad de la información que se manipula con el *software*, la medición del grado de usabilidad de las funcionalidades implementadas, así como también la correcta integración entre los diferentes componentes de la arquitectura del sistema.

3.4.1. Pruebas funcionales

Las pruebas funcionales son aquellas que se aplican a un *software* determinado, con el objetivo de identificar situaciones que no se ajustan a las especificaciones funcionales establecidas en la fase de análisis del proceso de desarrollo del *software*. A continuación, se exponen los resultados de estas pruebas luego de haber realizado los casos de pruebas correspondientes al CU crítico "Identificar Música en una página *web*" **(Ver Anexo 23. y 24.)**.

Con el objetivo de probar el correcto funcionamiento de las funcionalidades del sistema se realizaron 4 iteraciones de pruebas. En la siguiente tabla (Ver tabla 7.) se muestran los resultados obtenidos en cada iteración de prueba al subsistema de búsqueda de música, así como la corrección de cada uno de los errores. Donde en una primera iteración se detectaron 12 errores: 7 en la configuración de *Solr* y 5 en el proceso de Rastreo, los 12 errores fueron resueltos. Para una segunda iteración se detectaron 15 errores: 9 respecto a la *interface* y 6 con el rastreador *Nutch*, todos los errores fueron solucionados. En una tercera iteración se detectaron 4 errores en la interface, y al igual que en las restantes interacciones fueron solucionados. En una cuarta iteración no se detectó ningún error, funcionando correctamente cada componente del subsistema.

Tabla 7. Cantidad de no conformidades identificadas por cada iteración" (Elaboración propia).

No Conformidades	1ra Iteración	2da Iteración	3ra Iteración	4ta Iteración
Identificadas	**12**	**15**	**4**	**0**
Resueltas	**12**	**15**	**4**	**0**
Pendientes	**0**	**0**	**0**	**0**

En la siguiente figura (Ver figura 16.) se puede apreciar el comportamiento de las no conformidades de las pruebas funcionales ejecutadas.

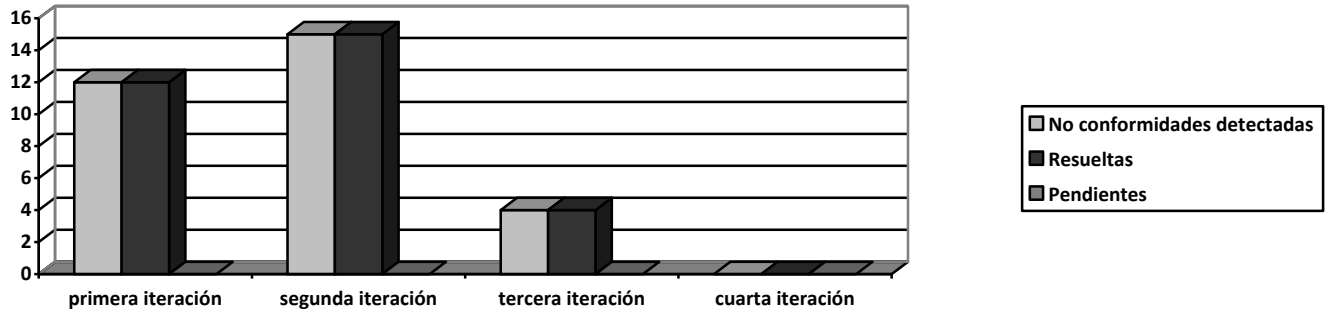

Figura 16. Comportamiento de las no conformidades para cada iteración de las pruebas funcionales (Elaboración propia).

3.4.2. Pruebas de integración

Una vez realizadas las pruebas funcionales a cada componente interno de manera independiente, y verificando que las funcionalidades implementadas se corresponden de acuerdo a los requisitos funcionales y no funcionales establecidos; se pudo comprobar el correcto funcionamiento de los componentes mediante el estudio del flujo de datos entre ellos. Posterior a estas pruebas, se hace necesaria la realización de pruebas de integración.

Las pruebas de integración son definidas para verificar el correcto ensamblaje entre los distintos módulos que conforman un sistema informático. Las mismas validan que estos componentes realmente funcionan juntos, sean llamados correctamente y transfieran los datos correctos en el tiempo preciso y por las vías de comunicación establecidas (Sommerville, 2005).

Para la realización de las pruebas de integración se llevaron a cabo diferentes acciones:

- Verificación de la conexión de *Nutch* y *Solr* para indexar los metadatos extraídos del archivo musical **(Ver Anexo 17.)**.
- Comprobar el enlace entre *Solr* y la interfaz de resultados para verificar si se muestran todos los archivos musicales almacenados en *Solr* **(Ver Anexo 15. y 18.)**.
- Comprobar la integración del subsistema de búsqueda de archivos musicales con la plataforma c.u.b.a., donde la plataforma cuenta con una estructura que facilita la integración de nuevos subsistemas para incrementar sus funcionalidades. Por lo planteado anteriormente se realizan pruebas descendentes, que consisten en desarrollar la infraestructura del sistema en su totalidad y luego añadirle los componentes funcionales. De igual forma, el subsistema implementado está desarrollado con las mismas tecnologías y herramientas que la plataforma c.u.b.a., todas en las mismas versiones, para así garantizar de forma seguro su integración.

3.4.3. Pruebas de carga y estrés

Las pruebas de carga consisten en probar el funcionamiento del *software* bajo condiciones extremas. Estudia la especificación del *software*, las funciones que debe realizar, las entradas y las salidas, analizando las frecuencias o volúmenes extremos. Mientras que las pruebas de estrés están diseñadas para enfrentar al programa a condiciones anormales. Las pruebas ejecutan un sistema, de manera que demande recursos en cantidad, pues resulta necesario comprobar el rendimiento del sistema soportando una cantidad máxima de usuarios que soliciten este recurso en la *web*.

Para la realización de las pruebas de carga y estrés se utilizó la herramienta Apache Jmeter. A continuación, se describen las variables que miden el resultado para estas pruebas:

Muestra: Cantidad de peticiones realizadas para cada *URL*.

Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.

Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

Min.: Tiempo mínimo que demora un hilo en acceder a una página.

Max.: Tiempo máximo que demora un hilo en acceder a una página.

Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.

% Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.

Rendimiento (Rend): El rendimiento se mide en cantidad de solicitudes por segundo.

KB/s: El rendimiento se mide en cantidad de kilobytes10 por segundo.

Como se muestra en la Tabla 8 (Ver tabla 8.), se simularon las peticiones realizadas al sistema por un total de 200, 500 y 1000 usuarios simultáneamente, obteniéndose los siguientes resultados por cada variable que mide los resultados de estas pruebas:

Tabla 8.Resultados Obtenidos a partir de las pruebas de carga y estrés (Elaboración propia).

Usuarios	Muestra	Media	Mediana	Min	Max	Línea 90 %	%Error
200	8700	1358	1368	1	2778	1745	0
500	42020	4578	2014	4	456123	3127	0.56
1000	75298	9567	2267	3	505489	5146	1.68

Las pruebas realizadas muestran que el subsistema de archivos musicales es capaz de responder:

- Para un total de 8700 peticiones iniciadas por 200 usuarios conectados el subsistema respondió en un tiempo promedio de 1358 milisegundos (1.6 segundos aproximadamente) con 0 % de error. Esto evidencia que el sistema puede procesar la carga esperada, cumpliéndose el **RnF10.**
- Para un total de 42020 peticiones iniciadas por 500 conectados el subsistema respondió en un tiempo promedio de 4578 milisegundos (4.4 segundos aproximadamente). Para ese total de peticiones realizadas, el subsistema no fue capaz de responder correctamente a un 0.56%, pero el subsistema responde en el tiempo esperado a un conjunto de peticiones 8.4 veces mayor que el propuesto en el **RnF10.**

A continuación, se realiza la prueba de estrés con el objetivo de analizar el comportamiento del subsistema en condiciones extremas:

- Para un total de 75298 peticiones iniciadas por 1000 conectados el subsistema solo responde a una parte de este valor en un tiempo promedio de 9567 milisegundos (9.6 segundos aproximadamente), que en este caso sería el doble del tiempo propuesto el en **RnF10.**

De manera general estos resultados son favorables teniendo en cuenta que en gran parte de las pruebas se cumple el **RnF10.**

3.4.4. Pruebas de seguridad

Las pruebas de seguridad comprenden la puesta en práctica de un conjunto de medidas preventivas y reactivas en los sistemas informáticos y tecnológicos que posibilitan la protección de la información, persiguiendo como objetivo principal la integridad, confidencialidad y disponibilidad de la misma (Pressman, 2010). Al subsistema desarrollado se le realizaron una serie de pruebas de seguridad mediante la herramienta *Acunetix Web Vulnerability Scanner*, las cuales se presentan a continuación:

- Ataques de inyección.
- *Cross-Site* Scripting (XSS).
- Falsificación de petición (CSRF).
- Detección de ficheros y directorios.

Mediante estas pruebas se encontraron en una primera iteración 15 vulnerabilidades en total: 5 poseen un nivel de criticidad medio, 7 son de nivel de criticidad bajo y 3 son informacionales. No se detectaron ninguna vulnerabilidad de nivel de criticidad alto. De manera general los resultados obtenidos se presentan en la tabla 12 y 13 (Ver tabla 12 y tabla 13).

Tabla 9. Vulnerabilidades del entorno (Elaboración propia).

Tipo	Cantidad	Descripción	Recomendaciones
Software	4	El *craw* de PHP 8.0 que se encuentra disponible para esta versión en *Software* Libre, presenta problemas a la hora de terminar el proceso de instalación del lenguaje.	Cambiar el craw actual disponible para PHP 8.0 por uno que no presente problemas.
Configuración	7	La configuración de Apache Nutch versión 1.9 presentó algunos problemas en cuanto a los plugins de configuración de Nutch y al tipo de servicio que se requiere.	Se recomienda trabajar con Apache Nutch versión 1.12 o superior, ya que resuelve todos los inconvenientes de configuración de *plugins* de configuración de Nutch.

Tabla 10.Vunerabilidades del sistema (Elaboración propia).

Tipo	Cantidad	Descripción	Recomendaciones
Falso positivo	4	El sistema permite inyección de código.	Esta vulnerabilidad no puede ser utilizada por el atacante para alterar el sistema ni obtener información por lo que se denomina falso positivo.

Todas las vulnerabilidades fueron resueltas en el servidor para una segunda iteración, llegando a la conclusión que la herramienta desarrollada es segura y está en condiciones de ser usada por el cliente con el propósito de realizar búsqueda de archivos musicales en la plataforma c.u.b.a. Se recomienda que se tenga en cuenta la nueva configuración en el servidor para un futuro despliegue del sistema.

