Universidad Carlos III de Madrid

Escuela Politécnica Superior

BUSCADOR MUSICAL

Proyecto Fin de Carrera

Ingeniería Técnica de Telecomunicación
Sonido e Imagen

Autor: Héctor Murcia Carbonell

Tutor: Julio Villena Román

Cotutora: Raquel M. Crespo García

Diciembre de 2009

Título: Buscador Musical

Autor: Héctor Murcia Carbonell

Tutor: Julio Villena Román

Cotutora: Raquel M. Crespo García

EL TRIBUNAL

Presidenta: Florina Almenárez Mendoza

Secretaria: Estrella María García Lozano

Vocal: Carlos Álvarez Ortega

Realizado el acto de defensa del Proyecto Fin de Carrera el día 10 de Diciembre de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Secretario Fdo: Vocal

"… Al andar se hace camino,

y a al volver la vista atrás

se ve la senda que nunca

se ha de volver a pisar…"

-A. Machado-

Agradecimientos

Quiero agradecer a mis padres, Paqui Carbonell y Jesús Murcia, su entrega y dedicación de tantos y tantos años, para mí como una meta casi inalcanzable de ejemplo a seguir. Y su padecimiento en esta larga espera del proyecto, sin saber muchos porqués. Las circunstancias me hacen mencionar especialmente en estos momentos a mi padre, por todo lo duro que ha estado viviendo y porque en el dolor he descubierto que nos parecemos más de lo que podría imaginar. Y no relego en absoluto el papel insustituible de mi madre, a quien seguramente me parezco en todo lo que no cabía en su cromosoma y mucho más. Suerte la mía de recibir todo aquello que recibí.

Lucía ha sido, es, y será la persona que mejor me conozca en el mundo. Por conocerme mejor de lo que yo me conozco, y por creer en mí, más que yo mismo, nada de esto habría sido posible sin ella. Ella ha sido el mejor apoyo en los momentos más duros de esta última etapa y en cada día que hemos pasado juntos. Si alguna vez tuviera que contar lo bueno que en la vida coseché, no diría que poseí objeto ni persona, sin duda tendré que mencionar que ella estuvo junto a mí. Gracias, porque me enseñaste los agujeros de gusano, para poder viajar cuando no encontramos destino o no nos gusta.

Es justo decir que aprendemos de todo lo que pasa por nuestras retinas, desde que tenemos vivarachos ojos de bebé hasta que somos bien grandes -si nunca queremos dejar de aprender-. Por eso, tengo que recordar y agradecer que un día mi abuelo Segundo me picara con el gusanillo de los libros, porque nada es casual y seguro que ser lo que soy también tiene mucho que ver con él.

Gracias al cariño de toda mi familia, capitaneada por la "yaya Ela", formada por mis tías, y mis primos y primas que han sido como hermanos desde niños.

Alberto, mi querido frei, aún en la distancia, has sido un amigo muy especial. Me has enseñado tanto que si no hubiera aprendido, también contigo, que hay cosas imposibles de comprar, tendría una deuda eterna. Sé que el valor de tu sonrisa multicolor nunca se puede apreciar lo bastante, pero desde el primer momento revela que has encontrado a un tipo raro que se atreve a vivir feliz. Gracias por todo.

Gracias a María de los Ángeles, por compartir cada viernes mi preocupación por los avatares del proyecto, y por ser como eres.

Gracias a Julio por ofrecerme la oportunidad que necesitaba, por su gran trabajo y por su infinita paciencia, especialmente en los meses más faltos de inspiración.

En definitiva, intentar nombrar a todos es exponerme a cometer la injusticia de olvidarme de alguien. Por ello, gracias a todos los que compartieron camino alguna vez junto a mí, cuando la senda era un tranquilo paseo, pero especialmente durante los momentos en que se volvió una carretera cuesta arriba, con curvas, y a veces cuesta abajo sin frenos.

Resumen

El presente proyecto consiste en el diseño e implementación de un buscador musical cuyo objetivo es identificar melodías a partir de secuencias de notas y otro tipo de información introducida por el usuario. El sistema podrá recuperar, a partir de los tonos de las notas y el ritmo, las obras musicales que más se correspondan con la expresión de búsqueda. Este sistema está formado por dos bloques: en el primero se introducen elementos en el índice para constituir el conjunto de búsqueda -las melodías de obras musicales-, y el segundo permite realizar el proceso de consulta del buscador -encontrar entre el conjunto de búsqueda la obra musical que más se corresponde con una melodía determinada-. En el primer bloque se realiza una extracción e indexación de la información musical. La extracción se lleva a cabo a partir de ficheros MIDI, de los cuales se obtiene la representación de las melodías, y a partir de esta representación se analiza y agrega al índice el contenido musical. El segundo bloque realiza la búsqueda de melodías coincidentes en el índice creado con anterioridad. El proyecto aborda el diseño e implementación completos de todos los bloques del sistema.

Índice de contenido

Índice de tablas

Índice de figuras

1.Introducción

1.1. Motivación y contexto

En la actualidad, la cantidad de información de toda clase disponible de una forma inmediata es inmensa. No obstante, la posibilidad de aportar conocimiento a un determinado individuo dependerá de la capacidad de seleccionar, de entre toda la información disponible, sólo la información relevante para él. En el ámbito de las modernas Tecnologías de la Información y el Conocimiento, ha aparecido la obligación de dar respuesta a esta necesidad. Si bien no se puede sustituir el criterio y la capacidad de análisis intelectual de la persona, en los últimos años se han popularizado enormemente los llamados buscadores, como herramientas útiles en el proceso de adquisición de conocimiento, principalmente a través de la Red.

Los desarrollos más populares y, en cierta medida, pioneros en este ámbito, comenzaron orientados a la información textual. Pero a medida que la tecnología informática y la Red se han ido transformando en entornos multimedia, han surgido herramientas de búsqueda destinadas a satisfacer la necesidad de recuperar otros contenidos. Tanto la búsqueda de contenido textual como de material multimedia, se basaron en principio en la clasificación o creación de categorías, frecuentemente con la existencia de múltiples niveles de jerarquía como método de organización. Actualmente, la tendencia predominante, y de mayor éxito, consiste en que los buscadores de documentos de texto ya no se basen exclusivamente en una labor de etiquetado previo, sino sobre todo en el análisis del contenido de los documentos.

En el caso de documentos multimedia, es cada vez más evidente la necesidad de emplear métodos que sean capaces de extraer y representar los contenidos, del mismo modo que en documentos textuales. Aún hoy, las herramientas predominantes de uso generalizado para la búsqueda de medios como vídeo, imágenes, etc. no presentan un enfoque basado en contenido, sino en clasificación (por ejemplo mediante *tags* o etiquetas). Sin embargo, sí existen numerosos desarrollos que posibilitan la recuperación basada en contenido. En el caso concreto de la música, se presenta una alternativa de especial interés: los contenidos musicales proceden de la interpretación de un lenguaje propio, el lenguaje de la notación musical.

El presente proyecto ha surgido con objetivos similares a los de Musipedia [42], que ofrece la posibilidad de buscar y localizar fragmentos de obras musicales a partir de las notas que conforman el contenido musical. De tal manera, se mejora un sistema de recuperación que resultaría incompleto si se empleara exclusivamente la clasificación. El proyecto expuesto a continuación pretende diseñar un buscador que pueda constituir una alternativa equiparable en términos de eficiencia a la mencionada aplicación. No en vano, hoy en día existen numerosas alternativas a considerar para construir un sistema de recuperación de información, y por tanto las soluciones posibles para satisfacer un mismo objetivo son múltiples. De tal modo, se ha concebido como un reto de máximo interés la tarea de diseñar e implementar un nuevo buscador empleando herramientas de recuperación que hasta ahora no se habían aplicado en este ámbito.

1.2. Objetivos

El objetivo fundamental de este proyecto es desarrollar un sistema buscador musical basado en melodías. En primer lugar, se extraerán las melodías a partir de los ficheros MIDI que las contengan, y se seleccionará la información necesaria con la finalidad de ser agregada a un índice. Mediante el empleo del índice, el sistema podrá satisfacer las consultas efectuadas en el buscador, las cuales podrán consistir en la introducción de una serie de notas junto con otro tipo de información descriptiva. El sistema debe devolver, ante dichas consultas de usuario, aquellas melodías más relevantes con respecto a la solicitud de información.

Se estudiarán las características de una melodía que la describen de forma única e inequívoca y cómo deben ser combinadas para obtener resultados óptimos en la búsqueda. Se deberá conseguir, de manera lógica, un sistema de representación válido para aplicar a las melodías almacenadas y a las melodías introducidas como consulta de búsqueda. La información almacenada tendrá que ser suficiente como para que el sistema sea capaz de tolerar omisiones o errores del usuario. Además se permitirá el uso de metadatos, información descriptiva de las obras (como por ejemplo: título, autor, etc.) para dar más riqueza a las consultas.

Para tales fines, será necesario emplear la teoría actual en relación a la Recuperación de Información, así como los nuevos desarrollos en este ámbito, que en este caso se particularizan en la API de Lucene, detallada convenientemente en la exposición de este proyecto.

Se requerirá que el sistema sea eficiente en la resolución de búsquedas, y la eficiencia del resto de tareas ejecutadas previamente se evaluará también de acuerdo a este rendimiento final. Además, debe existir un compromiso con la agilidad y rapidez como parámetros importantes de calidad en la aplicación. El adecuado uso de los desarrollos antes citados facilitará esta misión.

El sistema debe adecuarse a los requisitos necesarios que permitan su uso como aplicación web. Principalmente en lo referido a las operaciones de búsqueda, éstas se han de plantear orientadas a un escenario de este tipo. Por ello, un objetivo secundario planteado es la integración de una página web de consulta que pudiera ampliar el público objetivo a cualquier usuario de Internet.

Se realizará un diseño de carácter modular, dirigido a la obtención de un desarrollo que facilite la interoperabilidad y futuras ampliaciones para la adición de nuevas funcionalidades.

1.3. Estructura de la memoria

La memoria de este proyecto esta organizada en 7 capítulos.

El Capítulo 1 recoge la presente Introducción.

El Capítulo 2, de introducción musical, pretende exponer las bases teóricas en relación al arte y ciencia de la Música, sobre las cuales se asienta el diseño y desarrollo de este sistema.

El Capítulo 3, Estado del Arte, presenta los precedentes, así como la situación actual y fundamentos, de las tecnologías involucradas en la creación de este sistema.