3.5. Validación de la hipótesis de la investigación

Una vez terminado el subsistema de búsqueda de archivos musicales para la plataforma c.u.b.a. se hace necesario la validación de la hipótesis de la investigación, con el fin de comprobar si se mejora el proceso de recuperación de archivos musicales. La validación se realiza a partir de los indicadores definidos en la introducción referentes a la **precisión** y la **exhaustividad** en el subsistema de búsqueda de archivos musicales, correspondiente a la **variable dependiente** definida como parte de la hipótesis de investigación. Ambos indicadores son ampliamente aceptados en la comunidad de RI (Cleverdon & Tolosa, 2007), y se definen:

Precisión: El concepto de precisión propuesto por el estudioso en RI Gerald Salton en 1983, explica que la precisión es la proporción de material recuperado realmente relevante, del total de los documentos recuperados. A esta definición se le añade que el resultado de esta operación está entre 0 y 1. Así la recuperación perfecta es en la que únicamente se recuperan los documentos relevantes y por lo tanto tiene un valor igual a 1 (Cleverdon & Tolosa, 2007).

Exhaustividad: Salton se refiere a la exhaustividad como concepto bastante utilizado en la evaluación de los sistemas de recuperación. Es la proporción de material relevante recuperado, del total de los documentos que son relevantes en la base de datos, independientemente de que éstos, se recuperen o no. Esta medida es inversamente proporcional a la precisión (Cleverdon & Tolosa, 2007).

En la siguiente figura se muestra las fórmulas propuesta por Salton para medir los indicadores de precisión y exhaustividad:

Tabla 11. Fórmulas para evaluar métricas de calidad en los SRI (Cleverdon & Tolosa, 2007),

Nombre	Fórmulas
Fórmula para medir exhaustividad.	$Exhaustividad = \frac{Doc.\ relev.\ recup.}{Doc.\ relev.}$

Fórmula para medir precisión.	$Precisión = \frac{Doc.\ relev.\ recup.}{Doc.\ recup.}$

Mientras más se acerque el valor de la precisión al valor 0, mayor será el número de documentos recuperados que no sea relevantes. Si, por el contrario, el valor de la precisión es igual a 1, se entenderá que todos los documentos recuperados son relevantes. Por otra parte, si el resultado del valor de la exhaustividad es igual al valor 1, tendremos la exhaustividad máxima, ya que hemos encontrado todo lo relevante que había en la base de datos.

Luego de la explicación de los indicadores a utilizar como parte del proceso de validación de la hipótesis se realiza el preexperimento **antes** y **después** del desarrollo de subsistema para comprobar la efectividad de la búsqueda. A continuación, se muestra en las tablas 1 (Ver tabla 13 y tabla 14) los resultados obtenidos antes y después de la implementación del subsistema, dados diferentes criterios (nombre, formato, artista) de búsqueda de archivos musicales.

Tabla 12. Resultados de Precisión y exhaustividad antes del desarrollo de subsistema (Elaboración propia).

Indicadores	Buscadores/Criterios	Criterio 1	Criterio 2	Criterio 3
Precisión	Plataforma c.u.b.a.	0,16	0,23	0,02
Exhaustividad	Plataforma c.u.b.a.	0,1	0,06	N/A

Tabla 13. Resultados Precisión y exhaustividad después del desarrollo de subsistema (Elaboración propia).

Indicadores	Buscadores/Criterios	Criterio 1	Criterio 2	Criterio 3
Precisión	Subsistema de búsqueda música	1	0,7	0,7
Exhaustividad	Subsistema de búsqueda música.	0,8	1	0,9

A continuación, se muestra la comparación del indicador Precisión y el indicador Exhaustividad antes de la implementación del subsistema (Ver tabla 16. Y tabla 17.).

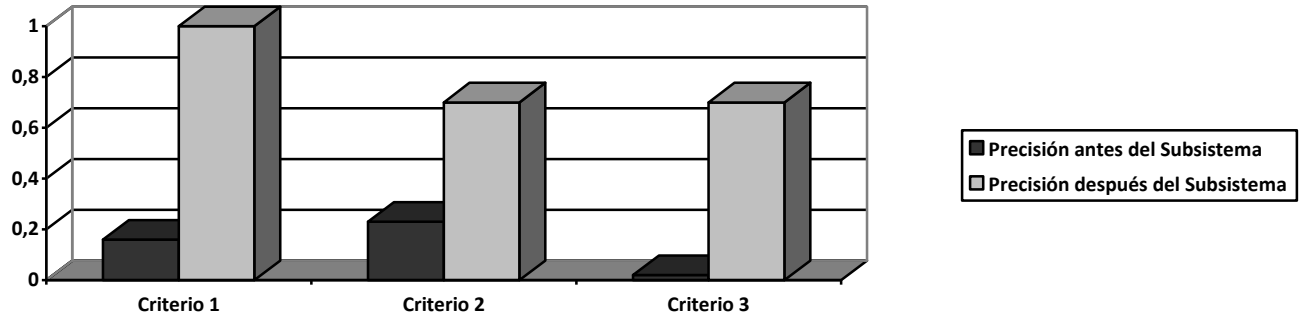

Figura 17. Comparación del indicador precisión antes y después del desarrollo del Subsistema (Elaboración propia)

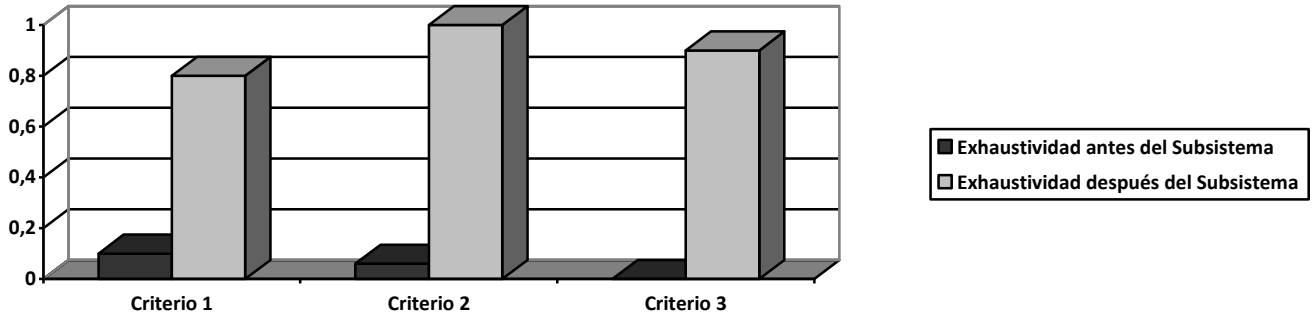

Figura 18. Comparación del indicador exhaustividad antes y después del desarrollo del Subsistema (Elaboración propia)

Teniendo en cuenta los valores de precisión y exhaustividad antes y después de la implementación del subsistema de búsqueda de archivos musicales, se evidencia que una vez implementado el subsistema los valores se acercan más a uno, por tanto, de demuestra que se mejoró la recuperación de información de archivos musicales en cuanto a precisión y la exhaustividad.

Conclusiones del capítulo

En el presente capítulo se abordaron una serie de aspectos correspondientes a la implementación y validación del subsistema propuesto, donde se llegó a las siguientes conclusiones:

- La representación y descripción del diagrama de componentes permitió visualizar con más facilidad la estructura general de la herramienta.
- El correcto uso de los estándares de codificación permitió que el código del sistema desarrollado fuera legible para lograr una fácil y mejor comprensión del mismo.
- La ejecución de pruebas al subsistema permitió detectar las deficiencias presentes, subsanarlas en el menor tiempo posible y ofrecer una aplicación con mayor calidad, seguridad y usabilidad.

CONCLUSIONES

En la presente investigación se ha llevado a cabo un proceso de desarrollo de *software* completo, dividido en flujos de trabajos e iteraciones, con el objetivo de lograr un producto de calidad en el tiempo establecido. Una vez completada, se puede concluir que:

- A partir del estudio realizado de los conceptos asociados a los referentes teóricos de la investigación se logró una mayor comprensión de la propuesta de solución y se determinó la necesidad de crear un subsistema para la búsqueda de archivos musicales bajo el dominio. cu.
- El enfoque ágil propuesto por la metodología AUP-UCI y el uso de tecnologías y herramientas seleccionadas, permitió analizar y describir los subprocesos que se debían de ejecutar, concretando así, en concordancia con las especificaciones del cliente, las características que debería de tener el subsistema a desarrollarse.
- La evaluación de las pruebas de *software* realizadas, permitieron erradicar las insuficiencias detectadas en el subsistema desarrollado, que comprometían la comunicación entre los diferentes componentes que integran el sistema y la facilidad de uso de las funcionalidades presentes.
- La validación de la hipótesis de investigación a través del preexperimento realizado permitió demostrar el mejoramiento de la calidad del proceso de recuperación de archivos musicales en la plataforma c.u.b.a.

RECOMENDACIONES

1. Extender la recopilación de los metadatos como el género y la duración del archivo musical e incluirlos en la búsqueda avanzada de archivos musicales.
2. Implementar nuevas funcionalidades que permitan al subsistema realizar búsquedas semánticas de archivos musicales en la *web* nacional.
3. Desarrollar un sistema basado en ontologías para la recuperación de archivos audiovisuales en la *web* cubana para la plataforma c.u.b.a.

BIBLIOGRAFÍA

Acosta, J., M. *Medición de atributos POO en frameworks de desarrollo PHP*. Buenos Aires, Argentina: s.n., 2014. ISSN 1695-549.

Acunetix. Audit your website security with Acunetix Web Vulnerability Scanner. [En línea]. Acunetix, 2015. [Citado el: 2 de mayo de 2018.]. Disponible en: http://www.acunetix.com/.

Agencia Informativa Latinoamericana Prensa Latina. [En línea]. Cubarte. noviembre de 2017. [Citado el: 12 de diciembre de 2017.] Disponible en: http://www.cubarte.cult.cu/es/article/50114.

Aksynoff, A. Sphinx 2.3.2-dev reference manual. [En línea]. Sphinx | Open Source Search Server, 2014. [Citado el: 13 de octubre de 2017.] Disponible en: http://sphinxsearch.com/docs/current.html.

Aleman, J. Y., Rodríguez Rueda , E., Estrada Velazco , A., & Thomas Sosa, Y. J. Propuesta de Arquitectura de un Módulo de Configuaración Web, 2014 [Citado el: 12 de diciembre de 2017.] Disponible en: https://www.researchgate.net/publication/264001924.

Apache JMeter. [En línea]. Apache JMeter, 2017. [Citado el: 9 de diciembre de 2017.] Disponible en: http://jmeter.apache.org/i.

Apache JMeter. [En línea]. Apache JMeter, 2015. [Citado el: 20 de octubre de 2017.] Disponible en: http://jmeter.apache.org.

Apache Software Fundation the Apache Solr Reference Guide. [En línea]. Apache Solr Resources, 2014. [Citado el: 22 de octubre de 2017.]. Disponible en: http://lucene.apache.org/solr/documentation.html.

ApacheLucene. [En línea]. Apache Solr, 2017. [Citado el: 20 de octubre de 2017.]. Disponible en: http://lucene.apache.org/solr/index.html.

Apple Music. [En línea]. Apple Music, 2017. [Citado el: 24 de octubre de 2017.]. Disponible en: https://www.apple.com/la/apple-music/

Aula Digital TelMex. [En línea]. Biblioteca Digital TelMex, 2012. [Citado el: 22 de noviembre de 2017.]. Disponible en: http://www.telmexeducacion.com/proyectos/DocsDobleclic/14-Doble%20clic-Buscadores%20o%20motores%20de%20busqueda.pdf.

Ayala Pichardo, J. A. [En línea]. Modelo de Recuperación Booleano, 2007. [Citado el: 22 de noviembre de 2017.]. Disponible en: http://recuperacioninf.orgfree.com/modelo_booleano.html.

Baeza Yates, R., & Ribeiro Nieto, B. Modern information retrieval. New York: ACM Press; Harlow [etc.]: Addison-Wesley, 1999. ISBN 0-201-39829-X.

Baeza Yates, R., & Ribeiro Nieto, B. Modelling: Boolean model. Modern Information Retrieval. New York: ACM Press, 2005 Disponible en: http://grupoweb.upf.es/WRG/mir2ed/pdf/slides_chap03.pdf.

Bahit, E. *POO y MVC en PHP. El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.* [En línea]. Slideshare, 2013. [Citado el: 20 de enero de 2018.] Disponible en: http://www.slideshare.net/eugeniabahit/poo-y-mvc-en-php-por-eugenia-bahit.

Bakken, S. S. Manual de PHP. s.l.: The PHP Documentation Group, 2013. pág. 1063. ISBN 0-201-39829-X.

Baptista, P. Metodología de la Investigación, 2016. Buenos Aires. Argentina [Citado el: 27 de septiembre de 2017.].

Barkov, A, mnoGoSearch 3.3.15 reference manual: Full-featured search engine software. [En línea] 2014. Disponible en: http://www.mnogosearch.org/doc33/msearchintro.html#features.

Bootstrap, **P**. Portal Oficial de Bootstrap, 2017 [Citado el: 27 de noviembre de 2017.] Disponible en: http://conocimientoabierto.es/

Borlum, P. and Ingwersten, P. The development of method for evaluation of interactive Information Retrieval System, 1997 Journal of Documentation 1997 53 (3) p. 225-250. ISBN 0-201-39829-X.

Cabrera Gonzáles, L. Extensión de Visual Paradigm for UML para el Desarrollo Dirigido por Modelos de aplicaciones de gestión de información, 2013 [Citado el: 23 de octubre de 2017.] Disponible en: hhttp://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05815_12i.

Caldera J. Análisis de la comercialización de los archivos audiovisuales televisivos por la red: posibilidades e implicaciones, 2015. Madrid, España. ISSN 1695-549.

Caldera, J Indexación automática de archivos audiovisuales, 2016 [Citado el: 23 de octubre de 2017.] Disponible en: hacialaIndizacionautomaticaaedocumentosaudiovisual-4800985.pdf.

Centro Cubano de Información de Red CUBANIC. CUBANIC, 2015 [Citado el: 25 de octubre de 2017.] Disponible en: http://www.nic.cu/estadisticas.php.

Eguiluz, J. Desarrollo web ágil con Symfony2, 2013, 2015 [Citado el: 25 de octubre de 2017.] pág. 618 .ISSN 1695-549.

Elizalde, R. La política de Informatización de Cuba partirá de una visión inclusiva, moderna y sostenible, 2015 [Citado el: 30 de octubre de 2017.] Disponible en: http://www.granma.cu/cuba/2015-02-13/ailynfebles-la-politica-de-informatizacion-de-cuba-partira-de-una-vision-inclusiva-modernay-sostenible.

Franco, J. Funcionamiento de un buscador, 2015. [Citado el: 30 de octubre de 2017.] Disponible en: http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node207.html.

Framework Process Eclipse [En línea] 2014. [Citado el: 24 de octubre de 2017.] Disponible en: http://epf.eclipse.org/wikis/openupsp/openup_basic/customcategories/introduction_to_openup_basic,_BTJ_YMXwEduywMSzPT].

Galdón, A. Motores de búsqueda para contenidos audiovisuales, 2016. Madrid, España ISSN 1695-549.

Gamma, E. P. Patrones GoF, 2016 [Citado el: 23 de marzo de 2018.] Disponible en: http://geektheplanet.net/5462/patrones-gof.xhtml.

Giordanino, E. E-prints in library & informatic science, 2015 [Citado el: 23 de marzo de 2018.]. Disponible en: http://eprints.rclis.org/14874/1/TAI2_introSRI_v4.pdf.

González, S. *Conocimiento científico e información científica,* 2018 (C. d. Cuba., Editor) [Citado el: 20 de abril de 2018.]. Disponible en: http://bvs.sld.cu/revistas/aci/vol14_6_06/aci03606.htm.

Gómez Díaz, Raquel. La evaluación en recuperación de la información [online]. "Hipertext.net", núm. 1, 2003.) [Citado el: 2 de abril de 2018.]. Disponible en: http://www.hipertext.net.

Gutiérrez, C. Cómo funciona La Web, 2013. Santiago, Chile, Universidad de Chile. ISSN 1695-549.

Guzmán Gonzáles, G. Componente para la indexación y búsqueda contextual de información audiovisual, 2011. [Citado el: 2 de abril del 2018.] págs.102. Disponible en: hhttp://repositorio_institucional.uci.cu/jspui/handle/ident/TD_04322_11i.

Henao, C. Ejemplo Patrón de Diseño, 2018. [Citado el: 2 de diciembre de 2017.] Disponible en: http://codejavu.blogspot.com/2013/07/ejemplo-patron-de-diseno-decorator.html.

Hernández, M. C. Manual de Usuario del Motor de Búsqueda Cubano, 2013. La Habana, Cuba, Universidad de las Ciencias Informáticas ISSN 1695-549.

Hernández Sampier, R. Metodología de la investigación, 2008.ç ISBN: 0321193121.

Ibarra, M. Ventajas y desventajas de motores de búsqueda, 2014 (© 2018 Prezi Inc. Condiciones). [Citado el: 2 de diciembre de 2017.] Disponible en: https://prezi.com/j7c88gnhpv_e/ventajas-y-desventajas-de-motores-de-busqueda/.

IEEE. Guide to the Software Engineering Body of Knowledge. 2013. ISBN 0-7695-2330-7.

Jong, A. Los metadatos en el entorno de la producción audiovisual, 2010. Ámsterdam, Holanda. ISBN: 0321193121.

JQuery. What is jQuery? The jQuery Foundation, 2017. [Citado el: 24 de noviembre de 2017.]. Disponible en: https://jquery.com/.

Kiva. Buscadores o Search Engine, 2015. [Citado el: 2 de febrero de 2018.]. Disponible en: http://www.searchoptimization.es/buscadores-search-engines/buscadores-search-engines.htm.

Kujawski, M. Navigating Digital Disruption, 2018. [Citado el: 2 de abril de 2018.]. Disponible en: http://www.mikekujawski.ca/wp-content/uploads/2017/02/We-Are-Social-Digital-Yearkbook-2018.pdf.

Larman, C. UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado, Segunda edición. s.l. : Prentice Hall, 2004. pág. 520. ISSN: 0975-3397.

Larman, C. Introducción al análisis y diseño orientado a objetos y al proceso unificado. Primera edición. s.l.: Prentice Hall,1999. ISSN: 0975-3397.

Larman, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado. s.l.: Prentice Hall, 2003. ISSN: 0975-3397.

Ledesma Rodríguez Y. Extensión de la herramienta Visual Paradigm for UML para el soporte al Desarrollo Dirigido por Modelos con Ext JS, 2011. [Citado el: 25 de octubre de 2017.]. 74 págs. Disponible en: hhttp://repositorio_institucional.uci.cu//jspui/handle/ident/TD_04358_11i.

Marca Huallpara, H. Análisis y diseño de Sistemas II: Diagrama de Despliegue, 2017. La Habana, Cuba.

Macmillan Publishers Limited. *MACMILLAN DICTIONARY*, 2017 [Citado el: 13 de octubre de 2017.]. Disponible en: https://www.macmillandictionary.com/.

MusicAll. Muscicall, 2017. [Citado el: 12 de noviembre del 2017.]. Disponible en: https://musicall.uptodown.com/android.

Nielsen, J. Usability Engineering. San Francisco, 2010. Editorial Morgan Kaufman. ISSN: 0975-3397.

Nioche, J. Large Scale Crawling with Apache Nutch and Friends. Proceedings of Lucene/Solr Revolution. Lucene/Solr Revolution, 2013. ISSN: 0975-3397.

Oracle. NetBeans IDE Features, 2014. [Citado el: 13 de octubre de 2017.]. Disponible en: https://netbeans.org/features/index.htm

PCC. Lineamientos, 2011. Habana, Cuba: Editora Política. [Citado el: 13 de octubre de 2017.].

PHP. ¿Qué es PHP? PHP Group, 2017. [Citado el: 15 de abril de 2017.] Disponible en: http://php.net/manual/es/intro-whatis.php.

PHP. Framework Interop Group (PSR-1), 2014. Codificación estándar básica. [Citado el: 18 de abril de 2018.] Disponible en: http://www.php-fig.org/psr/psr-1/es.

PHP. Framework Interop Group (PSR-2), 2014. Codificación estándar básica. [Citado el: 18 de abril de 2018.] Disponible en: http://www.php-fig.org/psr/psr-2/es.

PHP. Framework Interop Group (PSR-4), 2014. Codificación estándar básica. [Citado el: 18 de abril de 2018.] Disponible en: http://www.php-fig.org/psr/psr-4/es.

PowerData. *PowerData*. Recuperado, 2016, [Citado el: 18 de abril de 2018.] Disponible en: https://www.powerdata.es/migracion-de-datos.