El Capítulo 4, relativo al Diseño, recoge la estructura modular concebida para la creación de la aplicación y la funcionalidad asociada a cada elemento, junto con las decisiones debidamente justificadas que han determinado diversos aspectos del diseño.

El Capítulo 5, Implementación, describe el modo en que han sido realizados los diferentes módulos planteados en la fase de diseño, las clases Java programadas, así como las librerías externas empleadas.

El Capítulo 6, Validación del sistema, recoge la descripción de las pruebas

efectuadas para verificar de forma empírica el funcionamiento del sistema, de acuerdo a los parámetros previstos.

El Capítulo 7, incluye las conclusiones obtenidas al término del presente proyecto y los trabajos futuros que podrían constituir una continuación del mismo.

2.Introducción musical

2.1. Introducción histórica

La música es la sexta de entre las artes clásicas, y se puede remontar su origen desde tiempos prehistóricos hasta llegar a la actualidad. No es ni más ni menos que una expresión artística que se vale del sonido para manifestarse. Se cree que ésta se desarrolló en la raza humana a la par que otras formas de expresión, incluyendo el lenguaje oral. Es prácticamente imposible situar en un punto cronológico exacto la aparición de la música, ya que los restos arqueológicos no son un factor decisivo. Los únicos objetos inequívocamente ligados a la música son instrumentos musicales que se hayan podido identificar (tras una tarea no siempre sencilla) como tales, y los hallazgos para épocas muy antiguas son escasos. Lo más probable es que la música se empezará a desarrollar golpeando objetos habituales del entorno o incluso el propio cuerpo, y especialmente con la progresiva aparición de una mayor habilidad para controlar los sonidos producidos por las cuerdas vocales [12].

Es posible hablar de las primeros hallazgos significativos y relacionados a ciencia cierta con la interpretación musical en torno al final de la prehistoria y comienzo de la Edad Antigua, en Sumeria hacia el año 3000 a. C. y en posteriores culturas mesopotámicas, a través de Egipto hasta las tierras mediterráneas orientales en general. Algunos tipos de instrumentos y algunas prácticas (liras y arpas, la flauta doble, el canto de responsorios y antifonal) están documentados desde la época histórica más antigua (con la aparición de la escritura). Las liras, las arpas, y las flautas dobles eran conocidas en tierras sumerias, en Egipto desde los tiempos del Reinado Antiguo, así como en Creta, Micenas, y en toda Grecia en general desde siglos antes de Homero, en Asiria y en Babilonia. Los romanos heredaron muchos de estos instrumentos, como la flauta doble que les llegó de manos de los etruscos. Los instrumentos de percusión (generalmente atonales), de distintas clases, pero sobre todo los tambores y matracas, son universales, lo cual no sucede por ejemplo con las trompetas, que eran usualmente meros instrumentos para emitir llamadas antes que musicales, a decir verdad. La difusión de otras familias de instrumentos, como los de tipo laúd, está menos documentada pero, incidiendo en el ejemplo concreto, instrumentos similares se citan en Babilonia en torno al 2000 a. C. El canto para responsorios y el antifonal en los templos ya se practicaban en Sumeria, era conocido en Egipto y quizá continuó con

pocos cambios cuando los hebreos lo adoptaron para cantar los salmos. Instrumentos como las flautas de pan, de aparición relativamente tardía (durante el primer milenio a. C.), supusieron el antecedente de los grandes órganos, cuando se inventaron las bolsas o cajas para mantener la presión del aire constante que producía el sonido en los instrumentos de viento [1].

Esta es una distinción básica. La música es durante muchos años, la música de los reyes y los sacerdotes, los poetas o los filósofos, sin embargo, la música más común, la del pueblo ha dejado pocos testimonios escritos en muchas épocas. La música más antigua de la que se tiene certeza documental, mesopotámica y egipcia, fue la música de los templos y palacios, y posteriormente se tienen noticias de la música que buscaba entretener en los dramas griegos y circos romanos, de canciones nupciales y de cantos de vendimia, pero éstas son conocidas por el testimonio oral, en muchos casos sin prueba documental.

I)Pitágoras y las innovaciones de la Grecia Clásica

La civilización griega recoge presumiblemente todos los descubrimientos que en materia musical realizaron los pueblos mesopotámicos y añade otros nuevos. Así, por ejemplo, aunque es muy probable que la mayoría de los instrumentos que tocaban en Mesopotamia y Egipto no llegaran a la precisión del semitono (concepto que se explicará más adelante), sí es muy posible que se hubiesen aprendido las relaciones entre la altura de la nota y la longitud de la cuerda, entre las proporciones 1:2, 2:3, 3:4 y los intervalos de octava, quinta y cuarta. De esta forma, pudieron venir ya construidas las bases sobre las que se asentó la teoría matemática en relación a la música de Pitágoras. La teoría pitagórica se entiende como un todo, la concepción de un universo construido de acuerdo a reglas matemáticas, del cual la música forma parte [8]. Existe una teoría bastante consistente que defiende que el sistema precursor musical sumerio-babilonio, fuera cual fuese, tuvo una base científica organizada sobre una teoría cósmica y bien se podría intuir, por lo que se sabe de los egipcios, que su sistema musical, también un gran desconocido, fue alcanzado por la vía empírica [1].

Influencias aparte, las enseñanzas de Pitágoras de Samos suponen para la música, como también para otras muchas áreas del conocimiento, una innovación de importancia excepcional. Durante la segunda mitad del siglo VI a. C. desarrolló con su escuela de discípulos una amplia variedad de campos de conocimiento, entre los que también se encontraban la matemática "pura", la metafísica, política y ética. En lo que respecta al arte musical, además de fijar las relaciones matemáticas que le servían de sustento, destacó la influencia ética de la música en

el hombre [8]. Sobre todos estos aspectos hay que resaltar que Pitágoras no dejó escritos de ningún tipo, y todo los textos que se pueden encontrar fueron escritos por autores del siglo IV; no es fácil delimitar claramente aquellas conclusiones y/o invenciones que le pertenecen y, de hecho, se suele hablar de la escuela pitagórica en general, como evidencia de que no se sabe si un determinado concepto proviene del pensador de Samos o de alguno de sus discípulos.

Figura 1: Arpa babilonia heredada por la cultura griega.

Uno de los instrumentos más habituales de la época es una clase de arpa (heredada de la cultura sumeria). Primero fue un arpa de cuatro cuerdas, de la que se podía extraer una escala de cuatro únicas notas, obtenidas tras afinar las cuerdas con una tensión y longitud específica de acuerdo a las proporciones estudiadas. Las cuerdas se afinaron para producir intervalos: de dos tonos (el tono pitagórico correspondía a la proporción 8:9) y de un leîmma o semitono (el resto de la cuarta después de la sustracción de dos tonos). Aparecieron arpas que reproducían de forma aproximada algunas notas actuales, como por ejemplo: Mi 3, Fa 3, Sol 3, La3; y después de añadirles a éstas tres cuerdas más se podían producir las equivalentes a Sib 3, Do 4, Re 4. Estos sonidos se nombraban como "néte", "paranéte", "tríte", "mése", "lichanós", "parhypate" e "hypate", para hacer referencia no al sonido concreto (que variaba en cada instrumento), sino a la cuerda que se tocaba (la cuerda inferior, la cuerda próxima a la inferior, la media, etc.) [1] . Las arpas se irían sofisticando para llegar a instrumentos de hasta 40 cuerdas, y éstas supondrían un claro precursor de un instrumento mucho más moderno como el piano.

La música griega era esencialmente melódica, más que armónica, dicho de

otro modo, más centrada en la interpretación de sonidos sucesivos que en los simultáneos. Se caracterizaba por ser microtonal, es decir, su escala contenía muchos más sonidos que la escala actual de doce sonidos del mundo occidental. Esto no es algo inusual en las tradiciones musicales orientales donde la música es enteramente melódica. Los intervalos más pequeños no se pueden escribir en la notación actual aunque algunos cantantes modernos e instrumentalistas de jazz los ejecuten.

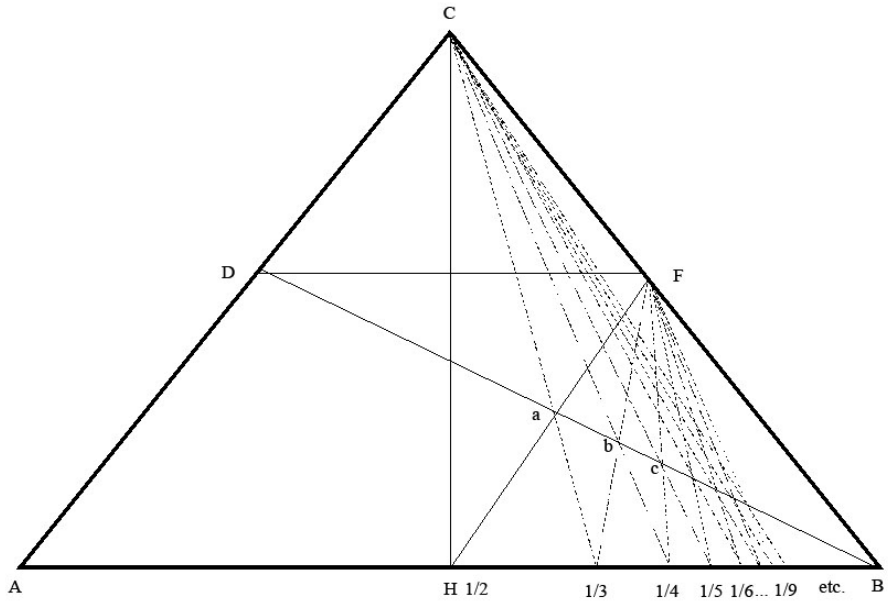


Figura 2: Método geométrico definido por Pitágoras para obtener los intervalos de un instrumento.