Pressman, R. Ingeniería del software. Un enfoque práctico. 6ta Edición, 2006. ISBN 970-10-5473-3.71.

Pressman, R. Ingeniería del software. Un enfoque práctico. 5ta Edición, 2005. ISBN 970-10-5473-3.71.

Pressman, R. Ingeniería del software. Un enfoque práctico. 7ta Edición, 2005. ISBN 970-10-5473-3.71.

Rodríguez, E. Los spiders y su función en los motores de búsqueda. 2012 [Citado el: 18 de abril de 2018.] Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/4130.

Rodríguez, Y. Extensión de la herramienta Visual Paradigm for UML para el soporte al Desarrollo Dirigido por Modelos con Ext JS. Repositorio Institucional, 74., 2011 [Citado el: 18 de abril de 2018.] Disponible en: http://repositorio__institucional.uci.cu//jspui/handle/ident/4133.

Sánchez, T. R. Metodología de desarrollo para la actividad productiva de la UCI, 2015. Universidad de las Ciencias Informáticas, La Habana, Cuba.

Santana, A. Tablero de control para entidades orientadas a proyecto. Tesis de Maestría en Gestión de Proyectos, 2015. Universidad de las Ciencias Informáticas, La Habana, Cuba.

SensioLabsNetework. Coding Standards.SensioLabs Product, 2014. [Citado el: 18 de abril de 2018.] Disponible en: http://symfony.com/doc/current/contributing/code/standards.html.

Sommerville, I. Ingeniería del software. 7 Edición. Pearson Educación, 2005 712 págs. ISBN 84-7829-074-5.

Solr-Wiki. Public Websites using Solr. [En línea]. PublicServers - Solr Wiki, 2014. [Citado el: 4 de noviembre de 2017.] Disponible en: https://wiki.apache.org/solr/PublicServers.

Spotify. 2017 [Citado el: 18 de abril de 2018.] Disponible en: https://webcache.googleusercontent.com/search?q=cache:5L1svYy2b7sJ:https://www.spotify.com/mx/legal/+&cd=1.

Spotify app. The History of Sportiy, 2017. [Citado el: 4 de noviembre de 2017.] Disponible en: http://docs.spotify.com/info/misc/history.html.

Symfony. Symfony, 2015 [Citado el: 4 de noviembre de 2017.] Disponible en: http://symfony.es/noticias/2015/11/19/nuevo-en-symfony-28-usando-symfony-como-un-microframework/.

Slideshare. Comparing open source search engines. [En línea]. Slideshare, 2016 [Citado el: 11 de marzo de 2018.]. Disponible en: http://www.slideshare.net/rboulton/comparing-open-source-searchengines.

The Apache Software Foundation-Tika. Apache Tika. [En línea]. Apache Solr, 2014. [Citado el: 13 de octubre de 2017.] Disponible en: http://tika.apache.org/.

The Apache Software Foundation-Solr. Apache Solr. [En línea]. Apache Solr, 2014. [Citado el: 3 de octubre de 2017.] Disponible en: http://lucene.apache.org/solr/.

The Apache Software Foundation-JMETER. Apache JMeter. [En línea]. Apache JMeter, 2015. [Citado el: 24 de marzo de 2018.] Disponible en: http://jmeter.apache.org/.

The History of Apple Music. Apple Music, 2016 [Citado el: 13 de octubre de 2017.] Disponible en: http://docs.applemusic.com/info/misc/history.html.

Thornton, J. Bootstrap3 Manual Official, 2013 [Citado el: 13 de octubre de 2017.]

Tortajada Cordero, J. La guía definitiva de diseño web: HTML, XHTML, CSS y herramientas de diseño, 2017. Santander y alrededores, España: Universidad de Europea del Atlántico. [Citado el: 4 de octubre de 2017.]

Visual Paradigm. Visual Paradigm for UML - Software design tools for agile software development. [En línea] 2014. [Citado el: 24 de octubre de 2017.] Disponible en: http://www.visualparadigm.com/product/vpuml/.

Visual Paradigm. Visual Paradigm for UML, 2017. [Citado el: 3 de octubre de 2017.] Disponible en: http://www.visual-paradigm.com/product/vpuml/.

WC3. Web World Wide, 2014 [Citado el: 3 de octubre de 2017.] Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

Web Application Security with Acunetix Web Vulnerability Scanner. Acunetix Web, 2017 [Citado el: 3 de octubre de 2017.] Disponible en: de https://www.acunetix.com.

Wire. Web Information Retrieval Environment, 2013 [Citado el: 3 de octubre de 2017.] Disponible en: hhttp://www.cwr.cl/projects/WIRE/.

Yahoo! Inc. The History of Yahoo! - How It All Started, 2017. [Citado el: 5 de octubre de 2017.] Disponible en: http://docs.yahoo.com/info/misc/history.html.

ANEXOS

Anexo 1 Estadísticas asociadas a los usuarios

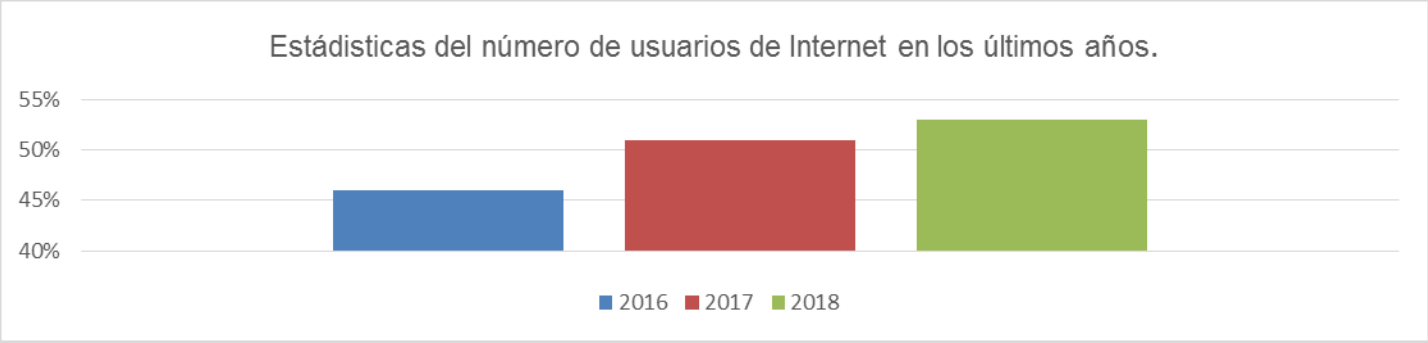

Figura 19. Comparación de la cantidad de usuarios online en los últimos años.

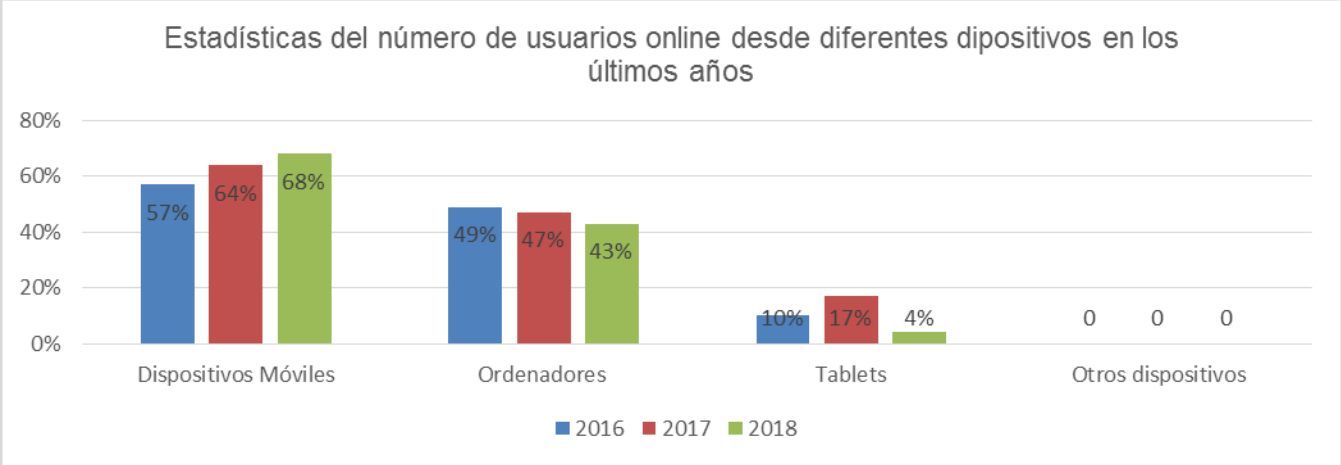

Figura 20. Comparación de la cantidad de usuarios online desde diferentes dispositivos.

Anexo 2 Plataforma c.u.b.a

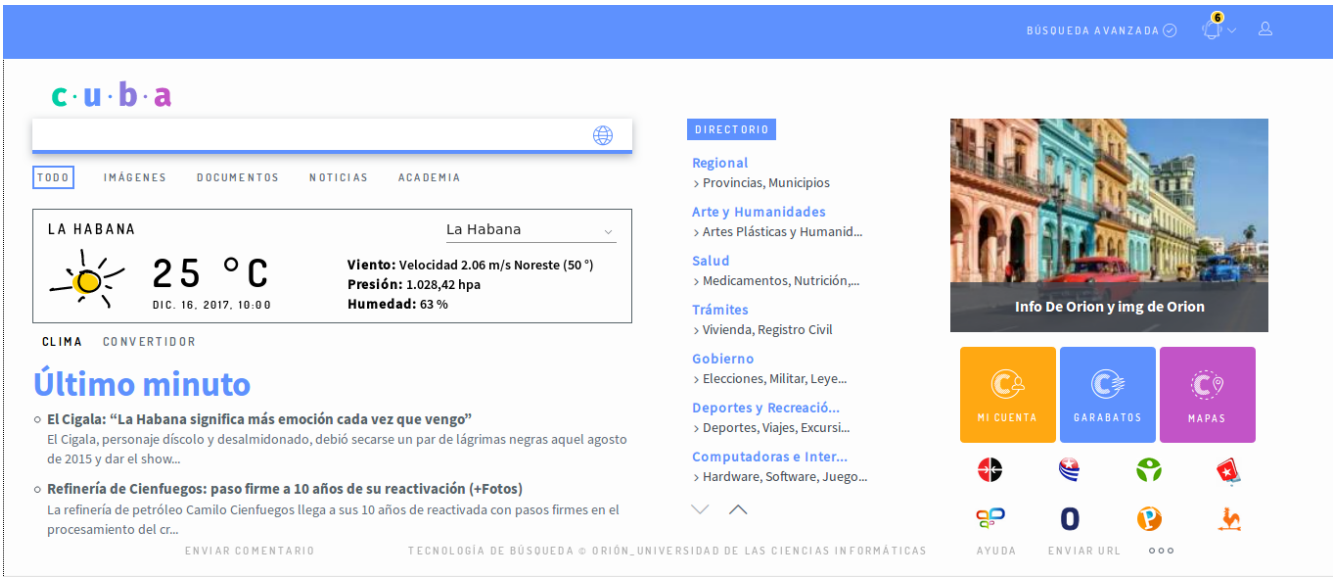

Figura 21. Interfaz de la plataforma c.u.b.a.

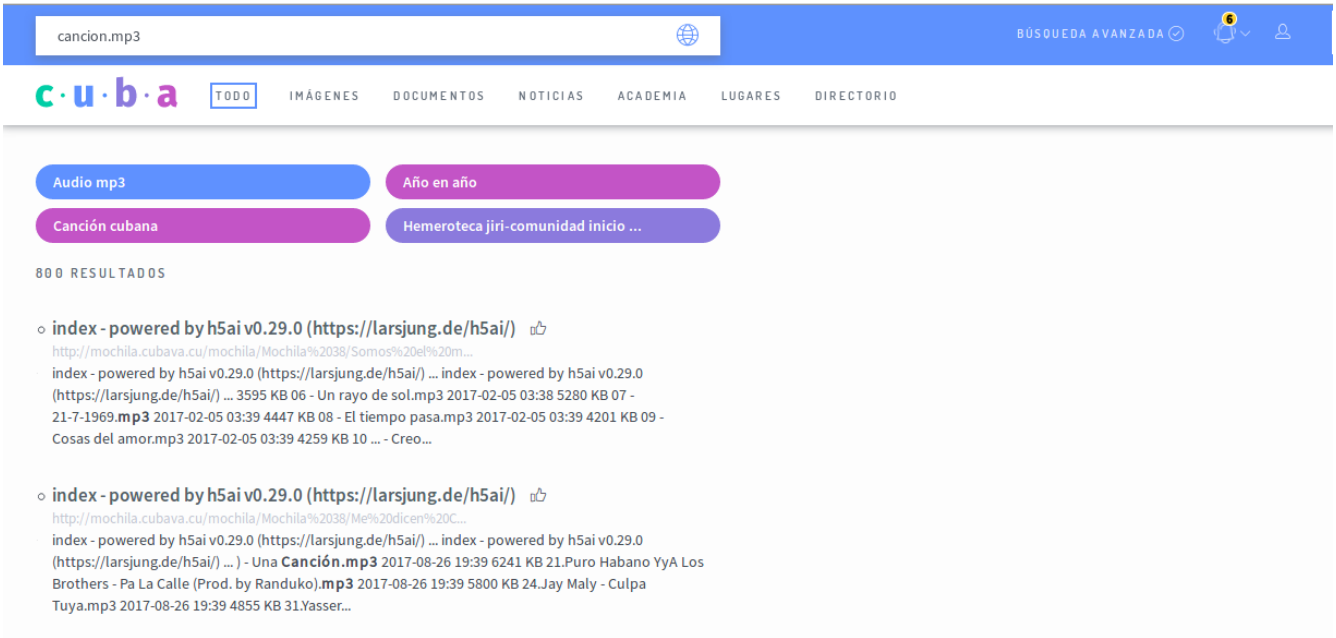

Figura 22. Pruebas de autor de la búsqueda de música en la plataforma c.u.b.a.

Anexo 3 Comparación entre las clasificaciones de los SRI.

Tabla 14. Comparación entre las clasificaciones de los SRI.

Criterio	Directorios	Motores de búsqueda	Metabuscadores
Base de datos	Bases de datos más pequeñas, menos actualizadas, y más elaboradas gracias a la presencia del factor humano.	Bases de datos más amplias y actualizadas.	No tienen bases de datos propias, sino que reenvían las consultas interrogando a varios buscadores a la vez.
Índices	Almacenan la información por temas y categorías, una vez recopilada, de forma manual en sus índices.	Colocan la información, que sean capaces de recoger en la red, en sus índices sin ordenarlas por temas, de manera automática y periódica.	No almacenan información porque no dependen de bases de datos propias.
Forma de búsqueda	No realizan las búsquedas en Internet de manera directa, sino que almacenan una breve descripción de las páginas y ofrecen enlace a éstas.	No realizan las búsquedas en Internet de manera directa, sino en las copias de las páginas que almacenan en sus índices.	Envían las consultas a varios motores de búsqueda, sus resultados dependen de que estos estén disponibles en el momento de la búsqueda, o se descarguen en el período de tiempo permisible.
Facilidad de uso	Son fáciles de usar, permiten, en primer lugar, ubicar la búsqueda en un tema determinado.	Son más difíciles de usar, se requiere explotar al máximo las opciones de búsqueda porque contienen más información.	Son difíciles de usar para búsquedas muy precisas, porque tienen menos control de la búsqueda al interrogar varias bases de datos con interfaces diferentes.
Utilidad	Son convenientes para buscar información general, institucional porque devuelven resultados a las páginas principales.	Se utilizan para buscar información más escasa, especializada, actualizada o incluida en páginas personales.	Se recomienda para temas difíciles de encontrar.

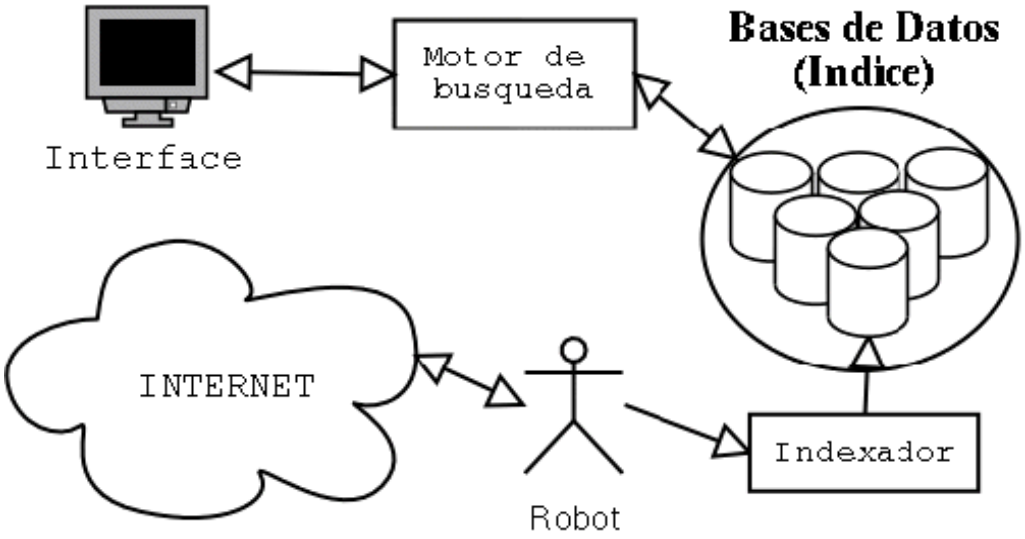

Figura 23. Diseño de un MB.

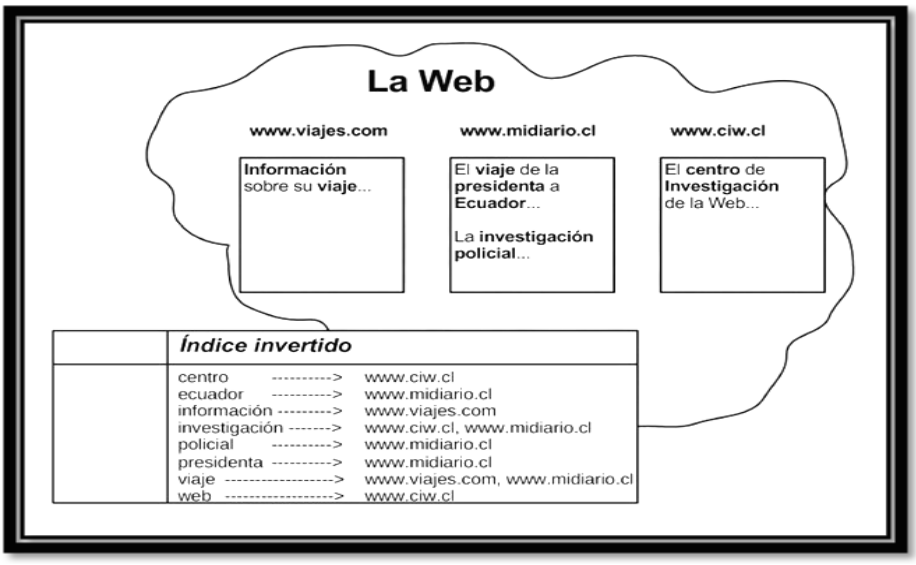

Figura 24. Proceso de Recolección de contenido web.

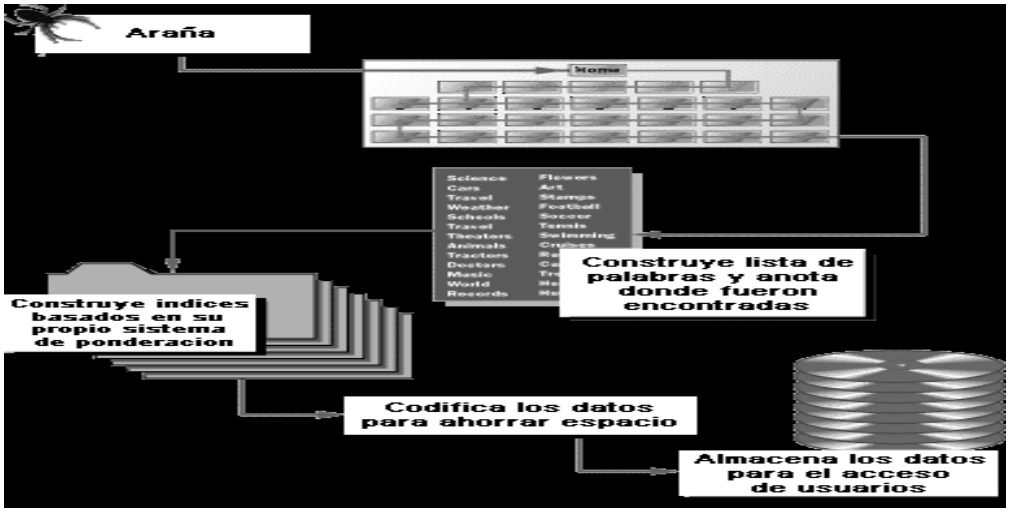

Figura 25. Procesos de un Rastreador.