La llamada afinación pitagórica surge de descubrir que al dividir una cuerda a la mitad, razón 1:2, se producía un sonido que era una octava más agudo que el original (por ejemplo la distancia que existe en las escalas actuales del Do al Do superior); que cuando la razón era 2:3 se producía una quinta (en el sistema actual, la distancia de Do a Sol) y que otras razones sencillas producían sonidos que se juzgaban como agradables [8]. No fue hasta el siglo XVII de nuestra era cuando fray Marin Mersenne formuló las reglas físicas que dan explicación científica a este hecho subjetivo: estableció que se producían determinadas relaciones de frecuencias y sus armónicos al seguir estas proporciones. A pesar de este desconocimiento, los experimentos de la escuela de Pitágoras con el monocordio condujeron al método de afinación con intervalos en razón de enteros conocido como la afinación pitagórica. La escala producida por esta afinación se llamó escala pitagórica diatónica y fue usada durante muchos años en el mundo occidental. Se deriva del monocordio y de acuerdo con la doctrina pitagórica, todos sus intervalos

pueden ser expresados como razones de enteros. Parte del axioma que obliga a cualquier intervalo a expresarse como una combinación de un número mayor o menor de quintas perfectas. Se toma una nota base y se obtienen las demás notas de una escala diatónica mayor encadenando hasta seis quintas consecutivas por encima y una por debajo, lo que da lugar a las siete notas de la escala [18] . Existen diferencias de afinación entre esta escala y la escala temperada usada actualmente. Sin embargo, es posible hacer un ejercicio de aproximación; así por ejemplo, si se hace la suposición de que se parte de la actual nota Do, se obtiene:

Fa ← Do → Sol → Re → La → Mi → Si

Al continuar el encadenamiento de quintas más allá de estos límites, se encuentran el resto de las doce notas de la escala cromática, formando un círculo de quintas que no llega a cerrarse:

Mi b ← Si b ← Fa ← Do → Sol → Re → La → Mi → Si → Fa # → Do # → Sol #

Los grandes teóricos de la música en la época helenística, casi siempre eran a la vez filósofos, pensadores, artistas, matemáticos, científicos y un largo etcétera. No se concebía de otra manera que no fuera en un todo: arte, ciencia y pensamiento, afirmación que se ha mantenido con desigual fuerza a lo largo de la historia. Por ello, hay que citar también a Platón, Aristóteles y Aristóxeno como algunos de los grandes contribuyentes a las ideas musicales de la época.

Destaco Aristóxeno de Tarento, discípulo de Aristóteles, por varias aportaciones relevantes: entre otros aspectos, se encuentra mención en su obra a uno de los primeros sistemas prototípicos de notación musical, si bien la descripción que se hace de ello es considerablemente incompleta. Es a través de estudiosos posteriores, sobretodo Alipio, como es posible hacerse una idea más aproximada de las representaciones ideadas en la época. La notación sólo indicaba los intervalos, no su posición en el tetracordio (el arpa de 4 cuerdas habitual), y supone conocidos muchos aspectos de la interpretación que no se declaran explícitamente. La parte esencial del sistema consistía simplemente en signos para una escala diatónica de dos octavas. Seis notas más altas se indicaban agregando una tilde (´) a las notas segunda a séptima de la serie y otros signos transcribían dos notas más bajas. Es hecho conocido que existían más alturas intermedias que las que se interpretan en la actualidad, por ello un signo acostado sobre la espalda del símbolo indicaba que se había agudizado su sonido en un microtono y cuando estaba transcrito en espejo señalaba una agudización de dos microtonos [1] .

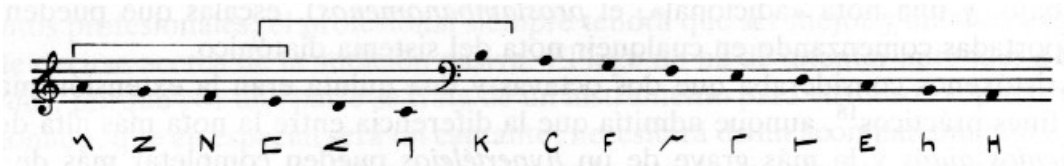


Figura 3: Notación simbólica de Aristóxeno y equivalencia aproximada con la actual

Posteriormente apareció una notación pensada para la interpretación vocal, de autoría incierta, pero que fue muy utilizada. Abarcaba sólo una de las actuales escalas (lo que se denomina como una octava), sin embargo daba cabida a más variaciones tonales que una escala actual, debido a que existían los ya citados microtonos entre un tono y otro. Esta escala estaría formada por veinticuatro sonidos progresivamente descendentes representados por las veinticuatro letras del alfabeto griego: alpha (la nota más aguda elevada dos microtonos), beta (la misma nota, elevada un microtono), gamma (la nota sin elevación), delta (la segunda nota más aguda elevada dos microtonos), y así sucesivamente. Para indicar tonos más agudos se agregaba en ocasiones una tilde (´) a las quince primeras letras y se invertían los caracteres de las últimas seis; en los casos en que se indicaban tonos más graves, se realizaba también una especie de inversión de las letras. Si bien sentaron un claro precedente, la correspondencia de los sonidos que se interpretaban con estas indicaciones y las notas actuales no se puede establecer con claridad. Existe una teoría suficientemente sostenible de la autora Isabel Henderson [21], que equipara esta escala a la secuencia de notas de La 3 a La 2, con unos sonidos peculiares indicados con un símbolo de "medio sostenido": estos serían los microtonos.

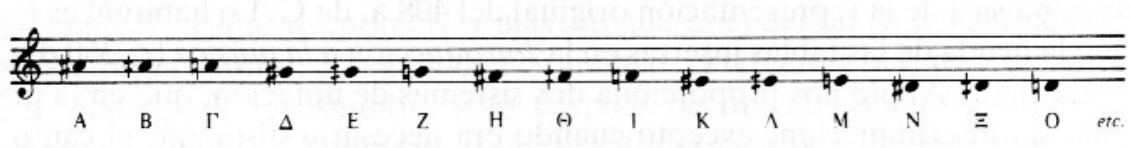


Figura 4: Notación de música microtonal griega. Aproximación a la actual [1].

Los griegos no mostraron interés por indicar en su notación nada acerca de las duraciones o los ritmos. La música que interpretaban tenía una base poético-métrica y no necesitaban tal cosa; generalmente el cántico marcaba el ritmo, que además se presentaba de una manera bastante uniforme.

II)El dominio de Roma

En los siglos sucesivos se producirían notables avances en la fabricación de instrumentos. Uno de los más importantes, que al parecer ya gozaba de cierta trayectoria cuando el incipiente imperio romano lo situó entre sus preferidos, fue el órgano. Según Ateneo se puede atribuir su invención a Ctesibio de Alejandría, un ingeniero de mediados del siglo III a. de C. que construyó varias máquinas explotando el poder del aire comprimido, y en particular del comprimido por el peso del agua. Por el contrario, si se atiende a lo expuesto por Tertuliano, el inventor del *organon hydraulos (hydraulus* en el latín romano) sería Arquímedes, contemporáneo en cualquier caso de Ctesibio; no en vano, se tiene constancia de que éste realizó experimentos con el aire que pasaba por un cuerpo merced a la presión que el agua ejercía.

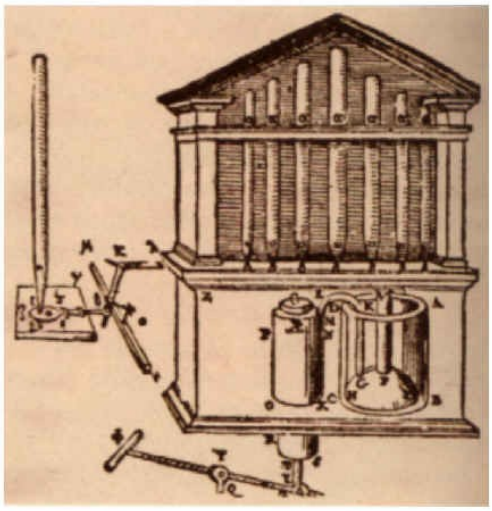


Figura 5: Boceto del instrumento hidráulico de Ctesibio

La música seguiría por mucho tiempo fuertemente ligada a las bases griegas, aunque se van incorporando numerosos elementos nuevos que convertirían a la época romana en un periodo de florecimiento musical. En Roma ya se interpreta a diferentes voces, aunque según un esquema que se puede denominar de heterofonía, el cual no se puede enmarcar aún en unas reglas ni de armonía, ni de contrapunto, como se conocen en la actualidad. La escala de sonidos existente sería equivalente a la que se denomina como diatónica, por tanto, con menos variedad tonal que la griega, aunque las alturas se indican con las antiguas notaciones. Aparece, por el contrario, una mayor diversidad rítmica, que ha de ser reflejada en la notación; se distinguen cinco duraciones posibles de un sonido, con sus cinco símbolos respectivos: la unidad menor es un *chronoi*, y se contempla la duración de dos, tres, cuatro y cinco *chronoi*. El ritmo musical puede ser indicado

de forma que sea independiente de la métrica de los versos, aunque, de acuerdo con la herencia griega, sigue marcando la pauta en muchos casos [1].

III)El dominio cristiano

Las fronteras de la cultura helenístico-romana comenzaban a difuminarse al tiempo que el cristianismo empezaba a ganar importancia religiosa y política, y el centro de gravedad cultural se trasladaba en buena medida a Bizancio. En este contexto la música floreció auspiciada por los grandes himnos religiosos.

Se puede decir que durante el primer milenio cristiano las antiguas notaciones hechas mediante letras desaparecieron del uso práctico para ir a dar a las manos de los teóricos, mientras otro sistema de signos, desarrollado a partir de los signos de acento que en sus orígenes regulaban la "pronunciación en voz alta" (ekphónesis) de los textos fue la raíz de donde crecería la notación práctica de occidente. Boecio utilizó letras latinas en su *De Institutione Musica* como una abreviación de los nombres griegos utilizados para indicar las notas en los tetracordios; de modo que el *hypáte méson* podía denominarse simplemente "D" (el *hypáte* del centro del tetracordio), el *parhypáte méson* sería "E", y así todos los demás. Provocaba gran confusión que Boecio utilizase la letra "A" para denominar la nota más baja, mientras que en la notación "vocal" del griego alpha representa la nota más aguda [1] .

La notación mediante letras indicaba la altura de cada nota y por consiguiente, de acuerdo al sistema de los tetracordios, su relación de intervalo con sus vecinas; una nueva notación, a la que se llamo ecfonética, marcaba la acentuación correcta, la inflexión y la puntuación, sin indicar ninguno de esos intervalos. Cuando por primera vez estos signos ectofonéticos se utilizaron para señalar la altura, derivando de los signos prosódicos helenísticos, no podían indicar más que relaciones indeterminadas dentro de las estructuras melódicas más pequeñas. Se piensa que empezaron a ser empleados en los leccionarios bizantinos en torno a la segunda mitad del siglo IV y un sistema de quince signos aparece en los manuscritos del siglo VIII [9] . El sentido de altura de los sonidos era en todo caso relativo: la *oxeîa* (acento agudo) indicaba una elevación, la *bareîa* (grave) un descenso, la *syrmatiké* (circunflejo, escrito como "~") una caída y una elevación. El conocimiento de estos recursos se expandió sólo de forma gradual hacia el oeste, y en muchas ocasiones es evidente la incapacidad de representar todo lo que los instrumentos y las voces podían conseguir. Así queda demostrado en muchos escritos de la época; a modo de ejemplo Isidoro de Sevilla decía: "si los sonidos no se conservan en la memoria, se pierden, porque no pueden ser escritos" [20] .

Durante largo tiempo la capacidad de notación de la música fue un paso por detrás de la capacidad interpretativa; las nuevas posibilidades musicales, surgidas evolución de los instrumentos y de la técnica de los intérpretes -tanto instrumentales como vocales-, no podía registrarse de un modo suficientemente estricto y eficaz.

A mediados del siglo IX aparecen los neumas latinos, precursores de lo que más tarde sería el sistema que se consideró idóneo para la notación musical: los pentagramas. La etimología de la palabra es griega, neûma significa señal, asentimiento mediante una inclinación de la cabeza. Los neumas latinos tienen un clara influencia de los neumas bizantinos, aunque no son idénticos a ellos. Uno de los ejemplos más arcaicos, en un manuscrito de St. Amand fechado en el 871, tiene los neumas anotados sobre el texto griego con caracteres latinos del Gloria, en tanto que la versión latina en la columna paralela no los tiene. En el curso del siglo X, los neumas aparecen en múltiples lugares, en especial en lo que hoy es Francia -St. Amand, Lyón, París, Reims, Corbie, Metz, Laón, Noyón-, pero también en St. Gall, Regensburgo, Essen, y Winchester. Pero hasta ese momento no aparecen casi en ningún lugar de Italia, con la excepción de un fragmento dudoso hallado en la biblioteca del Capítulo de Módena. Existían variedades locales de forma, pero a pesar de las diferencias, fundamentalmente caligráficas y en el significado de algunas ornamentaciones, las formas originales básicas eran siempre las mismas [1] .

El primer occidental que escribió acerca de los neumas, Aureliano de Réomé, cuya obra *Musica Disciplina* data de mediados del siglo IX, simplemente traduce alguno de los nombres griegos: de esta forma, al signo empleado para subir dos notas lo llama *acutus* (más tarde fue conocido como *podatus* o *pes*), al que utilizó para bajar dos notas *gravis* (más tarde, *clivis* o *flexa*). Otro escritor entre el siglo IX y X, el alemán del sur al que se conoce como *Anonymus Vaticanus* (por ser su escrito anónimo hallado en la Biblioteca Vaticana), revela que los neumas también podían indicar el ritmo:

«La modulación de la voz queda mostrada por los acentos de la nota y por el pie silábico. Por los acentos de la nota, en realidad, en el *acutus*, *gravis* y *circumflexus*; por el pie silábico en *brevis* y *longa*. De estos acentos ha surgido la figura conocida como neuma. Si es simple y de una *brevis*, forma un *punctum*; si es una longa es prolongada (*producta*). Pero este *punctum* se manifiesta de tres maneras: en *brevis*, *gravis* y *subpositus*. La *longa*, asimismo, se manifiesta de tres maneras: en la forma prolongada, en *acutus* y en *circumflexa*. Además, pueden ser

unidos en un ascenso o descenso de dos notas, tres, cuatro, cinco, seis, hasta el séptimo u octavo grado, como en el *Aleluya Beatus vir sanctus Martinus*…»

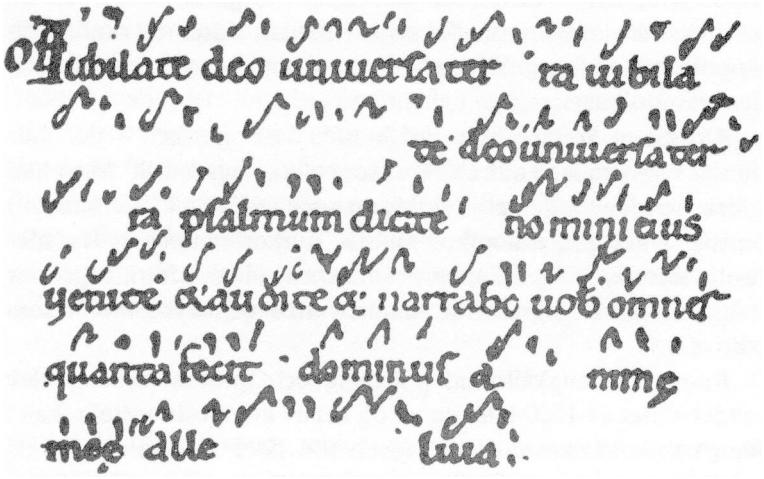


Figura 6: Ejemplo de neuma primitivo

Puede que el texto anterior del anónimo vaticano no sea suficientemente clarificador, y algunos de sus términos técnicos no son los mismos que se emplean en un periodo posterior. Pero es evidente que los neumas indican un ascenso y un descenso de altura y la duración (se puede establecer que una *longa* era una unidad de tiempo que equivalía aproximadamente al doble de una *brevis*).

Los grandes imperios culturales de las Iglesias de Oriente y Occidente fueron poco a poco separándose por motivos dogmáticos (entendidos también en un contexto histórico y político), hasta llegar a una separación definitiva en 1054. Esto supuso también evoluciones dispares de las formas musicales en cada área de influencia. Oriente, Bizancio, siguió fundamentalmente en unos modos interpretativos monofónicos (a una sola voz) y usualmente sin apoyo instrumental, por lo que la notación con neumas cubría sus necesidades, sólo en ocasiones se requerían algunos elementos adicionales. Mientras, en las tierras mediterráneas occidentales se producían desarrollos muy importantes, y la aparición de elementos considerablemente más complejos, como los himnos polifónicos, hizo replantearse la importancia de una teoría musical más exhaustiva y una notación más precisa.

IV)Dominio de Europa Occidental. Surgimiento de las actuales bases musicales

En torno al comienzo del segundo milenio es notable la extensión del arte musical, tanto del que se podría denominar profano o popular, como del religioso.

Del primero no se conserva demasiado en la actualidad, aunque sí existe una intuición fundamentada acerca de su importancia e incluso influencia en el segundo, éste de carácter eminentemente más culto y que sobrevivió en su forma y en sus frutos (muchas de las innovaciones que provocó y que siguen vigentes) [12] .

Uno de los hitos de mayor notoriedad lo constituye la aparición de la polifonía. La primera mención al respecto la hace Hucbald, un monje de *St. Amand* en el tratado *De Institutione Harmonica*, donde se define la *Consonantia*: "Es la mezcla prescrita y armoniosa de dos sonidos que se producirán sólo en el caso de que dos notas diferentes coincidan al mismo tiempo en una *modulatio*, como cuando la voz de un hombre y un niño cantan al mismo tiempo; o también, en lo que la gente comúnmente denomina *organizatio*".

Este hecho marca la necesidad de una notación de altura más precisa que los neumas, y así en el mismo tratado de Hucbald se encuentra una notación que indica los intervalos consecutivos precisos entre notas. Dibuja seis líneas paralelas que corresponden a las cuerdas de la cítara, pone letras en el margen para señalar el intervalo que existe entre ellas (T para un *tonus*, S para un *semitonus*) y escribe cada sílaba muy cerca de la línea de altura relevante [1] .

Las seis cuerdas del diagrama de Hucbald podrían haber representado un papel importante en la práctica medieval, pero no fue completamente adecuado para un sistema de notación general: resultaba demasiado complejo y, de hecho, él mismo utiliza los neumas en otros pasajes de su tratado, porque mediante ellos le era posible indicar la ralentización y la aceleración, así como los *tremula* de la voz y otras particularidades. El autor sustentaba que su sistema derivaba de Boecio, como una versión simplificada y actualizada de las propuestas de éste, aunque en rigor se remonta a Aristóxeno [1] . Su sistema, expuesto con la ayuda de diagramas de tonos y semitonos, consiste en esencia en dos octavas idénticas conjuntas:

$$_{TSTTSTT}{}^{TSTTSTT}$$

Hucbald detalla que un tono completo es el intervalo que existe entre las primeras dos cuerdas de un instrumento, cuando está afinado como corresponde, o entre los dos primeros tubos de un órgano (*hydraulus*). En uno de sus diagramas, además, emplea las letras del alfabeto latino para denominar a cada sonido de la escala, de forma que se puede establecer una correspondencia aproximada con la notación anglosajona vigente (en la que las notas van desde la A la G), aunque en este caso de la serie va desde la A a la P. La distancia entre estos sonidos podía ser

de tono o semitono, según el caso (de la misma manera que se establece en la actualidad).

De forma prácticamente contemporánea a los escritos de Hucbald, debieron escribirse unos manuscritos titulados *Musica Enchiriadis* que recogían la polifonía musical en un sistema de notación muy similar que utilizaba exactamente los mismos conceptos que éste.

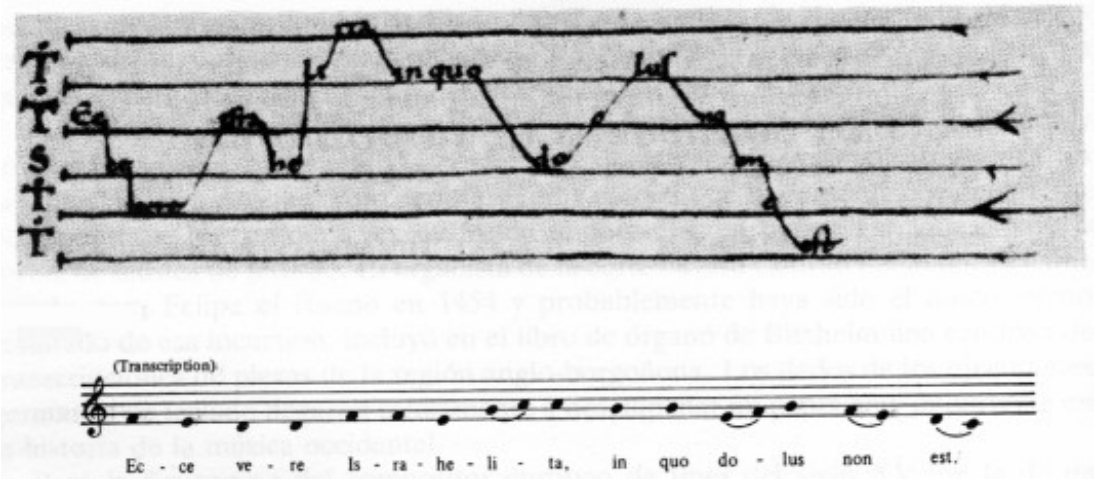


Figura 7: Obra musical según la notación de Hucbald y transcripción aproximada a la actual [1]

Estos fueron los comienzos teóricos un sistema de representación que habría de distinguir la música occidental de las del mediterráneo oriental y en general de las del resto del mundo, antes de que éste se empezara a divulgar. En rigor, es difícil definir de un comienzo marcado, y resulta más apropiado hablar de un proceso gradual marcado por los diferentes hitos que se han ido detallando, provocado por una evolución desde la heterofonía a la polifonía.