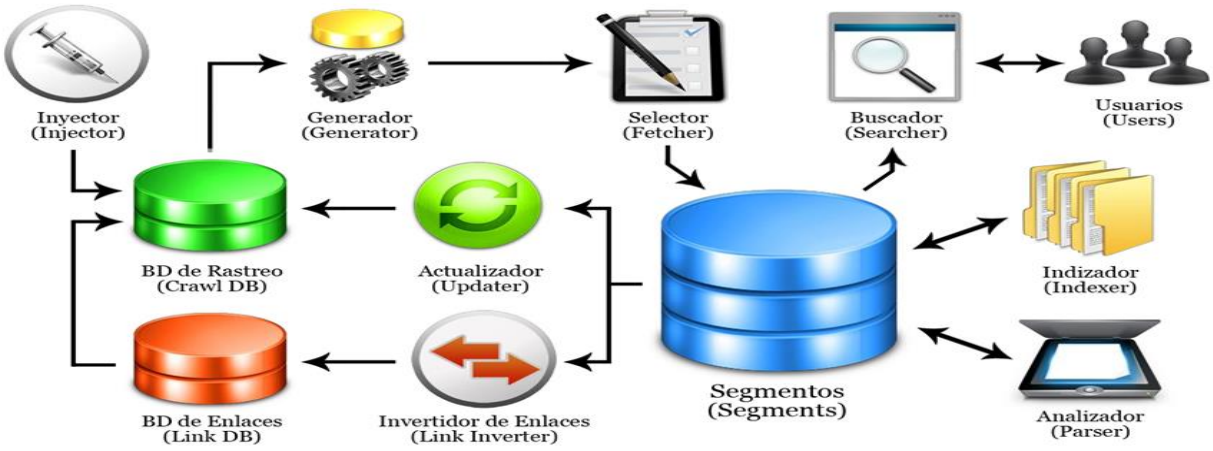

Figura 26. Fases del Rastreo de Nutch.

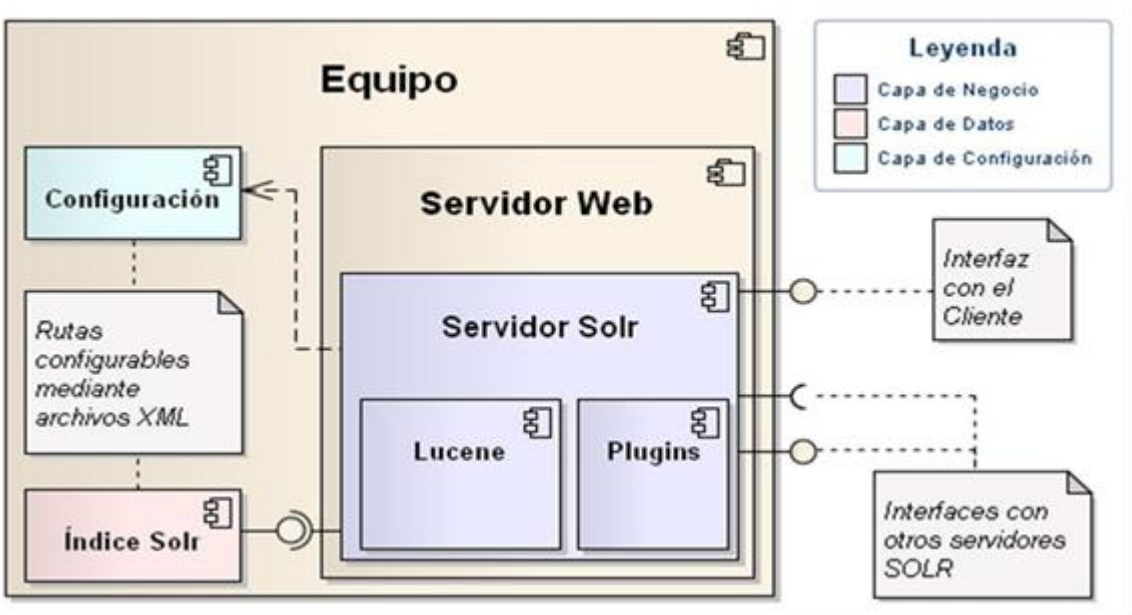

Figura 27. Elementos de Apache Solr.

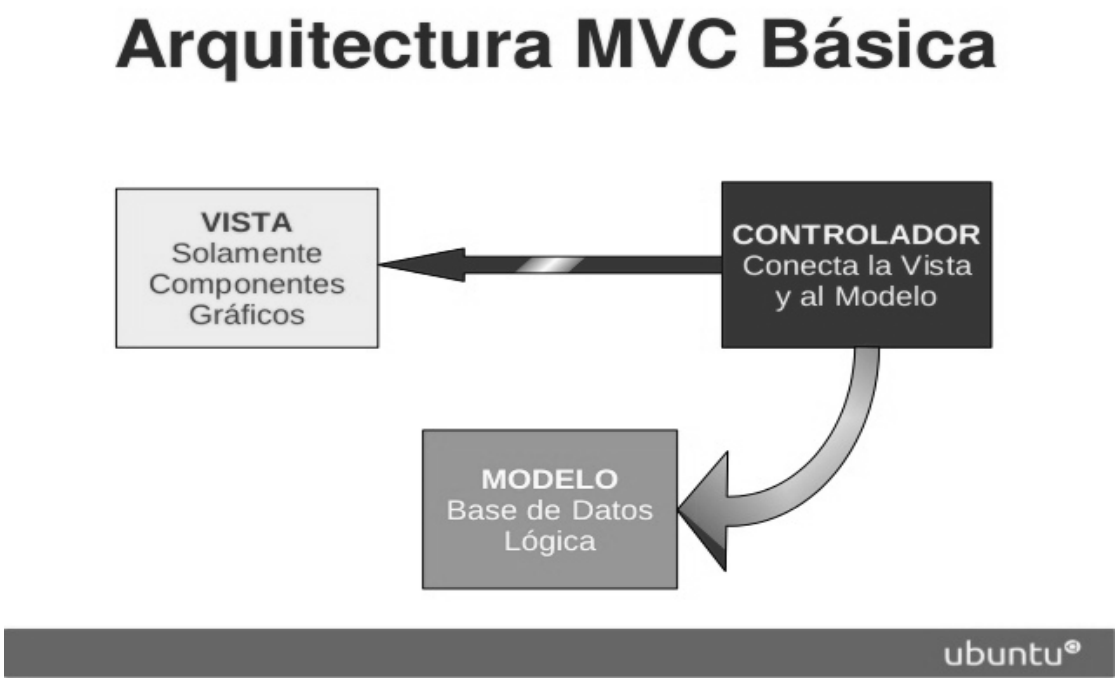

Figura 28. Arquitectura MVC.

Tabla 15. Descripción del CU Filtrar Contenido

Objetivo		**Extraer la dirección donde se encuentra el archivo musical (URL) dado un criterio.**
Actores		Administrador (inicia), Rastreador.
Resumen		El sistema indexa las URL previamente identificada de forma rápida, dado un criterio de donde se encuentra la música en la red.
Complejidad		Alta
Prioridad		Alta
Precondiciones		El administrador ha configurado el sistema y ha iniciado el rastreo.
Postcondiciones		Se han extraído las URL de donde está la música y se ha indexado la dirección en Solr.
Flujo de eventos		
Flujo básico Procesar Música		
Actor		**Sistema**
1	Se introduce un criterio.	
2		Inicia el rastreo
3		Realiza Búsqueda de sitios que contengan archivos.mp3 u otro formato
4		Encuentra los sitios donde se encuentra el archivo.mp3 u otro formato
5		Guarda las direcciones de los sitios encontrados en Nutch
6		Termina el CU
Requisitos no funcionales		

Tabla 16. Descripción del CU Transformar el Criterio de Búsqueda.

Objetivo	**Trasformar la consulta del lenguaje natural a un lenguaje documental o entendible para el sistema.**
Actores	Rastreador
Resumen	El sistema trasforma la consulta introducida por el usuario de un lenguaje natural a un lenguaje documental o entendible para el sistema.
Complejidad	Alta
Prioridad	Alta

Precondiciones		El sistema ha transformado la consulta y ha iniciado el rastreo.
Postcondiciones		Los archivos musicales fueron identificados.
Flujo de eventos		
Flujo básico Procesar Música		
Actor		**Sistema**
1	Transforma la consulta.	
2		Realiza un recorrido en profundidad X por el las páginas identificadas en el semillero hasta que alcance su límite máximo de búsqueda de música.
3		Si se alcanza el límite máximo de búsqueda de música por página se vuelve a recorrer el árbol con una nueva profundidad.
4		Analiza cada archivo musical identificado.
5		Devuelve los archivos musicales encontrados.
6		Termina el CU.
Requisitos no funcionales		

Tabla 17. Descripción del CU Identificar Música en página web.

Objetivo	**El rastreador identifica la URL en donde se encuentra el archivo musical**
Actores	Rastreador
Resumen	El sistema identifica los enlaces donde se encuentra el archivo musical verificando que sea .mp3 u otra etiqueta establecida por el administrador. .
Complejidad	Alta
Prioridad	Alta
Precondiciones	El sistema encuentra las URL y la indexa automáticamente.
Postcondiciones	Las URL fueron identificadas.
Flujo de eventos	

Flujo básico Procesar Música		
Actor		**Sistema**
1	Identifica URL.	
2		Realiza un recorrido en profundidad X por el las páginas identificadas en el semillero hasta que alcance su límite máximo de búsqueda de música.
3		Si se alcanza el límite máximo de búsqueda de música por página se vuelve a recorrer el árbol con una nueva profundidad.
4		Analiza cada archivo musical identificado.
5		Devuelve los archivos musicales encontrados.
6		Verifica si los archivos son.mp3 u otras etiquetas establecidas por el administrador.
7		Termina el CU.
Requisitos no funcionales		