Los sistemas presentados por Hucbald y por otros autores de la época constituyeron el intento de representación más completo hasta aquel momento, si bien, el actual sistema centrado en el pentagrama provino de evoluciones posteriores más sofisticadas de los neumas. La evolución comenzó escribiendo los neumas a diferentes alturas para marcar las diferencias de tono de los sonidos, pero la notación de altura era inexacta, sin la precisión que empezaba a ser necesaria para la perfecta armonía polifónica. Posteriormente se añadirían varias líneas horizontales que permitieran marcar la altura con más precisión. De tal manera, los neumas se convirtieron en un medio cada vez más adecuado para anotar la altura en lo que se puede calificar como el "pentagrama embrionario". Paulatinamente, éstos dejaron de escribirse como agrupamientos complejos para dar forma a notas separadas.

El primer paso hacia el definitivo pentagrama fue definir una de las líneas horizontales, pintada de rojo, como Fa 1, colocando a su lado una letra 'f' en una determinada caligrafía (que hoy en día sólo se emplea para indicar la clave de Fa). En segundo lugar, se definió una línea coloreada de amarillo como Do 2. Es de suponer que Fa y Do fueran elegidas como puntos de referencia fijos porque eran las únicas dos notas separadas sólo por un semitono de su precedente. Entre las líneas de esas dos notas se representaban los sonidos existentes entre ellas, así como por encima y debajo de ellas se hacía con las que estaban respectivamente más arriba y más abajo en altura de lo que estaban aquellas [12] .

V)Guido D'Arezzo. El sistema definitivo

Entre los años 1025 y 1033 se data la obra *Micrologus de Musica*, escrita por un monje benedictino llamado Guido D'Arezzo. Esta obra fue uno de los tratados sobre música con mayor difusión en la Edad Media junto con las obras de Boecio, aunque si se comparan desde la perspectiva actual las aportaciones de los escritos de Guido fueron, si no más importantes, sin duda más definitivas. El autor parte de la concepción de Hucbald sobre los instrumentos de seis cuerdas, con intervalos de tono (T) y semitono (S); y él concebía tres posiciones: el *hexachordum naturale* -con los sonidos equivalentes a Do 2, Re 2, Mi 2, Fa 2, Sol 2, La 2-, el *hexachordum durum* -con Sol 1, La 1, Si 1, Do 2, Re 2, Mi 2- y el *hexachordum molle* -con Fa 2, Sol 2, La 2, Sib 2, Do 3, Re 3- [1].

Guido D'Arezzo fue conocido tanto por su capacidad musical como por su inquietud pedagógica, y fue esto último lo que le llevo a idear un sistema mejor con el que enseñar música a sus alumnos de la escuela coral de la catedral de Arezzo. Durante su estancia en ella se percató de la dificultad de los cantantes para recordar los cantos gregorianos, e inventó un método para enseñar a los cantantes a aprenderlos en poco tiempo. Así fue como desarrolló nuevas técnicas, incluyendo el tetragrama (pauta musical de cuatro líneas), precursor del pentagrama, y la escala diatónica. Empleó como apoyo algunas técnicas mnemotécnicas y quironómicas; y este método pronto se hizo famoso en todo el norte de Italia.

Guido D'Arezzo es también el responsable de los nombres de las notas musicales. En la Edad Media, las notas se denominaban por medio de las primeras letras del alfabeto: A, B, C, D, E, F, G (comenzando por la actual nota La). Los

nuevos nombres tienen su origen en un himno a San Juan el Bautista, conocido como *Ut queant laxis*, que en aquella época era bastante popular, atribuido a Pablo el Diácono, y que Guido utilizó para sustituir la letra en una melodía que halló en una oda del IV libro de Horacio, de contenido considerablemente más profano. De este modo compuso un acompañamiento en el que cada línea comenzaba con una nota más aguda en el hexacordo que la línea anterior y denominó a las notas con la primera sílaba de cada verso:

« ***Ut*** *queant laxis,*
Re*sonare fibbris,*
Mi*ra gestorum,*
Fa*muli tuorum,*
Sol*ve polluti,*
La*bii reatum,*
S*ancte* ***I****oannes.* »

Guido de Arezzo denominó a este sistema de entonación solmisación (en latín, solmisatio), y más tarde se le denominó solfeo. Posteriormente, en el siglo XVII, Giovanni Battista Doni sustituyó la nota Ut por Do, pues esta sílaba, por terminar en vocal, se adaptaba mejor al canto. También mucho más tarde, a finales del siglo XVI, fue introducida por Anselmo de Flandes la séptima nota, que recibió el nombre de SI (de Sancte Ioannes). Los países donde no llegaron los músicos latinos siguieron con el antiguo sistema de las letras del alfabeto, tal es el caso de los países anglosajones, Alemania, los países escandinavos, etc.

Si bien la nomenclatura de Guido se encuentra extendida en términos geográficos sólo de forma parcial, todo lo que él aportó en relación a la notación es base de la empleada en la actualidad y aceptada mundialmente. Fue este benedictino quien perfeccionó la escritura musical con la implementación definitiva de líneas horizontales que fijaron alturas de sonido, y acabando con la notación neumática. Finalmente, después de ensayar varios sistemas de líneas horizontales, se fue imponiendo en los siglos sucesivos el pentagrama (cinco líneas).

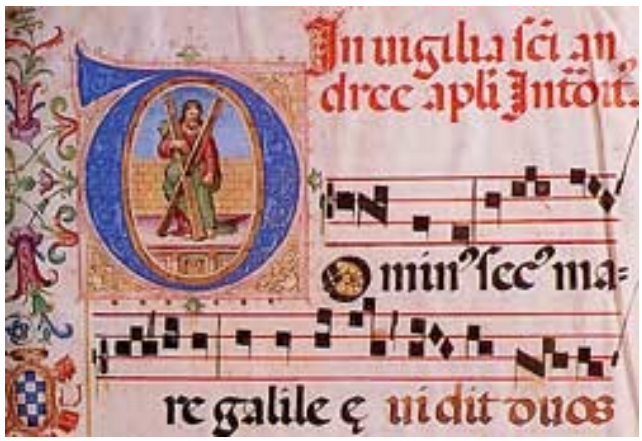


Figura 8: Ejemplo de notación en tetragrama.

VI)Evolución posterior

La escala pitagórica, ajustada a lo necesario para la música predominante hasta el final de la Edad Media, fue sustituida por la escala temperada utilizada en la actualidad, que se desarrolló para resolver problemas de afinación y llevó a una música en la que se podía modular (cambiar) de una tonalidad a otra sin tener que cambiar la afinación de los instrumentos. El temperamento es la forma musical de mantener series dentro de un espacio definido. La transición de la afinación pitagórica a la temperada tomó siglos, y ocurrió de manera paralela al cambio en la relación entre música y matemáticas.

En el siglo XII, compositores y ejecutantes empezaron a separarse de la tradición pitagórica creando nuevos estilos y tipos de música. Se creó una nueva división de las ciencias, llamada escolástica divina, que no incluía específicamente a la música. El canto monódico gregoriano (una sola voz ejecutada al unísono) poco a poco fue evolucionando en música polifónica con diferentes instrumentos y voces. La ejecución de composiciones más complejas llevaba a experimentar con afinaciones alternativas y temperamentos. Los experimentos de afinación resultaron en una variación de la afinación pitagórica llamada afinación justa.

Las nuevas afinaciones seguían utilizando las matemáticas para calcular los intervalos, pero no necesariamente seguían los principios pitagóricos. Ahora eran utilizadas de una forma práctica y no como un fin. Este cambio de actitud causó desacuerdo entre los matemáticos, quienes querían una adherencia estricta a sus fórmulas, y los músicos, que buscaban reglas fáciles de aplicar. De hecho los músicos empezaron a basarse más en su oído y menos en el monocordio [8] .

El temperamento no se popularizó sino hasta 1630, cuando el padre Mersenne (el mismo que estableció las relaciones de frecuencia para las cuerdas) formuló las reglas para afinar, usadas todavía hoy.

En el siglo XVIII, músicos como Johann Sebastian Bach (1685-1750), empezaron a afinar sus instrumentos usando el temperamento, es decir una escala en la que los doce sonidos fueran afinados sin diferencia entre un Fa sostenido y un Sol bemol. La complejidad de rango y modulaciones lo necesitaban. Fue así como Bach compuso *El clavecín bien temperado*, que consiste en 24 piezas en las doce tonalidades, usando el modo mayor y menor de cada una de ellas, demostrando de esta manera las posibilidades de modulación creadas por una afinación igual.

Aunque la música ya no es una disciplina estrictamente matemática, las matemáticas son inherentes a la música y continuarán influyendo en la evolución de la teoría musical [18] .

2.2.Definiciones básicas

En el siguiente apartado se tratan aspectos musicales básicos, que no obstante resulta conveniente definir dada la naturaleza del sistema a desarrollar. Como suele ocurrir con todas las artes, la definición de la música es compleja y abarca contenidos heterogéneos. Sin embargo, casi todas sus manifestaciones consisten en la producción de sonidos que varían de altura (tono) a lo largo del tiempo, o bien que ocurren en un determinado instante siguiendo un patrón de ritmo, y usualmente ambos a la vez.

El sonido tiene varias cualidades que se definen desde el punto de vista musical: la duración, la intensidad, el timbre y la altura [14].

2.2.1. Duración

La **duración**, como su propio nombre indica, es el tiempo que transcurre mientras permanece un determinado sonido, en concreto una nota o un silencio musical. La duración en la música se establece siempre desde un punto de vista proporcional, mediante las llamadas figuras musicales. Las figuras musicales son una representación gráfica de la duración de un sonido o silencio.

La figura que representa la duración mayor es la **redonda**, y a partir de ella se pueden definir la **blanca**, **negra**, **corchea**, **semicorchea**, **fusa** y **semifusa**, cada una de ellas con una duración que es la mitad de la de su figura precedente; y lo mismo se aplica a la duración de sus silencios respectivos. Existen otras figuras para representar duraciones todavía más breves que una semifusa, si bien, no se utilizan casi nunca. De igual manera también existen figuras de mayor duración que la redonda que se suelen considerar como prescindibles, por ello, aunque se usaron en la antigüedad, actualmente están en desuso.

Redonda	Blanca	Negra	Corchea	Semicorchea	Fusa	Semifusa
						

Figura 9: Figuras musicales

La relación numérica de proporción entre figuras es fundamental, a partir de ella se puede definir la figura redonda con valor 1, la blanca con valor ½, la negra con valor ¼, y así sucesivamente. De hecho, en la nomenclatura americana, para nombrar las figuras musicales se habla de nota entera (la redonda), nota media (la blanca), etc., hasta llegar a la nota sesentaicuatroava, que es la semifusa. Este fraccionamiento será decisivo para conceptos posteriores como el compás.

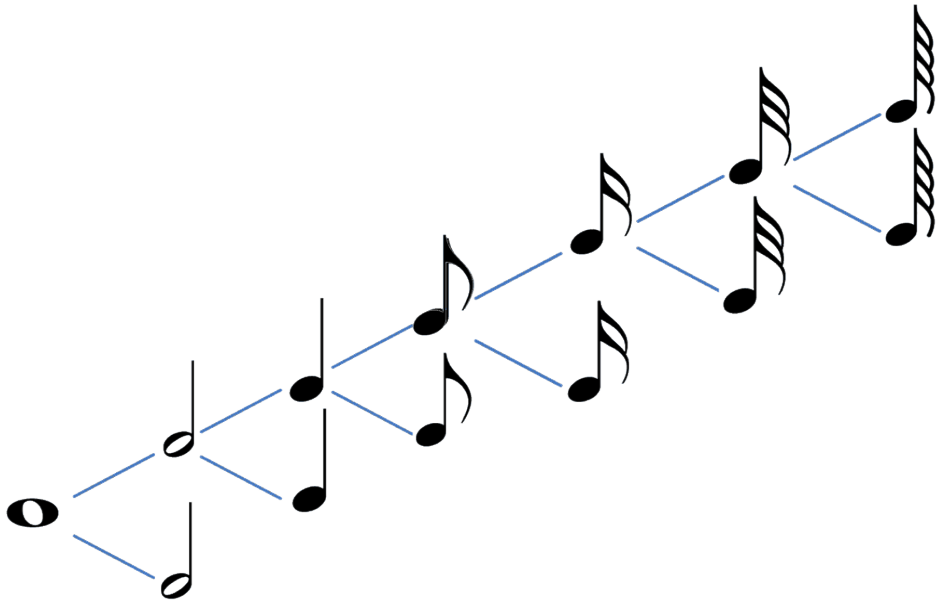


Figura 10: Relación de las figuras musicales.

Tabla 1: Descomposición de cada figura en las otras figuras.

	𝅝	𝅗𝅥	♩	♪	𝅘𝅥𝅯	𝅘𝅥𝅰	𝅘𝅥𝅱
𝅝		2	4	8	16	32	64
𝅗𝅥	0.5		2	4	8	16	32
♩	0.25	0.5		2	4	8	16
♪	0.13	0.25	0.5		2	4	8
𝅘𝅥𝅯	0.06	0.13	0.25	0.5		2	4
𝅘𝅥𝅰	0.03	0.06	0.13	0.25	0.5		2
𝅘𝅥𝅱	0.02	0.03	0.06	0.13	0.25	0.5	

Además de las figuras para representar la duración de los sonidos musicales, también existen las figuras que indican la ausencia de ellos, las pausas o **silencios**. La exactitud en la representación, así como lo es para su posterior interpretación, es tan importante como la de las figuras sonoras. Se los considera en ocasiones como un sonido de valor negativo, aunque en su sentido físico estricto sólo se podría hablar de sonidos de valor nulo o de valor no audible. Se definen silencios de duración idéntica a la de las figuras antes descritas, de tal forma que la ausencia de sonido más prolongada corresponde a la duración de una redonda y la menor a una semifusa. No obstante, con ellos sucede una particularidad que no se halla en las figuras sonoras: por su naturaleza, los silencios que aparecen seguidos suman su duración como si fueran uno sólo, algo que no es aplicable a las figuras sonoras a no ser que se indique explícitamente.

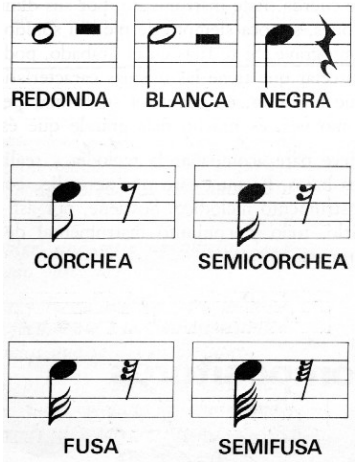


Figura 11: Figuras musicales y sus silencios correspondientes.

2.2.1.1.*Modificadores de duración*

Existen una serie de símbolos que pueden acompañar a las figuras musicales y sirven para modificar su duración. El más importante de ellos es el **puntillo**, que agregado a cualquiera de las figuras añade la mitad de su duración; de tal forma, una negra con puntillo vale 1'5 veces su duración, o lo que es lo mismo tiene la duración de tres corcheas, y de igual manera para el resto de figuras.

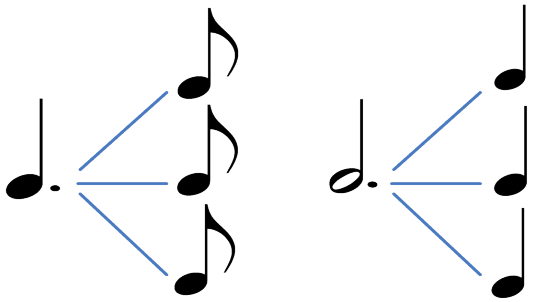


Figura 12: Ejemplos de equivalencias de figuras con puntillo.

Tal y como se expuso con anterioridad, los silencios por naturaleza suman sus duraciones cuando aparecen seguidos; sin embargo dos notas que se encuentren adjuntas, aunque produzcan idénticos sonidos, constituyen dos golpes sonoros diferenciados. Tan sólo existe un modificador que puede alterar esta regla: la **ligadura**. Cuando las notas aparecen ligadas (se representa gráficamente con una línea curva), se interpreta un único sonido cuya duración es suma de todas las de las notas ligadas.

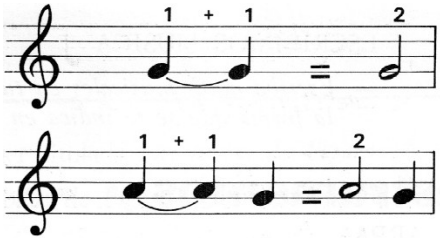


Figura 13: Modificaciones de duración resultantes de ligar notas.

También existen otra clase de alteraciones que provocan que una nota tenga una duración proporcional menor que la indicada por su figura. La más empleada es el **tresillo**, se utiliza con grupos de tres notas de igual duración y permite interpretarlas en el tiempo que se deberían tocar solo dos, la duración de cada una de las tres notas individuales se reduce para que así sea. Se denotan generalmente colocando una línea sobre las figuras con el número 3; es la subdivisión más usada, pero también existen otras como el seisillo y el quintillo.

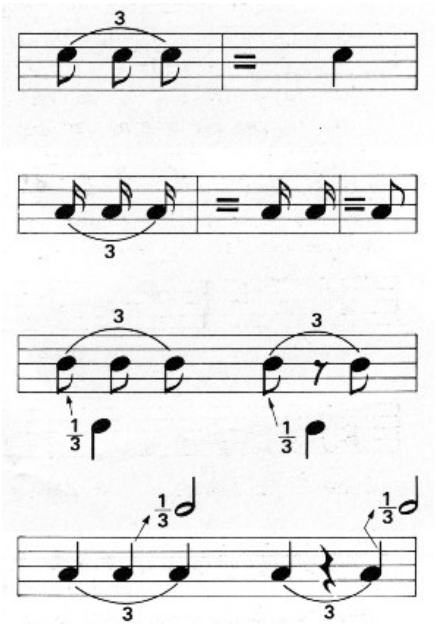


Figura 14: Ejemplos de tresillo.

Se pueden encontrar muchos otros recursos empleados en música que afectan a la duración de los sonidos individuales, tan sólo se recogen aquí los más significativos. A modo de ejemplo, se puede citar el "*Staccato*", un recurso estilístico que para resaltar las notas individuales acorta su duración, a fin de insertar una brevísima pausa anterior y posterior.

2.2.1.2.*Compás*

Las composiciones musicales organizan su duración en torno a compases. El tiempo del compás regula cuántas unidades de tiempo deben existir en cada compás. En otras palabras, cuántas figuras de un determinado valor caben en cada compás.

Por definición, cada compás se subdivide en fracciones principales que reciben el nombre de tiempos. Y estos tiempos, a su vez, se dividen en otras secundarias llamadas partes del compás. Las partes indicarán las notas que entran en cada tiempo.

Tabla 2: División temporal en compases.

Composición																							
Compás 1							**Compás 2**							…	**Compás *n***								
1er Tiempo			**2º Tiempo**			…	**1er Tiempo**			**2º Tiempo**			…	…	**1er Tiempo**			**2º Tiempo**			…	
Parte	**Parte**	…	**Parte**	**Parte**	…	…	**Parte**	**Parte**	…	**Parte**	**Parte**	…	…	…	**Parte**	**Parte**	…	**Parte**	**Parte**	…	…		

El compás de una obra musical se indica mediante una fracción con dos cifras, la fracción de compás, que representa el tiempo de la partitura. El numerador indica los tiempos que tiene el compás (en un 2/4, dos unidades o dos tiempos); el denominador, la figura que tiene cabida en cada unidad o tiempo del compás, lo que se llama unidad de parte del compás. El denominador que indica duración de una redonda es 1; el de la blanca 2; 4 el de la negra; 8 la corchea; 16 la semicorchea. O dicho de otra manera, el de la redonda es 1; el de la blanca 1/2; 1/4 el de la negra; 1/8 la corchea; 1/16 la semicorchea, ya que la duración en tiempo de cada figura es la mitad de la anterior.

Se pueden resumir de la siguiente forma:

$$\left.\begin{matrix} x \\ y \end{matrix}\right\}$$ Indica que en el compás caben 'x' figuras de valor 'y'.

De esta forma, el compás en 2/4 tiene, por tanto, dos unidades de tiempo y cada una de ellas del valor de una negra. Como cada negra es equivalente a dos corcheas, también puede albergar cuatro corcheas, u ocho semicorcheas. Una negra ocupa el valor de cada unidad de tiempo del compás.