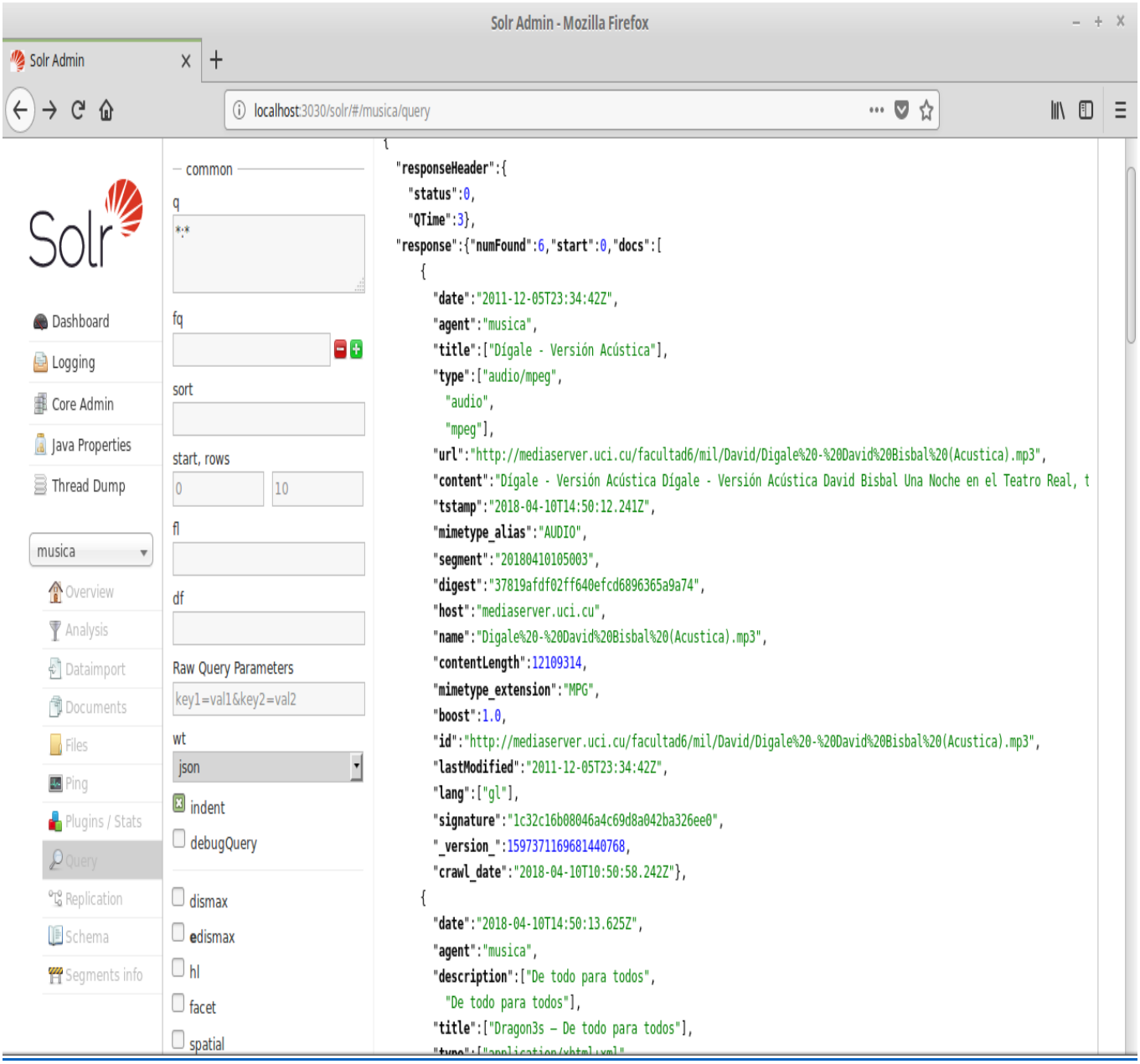

Figura 29. Ejemplo de Indexación de Música en Solr.

```
Mp3File mp3file = new Mp3File("SomeMp3File.mp3");
if (mp3file.hasId3v1Tag()) {
  ID3v1 id3v1Tag = mp3file.getId3v1Tag();
  System.out.println("Track: " + id3v1Tag.getTrack());
  System.out.println("Artist: " + id3v1Tag.getArtist());
  System.out.println("Title: " + id3v1Tag.getTitle());
  System.out.println("Album: " + id3v1Tag.getAlbum());
  System.out.println("Year: " + id3v1Tag.getYear());
  System.out.println("Genre: " + id3v1Tag.getGenre() + " (" + id3v1Tag.getGenreDescription() + ")");
  System.out.println("Comment: " + id3v1Tag.getComment());
}
```

Figura 30. Metadatos Identificados en la librería JavaMusic.

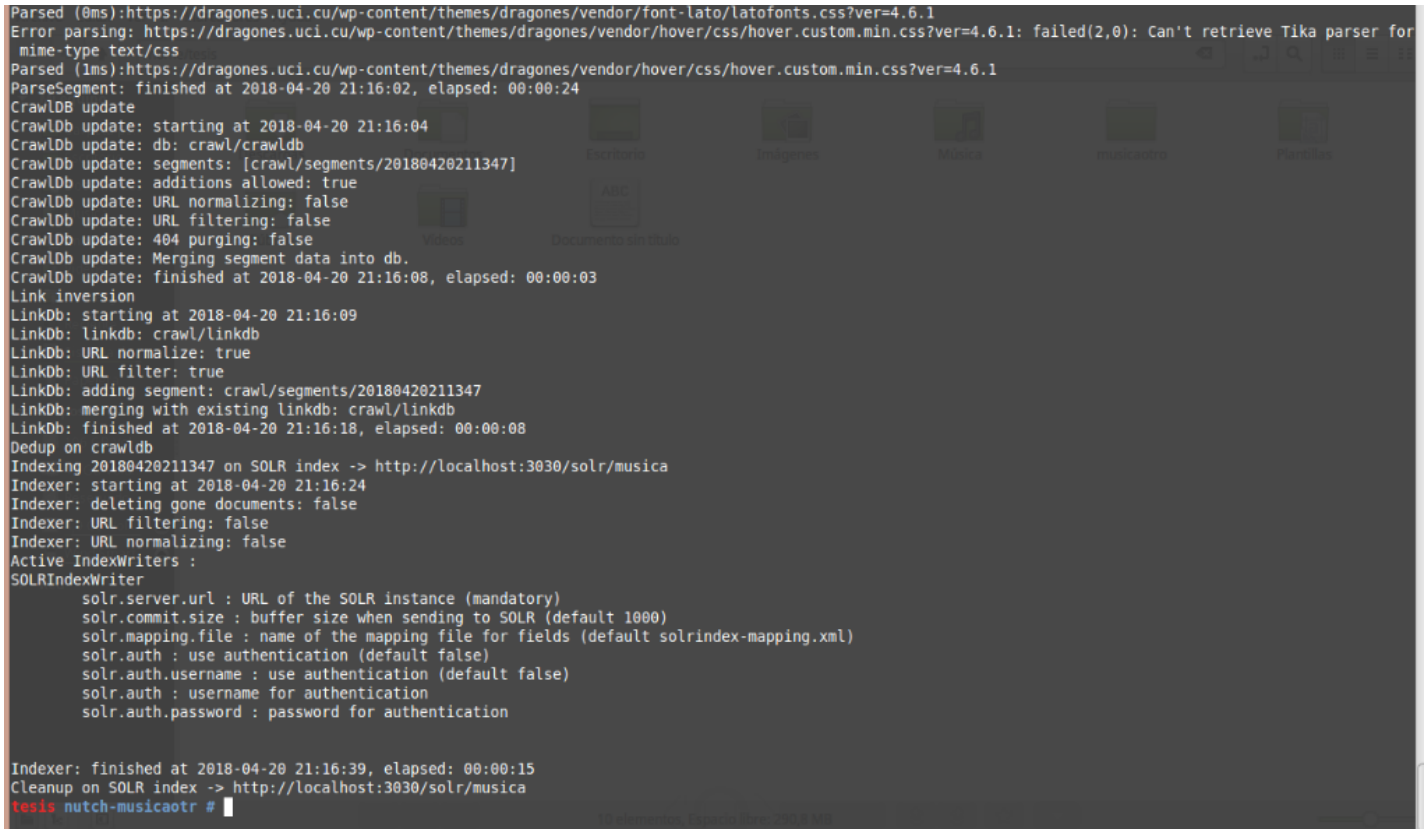

Figura 30. Integración de Apache Nutch con Apache Solr.