Los compases, según la cantidad de partes de las que constan, se pueden clasificar en binarios, ternarios o cuaternarios.

Los **compases binarios** se dividen en dos tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Por ejemplo, entre los compases más habituales, el 2/4 y 6/8 son tiempos binarios.

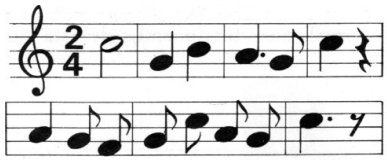


Figura 15: Ejemplo de compás de dos por cuatro -binario con subdivisión binaria-.

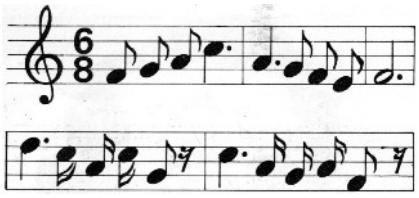


Figura 16: Ejemplo de compás de seis por ocho -binario con subdivisión ternaria-.

El compás de 2/4 es un compás binario, en cada uno de los dos tiempos que lo forman entra una negra y la figura que llena el compás es la blanca.

Los **compases ternarios** se dividen en tres tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Dentro de los tiempos ternarios están por ejemplo el 3/4 y el 9/8.

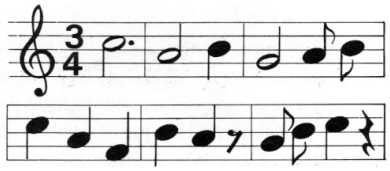


Figura 17: Ejemplo de compás de tres por cuatro –ternario-.

El compás de 3/4 es un compás ternario, en cada uno de sus tres tiempos entra una negra y la figura que llena todo el compás es la blanca con puntillo.

Los **compases cuaternarios** se dividen en cuatro tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Por ejemplo los compases de 4/4 y 12/8 son tiempos cuaternarios.

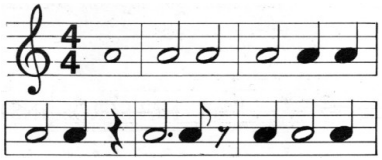


Figura 18: Ejemplo de compás de cuatro por cuatro -cuaternario-.

El compás de 4/4 es un compás cuaternario, en cada uno de los cuatro tiempos entra una negra y la figura que llena todo el compás es la redonda.

2.2.1.3.*Tempo*

Sin alterar la proporción existente entre las figuras, una misma composición musical se puede interpretar de forma diferente de manera que la duración global sea mayor o menor. Tal variación se debe a la velocidad con la que se interpreta la obra, definida por el **movimiento** o **tempo**.

Se trata de una medida cualitativa, pero se puede explicitar una velocidad más cuantitativa especificando la cantidad de pulsaciones por unidad de tiempo. Usualmente, para ajustar la medición del metrónomo y como medida equivalente a la anterior, se indica cuántas figuras de un determinado valor deben tocarse en un

minuto; en este caso, casi siempre se hace referencia a la figura negra, así, por ejemplo, se incluiría la indicación  para explicitar que en el intervalo de un minuto se da cabida a 60 negras.

Tabla 3: Velocidad de la interpretación: movimientos principales.

Nombre	Definición cualitativa	Medida cuantitativa [*pulsaciones por minuto*]
Largo	Muy lento	20 ppm
Adagio	Lento	66 a 76 ppm
Andante	Andando	76 a 108 ppm
Moderato	Moderado	108 a 120 ppm
Allegro	Ágil, alegre	120 a 168 ppm
Presto	Muy rápido	168 a 200 ppm

2.2.2. Intensidad

La **intensidad** musical es proporcional al parámetro físico de idéntico nombre, ya que depende de la amplitud de las ondas sonoras que produce una nota. La **dinámica** musical hace referencia a la diferente intensidad con la que se interpreta una composición pero, al igual que ocurría con el tempo, se define en términos cualitativos, en este caso para clasificar a los sonidos como fuertes o débiles. De tal modo, las diferentes posibilidades dinámicas se ejecutan con una intensidad que no esta definida matemáticamente, y que suele variar ligeramente dependiendo de la consideración del intérprete y del estilo o periodo histórico.

Tabla 4: Descripción de dinámicas

Símbolo	Nombre	Ejecución
pp	*pianíssimo*	Muy suave
p	*piano*	Suave
mp	*mezzo piano*	No tan suave
mf	*mezzo forte*	Poco fuerte
f	*forte*	Fuerte
ff	*fortísimo*	Muy fuerte

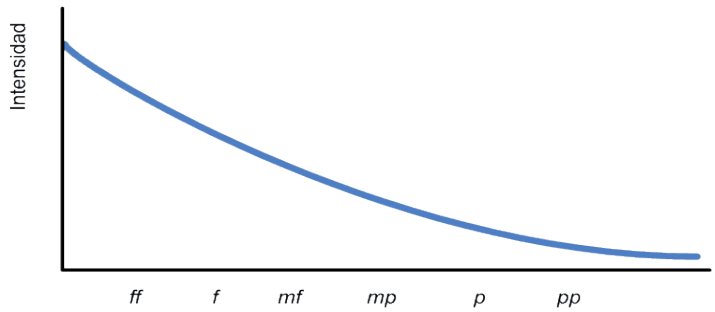


Figura 19: Relación cualitativa Intensidad - Dinámica

También existen patrones que cambian la intensidad de una forma gradual a lo largo de una sucesión de sonidos. Son el ***crescendo*** y ***diminuendo***, que indican gráficamente un punto de mayor intensidad y otro de menor; cuando es *crescendo* se parte del menos intenso para ir incrementando progresivamente la intensidad, hasta llegar al marcado como más intenso; cuando se trata de un *dimiuendo* el proceso es el contrario, se disminuye progresivamente la intensidad.

2.2.3. Altura

2.2.3.1. *La representación en pentagrama*

Se denomina **pentagrama** o **pauta musical** al conjunto que constituyen cinco líneas horizontales y equidistantes, con sus correspondientes cuatro espacios entre líneas, sobre el cual se representa la música, llegando a constituir una partitura. El pentagrama ha facilitado la escritura musical durante siglos hasta hacer prácticamente imprescindible el papel pautado con las cinco líneas. Sobre el pentagrama se representan las notas que expresan los sonidos musicales y determinan su entonación y su duración.

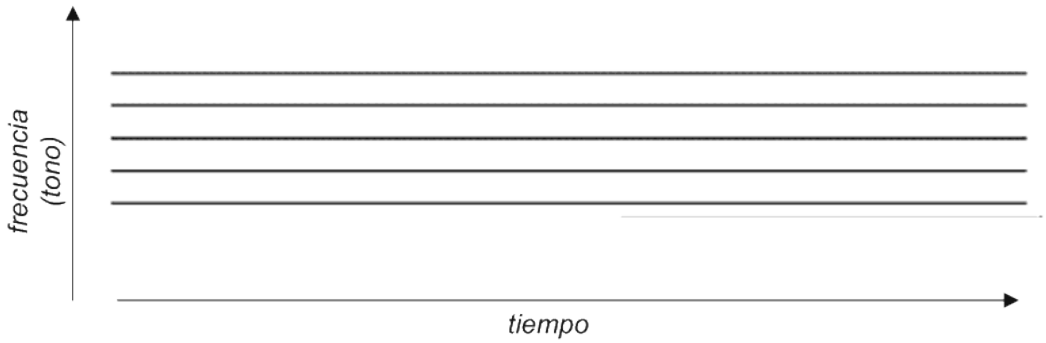


Figura 20: Pentagrama y significado físico.

Las claves musicales que acompañan a un pentagrama permiten asociar cada línea o espacio a un sonido concreto, determinado por una **altura** o **tono**. Las claves siempre llevan el nombre de una de las tres notas Sol, Fa y Do y se sitúan señalando una determinada línea del pentagrama, en la cual se encuentra la nota que lleva su nombre. A partir de ella, se ubican las notas anteriores y posteriores a la misma.

La clave de Sol en segunda línea es la más utilizada por dar cabida en sus líneas a las notas musicales más habituales, a los sonidos centrales del piano, a los principales de muchos instrumentos y a gran parte de los existentes en melodías vocales. También se pueden emplear claves de Sol en otras líneas, pero es un hecho bastante infrecuente.

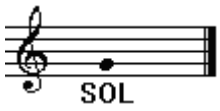


Figura 21: Clave de Sol en segunda.

Las claves de Fa se usan con bastante frecuencia para recoger sonidos graves de voces e instrumentos, así como para los acompañamientos de piano. Lo más habitual es encontrarla señalando la cuarta línea, pero también se utiliza en la tercera.

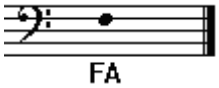


Figura 22.a: Clave de Fa en cuarta.

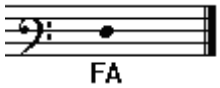


Figura 22.b: Clave de Fa en tercera.

Las claves de Do se emplean en ciertas ocasiones para sonidos medios, principalmente en tercera y cuarta línea. La situada en tercera también se llama de contralto, ya que las partes de canto para esta voz se representaban en la antigüedad en esta clave, y en la actualidad se emplea casi en exclusiva para las violas. La clave de Do en cuarta recibe el nombre de clave tenor, también por el uso que de ella se hacía en el canto en tiempos pasados, y actualmente sólo se usa para el fagot y para representar algunas notas agudas del violonchelo y del trombón. Las claves en primera y segunda prácticamente han desaparecido en la notación actual.

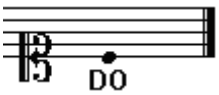


Figura 23.a: Clave de Do en primera.

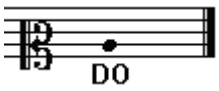


Figura 23.b: Clave de Do en segunda.

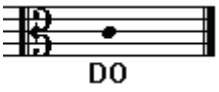


Figura 23.c: Clave de Do en tercera.

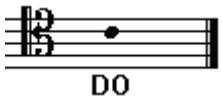


Figura 23.d: Clave de Do en cuarta.

2.2.3.2. *Definición de altura o tono*

La **altura** o **tono** depende directamente de la frecuencia fundamental que adopta la vibración generadora del sonido musical. Se diferencian alturas o tonos agudos cuando se producen por frecuencias altas y alturas graves cuando se producen por frecuencias más bajas.