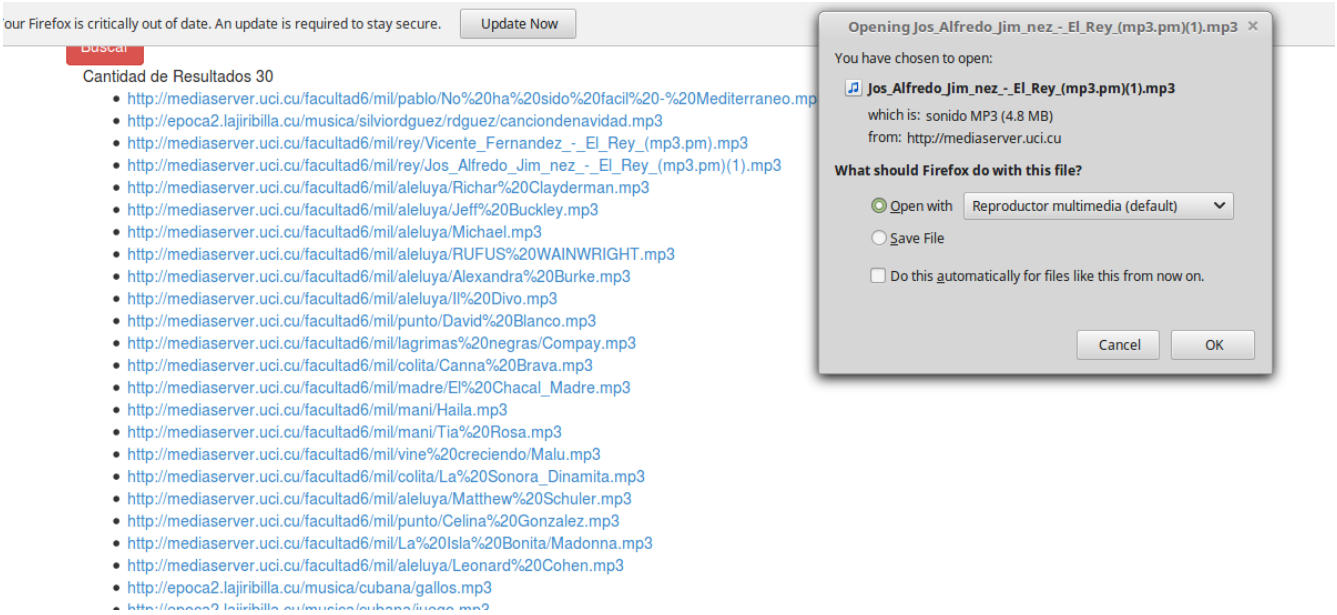

Figura 31. Integración de la interfaz web y Solr

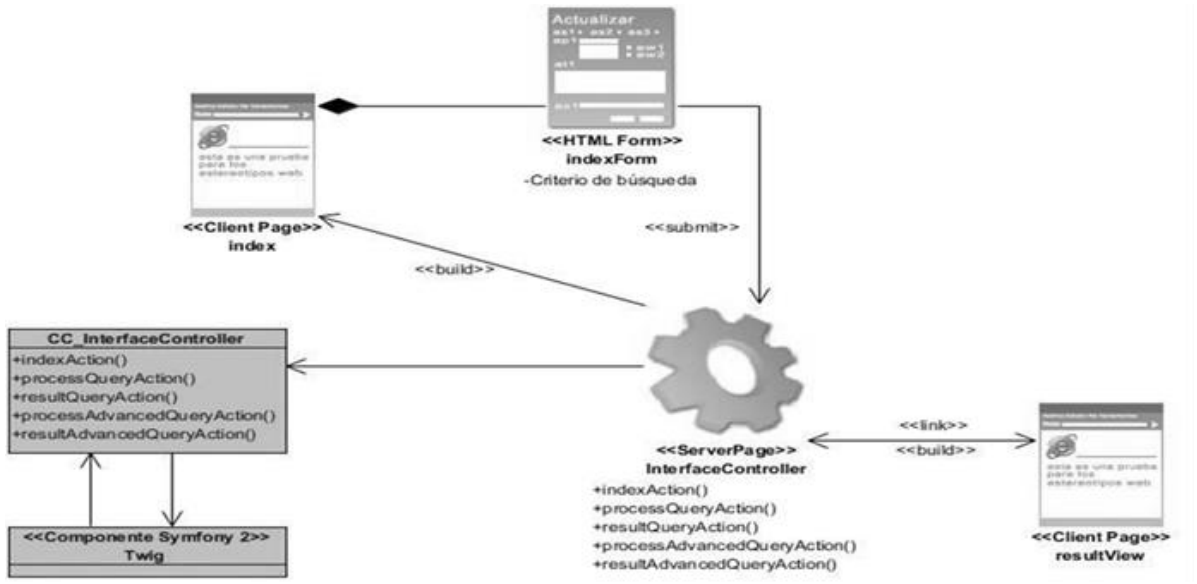

Figura 32. Diagrama de clases del diseño con estereotipos web del CU "Buscar Música"

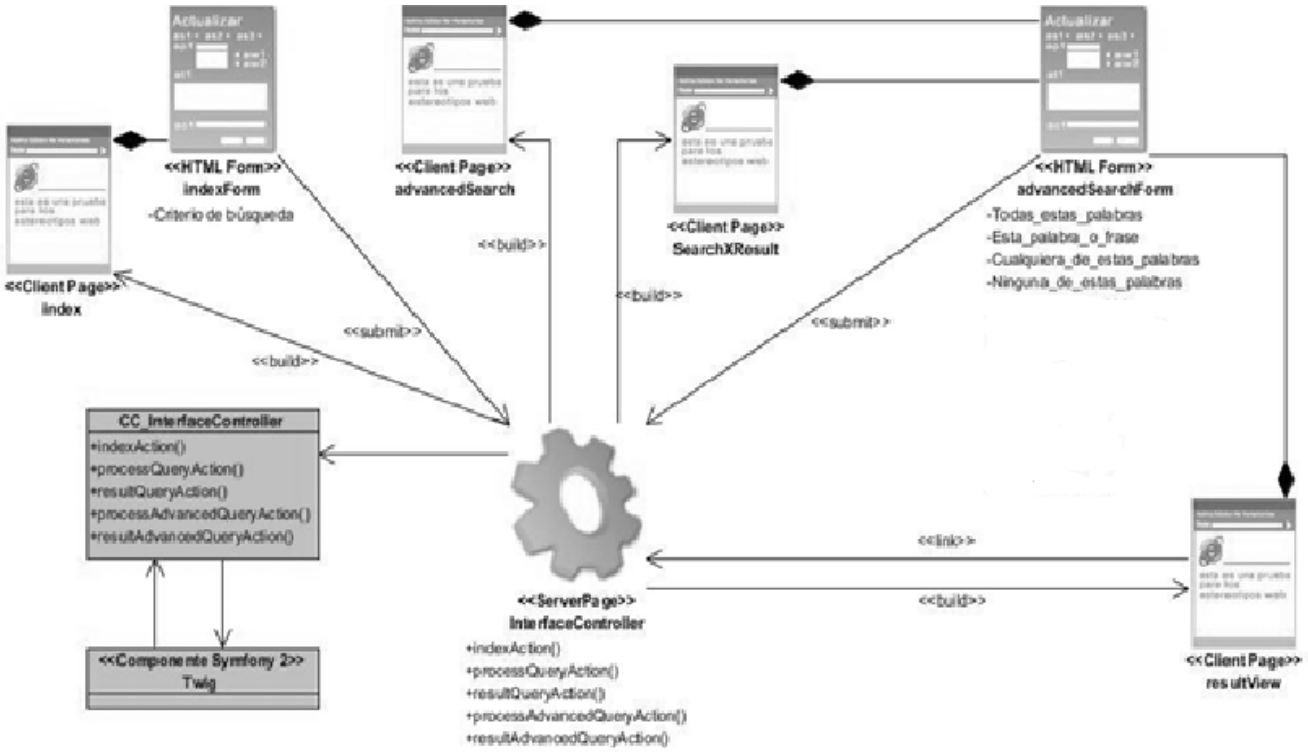

Figura 33. Diagrama de clases del diseño con estereotipos web del CU "Filtrar contenidos"

```
Mp3File mp3file = new Mp3File("SomeMp3File.mp3");
ID3v1 id3v1Tag;
if (mp3file.hasId3v1Tag()) {
  id3v1Tag =  mp3file.getId3v1Tag();
} else {
  // mp3 does not have an ID3v1 tag, let's create one..
  id3v1Tag = new ID3v1Tag();
  mp3file.setId3v1Tag(id3v1Tag);
}
id3v1Tag.setTrack("5");
id3v1Tag.setArtist("An Artist");
id3v1Tag.setTitle("The Title");
id3v1Tag.setAlbum("The Album");
id3v1Tag.setYear("2001");
id3v1Tag.setGenre(12);
id3v1Tag.setComment("Some comment");
    mp3file.save("MyMp3File.mp3");
```

Figura 31. Desarrollo del plugin de Nutch"

```
Mp3File mp3file = new Mp3File("SomeMp3File.mp3");
if (mp3file.hasId3v2Tag()) {
  ID3v2 id3v2Tag = mp3file.getId3v2Tag();
  System.out.println("Track: " + id3v2Tag.getTrack());
  System.out.println("Artist: " + id3v2Tag.getArtist());
  System.out.println("Title: " + id3v2Tag.getTitle());
  System.out.println("Album: " + id3v2Tag.getAlbum());
  System.out.println("Year: " + id3v2Tag.getYear());
  System.out.println("Genre: " + id3v2Tag.getGenre() + " (" + id3v2Tag.getGenreDescription() + ")");
  System.out.println("Comment: " + id3v2Tag.getComment());
  System.out.println("Lyrics: " + id3v2Tag.getLyrics());
  System.out.println("Composer: " + id3v2Tag.getComposer());
  System.out.println("Publisher: " + id3v2Tag.getPublisher());
  System.out.println("Original artist: " + id3v2Tag.getOriginalArtist());
  System.out.println("Album artist: " + id3v2Tag.getAlbumArtist());
  System.out.println("Copyright: " + id3v2Tag.getCopyright());
  System.out.println("URL: " + id3v2Tag.getUrl());
  System.out.println("Encoder: " + id3v2Tag.getEncoder());
  byte[] albumImageData = id3v2Tag.getAlbumImage();
  if (albumImageData != null) {
    System.out.println("Have album image data, length: " + albumImageData.length + " bytes");
    System.out.println("Album image mime type: " + id3v2Tag.getAlbumImageMimeType());
  }
}
```

Figura 32. Desarrollo del plugin de Nutch”

Tabla 18. Caso de prueba correspondiente al CU "Identifica Música en una página web".

Escenario	Descripción	V1	V2	V3	Respuesta del Sistema.	Flujo central
EC 1.1 Identificar Música.	Se le envía al sistema el árbol de la página web y configuración válida para que realice el análisis.	"http://mediaserver.uci.cu/facultad6/mil/David/Digale%20-%20David%20Bisbal%20(Acstica).mp3"	NA	NA	Metadatos de la música.	Crear una carpeta con el nombre "URLs" dentro de la raíz del rastreador. Crear fichero de texto con un nombre aleatorio y escribir dentro de él la URL del archivo musical que se describe en la variable1. Ejecutar en un terminal el comando: "bin/crawl-music urls/ crawl/ Solr 2", donde el número 2 va hacer la profundidad del rastreo en las URLs a la hora de identificar la música.
EC 1.2 Filtrar metadatos de un a	Se le envía al sistema toda la información			Contiene los siguientes metadatos: Tittle: "Dígale" Genero:"Acústica" Date: "2011-12-05T23:34:42Z"	Objeto que contiene todos los metadatos	Crear una carpeta con el nombre "URLs" dentro de la raíz del rastreador. Crear un fichero de texto con

archivo de música para ser indexado.	obtenida de un archivo de música para que lo filtre y posteriormente pueda ser indexada por el rastreador.			Type: “audio”, “mp3” Urls: “http://mediaserver.uci.cu/facultad6/mil/avDid/Digale%20%20David%20Bisbal%20(Acstica).mp3”	descritos en las variables 3, los cuales son los que se van a indexar.	un nombre aleatorio y escribir dentro la URL:“http://mediaserver.uci.cu/facultad6/mil/David/Digale%20-%20David%20Bisbal%20(Acústica).mp3” Ejecutar el comando: “bin/crawl-music urls/ crawl/ Solr 2”.

Tabla 19. Variables empleadas en el diseño del caso de prueba basado en el CU “Identificar Música en una página web”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	content	Variable JAVA (instancia de la clase org.apache.nutch.protocol.Content)	No	Contiene información referente al contenido de la página que se está analizando; tal como: URL, el binario del contenido, metadatos propios del protocolo de la URL y el tipo de contenido.
2	nutchMusicotr	Variable JAVA (instancia de la clase org.apache.nutch.indexer.NutchMusicotr)	No	Contiene los metadatos del documento que se va a indexar.
3	Parse-Music	Variable JAVA (instancia de la clase org.apache.nutch.parse.ParseMusic)	No	Contiene los metadatos extraídos durante el análisis del archivo musical.

Figura 33. Propuesta de solución.