La descomposición más pequeña de los sonidos audibles provocados por un instrumento musical o voz la constituyen las notas, unidad básica de interpretación. Las notas se organizan de acuerdo a la frecuencia fundamental que las define, que se supone invariante, en octavas de frecuencia. Dentro de cada octava se encuentran siete notas: Do, Re, Mi, Fa, Sol, La, Si. Éstas se encuentran separadas entre sí por un intervalo que se define como un tono, a excepción de la distancia existente entre el Mi y el Fa de una misma octava y entre el Si de una octava y el Do de la posterior, separadas por un intervalo de semitono.

Un tono es una diferencia de frecuencia entre sonidos que responde a la razón entre las frecuencias respectivas, mientras que un semitono está definido como una relación de $\sqrt[12]{2}$ entre ellas. Tales relaciones se cumplen para todo instrumento que emplee la afinación temperada, la cual hace posible que, sin ninguna desviación, una octava se divida exactamente en seis intervalos de tono o doce de semitono.

Esta división es, con carácter general, válida internacionalmente, aunque en sentido estricto proviene de las tradiciones musicales de Occidente, herederas de la estructura TTSTTTS (donde 'T' es intervalo de tono y 'S' de semitono) del canto gregoriano y otros antecedentes. Ciertamente, se observa una mayor diversidad en la nomenclatura de las notas, los nombres de 'Do', 'Re', 'Mi', etc. se emplean en el entorno mediterráneo y latino, en países como España, Francia, Italia o Turquía, entre otros muchos. Por otro lado, la que se conoce como nomenclatura inglesa (o

anglosajona) es en realidad anterior a ésta y cuenta con bastante expansión, emplea las letras del alfabeto al igual que ya se hacía en la baja Edad Media; la 'A' designa a la nota 'La' latina, y continua la serie alfabética hasta llegar a la nota 'G' que es equivalente a 'Sol', aunque estrictamente, debido a que se preserva la misma estructura en octavas, se debe decir que se comienza en 'C' (Do) y se acaba en 'B' (Si). Es habitual tener que trabajar con las dos nomenclaturas cuando se trabaja con materiales de procedencias diversas.

Para representar las notas musicales se recurre al pentagrama, donde quedan reflejados todos los aspectos de la interpretación: las duraciones, la intensidad y, especialmente, los tonos o alturas de las mismas que de otra manera no se podrían representar. Así, se distingue un determinado tono (Do, Re, etc.) según su posición en el pentagrama y la clave que afecte a éste.

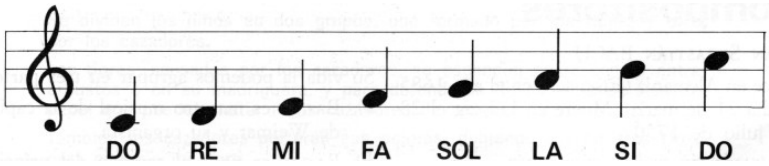


Figura 24: Representación de las notas centrales en clave de Sol

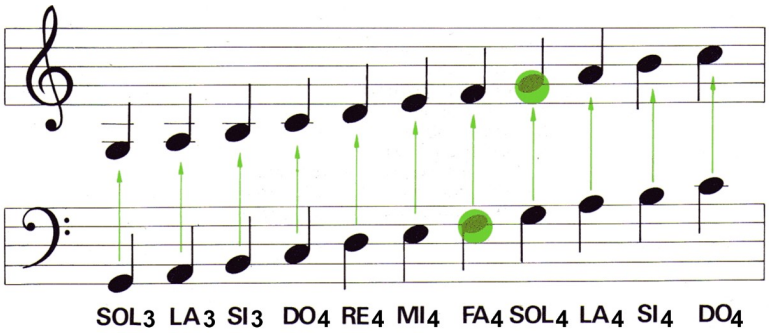


Figura 25: Posición de las notas y correspondencias en claves de Sol y Fa. El número que acompaña al nombre de la nota hace referencia a la octava a la que pertenece.

2.2.3.3. *Alteraciones tonales*

Se definen varias alteraciones que afectan al tono de las notas en la forma que se ha definido antes. Se representan mediante signos que se colocan delante de las notas e implican esta alteración. En concreto son cinco: sostenido, doble sostenido, bemol, doble bemol y becuadro.

El **sostenido** eleva un semitono el sonido de la nota natural, y el **doble sostenido** la eleva dos semitonos. El más habitual es el sostenido simple y el doble se encuentra en contadas ocasiones.

El **bemol** decrementa el sonido de la nota natural en un semitono, y el **doble bemol** dos semitonos. También, como en el caso anterior, es bastante más frecuente hallar el bemol sencillo que el doble.

El **becuadro** anula el efecto de todas las alteraciones anteriores, en situaciones en las que de otra forma seguirían vigentes (detalladas más adelante).

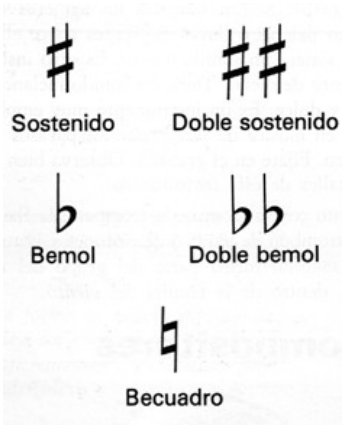


Figura 26: Alteraciones

Por la forma en que están definidas las octavas musicales, expuesta con anterioridad, el sostenido de una nota es equivalente al bemol de la siguiente (ej.: un Sol sostenido tiene el mismo valor en frecuencia que un La bemol). En el caso de Mi y Fa, se observa que un Mi sostenido es equivalente a un Fa, y un Fa bemol equivalente a un Mi (ya que la distancia entre ellos es sólo de un semitono), y el caso análogo sucede con un Si y un Do de octavas sucesivas.

Tabla 5: Frecuencias fundamentales de las notas musicales

NOTA	Frecuencia [Hz]					
	Octava 2	Octava 3	Octava 4 (central)	Octava 5	Octava 6	Octava 7
DO	65,41	130,81	261,63	523,25	1046,50	2093,00
DO#	69,30	138,59	277,18	554,36	1108,73	2217,46
RE	73,42	146,83	293,66	587,33	1174,66	2349,32
RE#	77,78	155,56	311,13	622,25	1244,51	2489,02
MI	82,41	164,81	329,63	659,26	1318,51	2637,02
FA	87,31	174,61	349,23	698,45	1396,91	2793,83
FA#	92,50	185,00	369,99	739,99	1479,98	2959,96
SOL	98,00	196,00	392,00	783,99	1567,98	3135,96
SOL#	103,83	207,65	415,30	830,61	1661,22	3322,44
LA	110,00	220,00	**440,00**	880,00	1760,00	3520,00
LA#	116,54	233,08	466,16	932,33	1864,66	3729,31
SI	123,47	246,94	493,88	987,77	1975,53	3951,07

Las alteraciones pueden definirse para un compás concreto o para toda una obra musical. Las llamadas alteraciones accidentales se señalan junto a la nota a la que afectan, y por definición modifican a todas las notas que sean iguales a ésta en el compás, salvo que se anule su efecto mediante un becuadro. Las alteraciones de

armadura son aquellas que constan al principio del pentagrama con objeto de afectar a toda la obra; se indican mediante los símbolos de bemol o sostenido después del símbolo de la clave y antes de la indicación del compás. Rigen toda la composición mientras otras alteraciones no las anulen o se cambie la armadura por otra. El orden en que se añaden sostenidos y bemoles es siempre el mismo, ya que están definidos por un concepto llamado tonalidad, y acumulativo, cuando se añade una alteración para una nota no se quita la anterior. El orden en que se añaden sostenidos en la armadura es siempre: Fa, Do, Sol, Re, La, Mi, Si; y para los bemoles el inverso: Si, Mi, La, Re, Sol, Do, Fa.

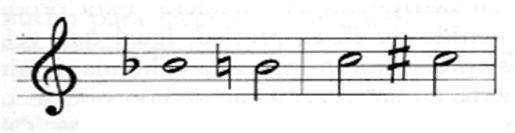


Figura 27.a: Alteraciones accidentales, afectan a la nota a la que acompañan durante todo el compás

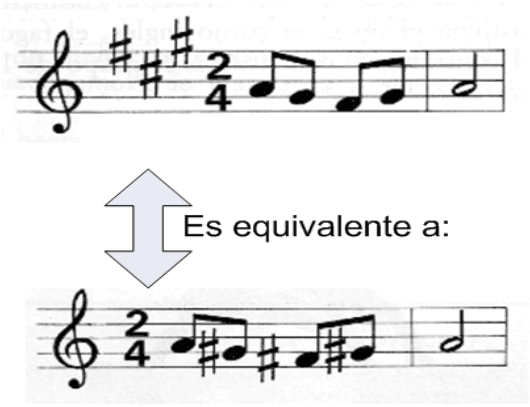


Figura 27.b: Alteraciones propias, constituyen la armadura y actúan a lo largo de toda la composición

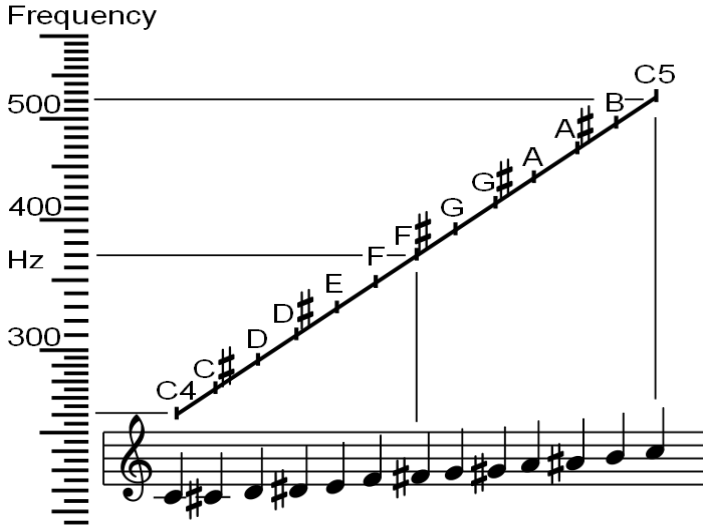


Figura 28: Escala cromática –intervalos de semitono- y frecuencias fundamentales. En esta ocasión se emplea la nomenclatura anglosajona.

2.2.4. Timbre

El **timbre** es una característica que identifica a cada voz o instrumento. Permite distinguir una determinada voz o instrumento con respecto a los demás. Tiene su razón física en las diferentes componentes frecuenciales que éstos generan a partir de una misma frecuencia fundamental (la nota musical interpretada). Como norma general, dicha diferencia estriba en las amplitudes para los sucesivos armónicos. Los materiales y los principios de funcionamiento de cada instrumento determinan esta peculiaridad.

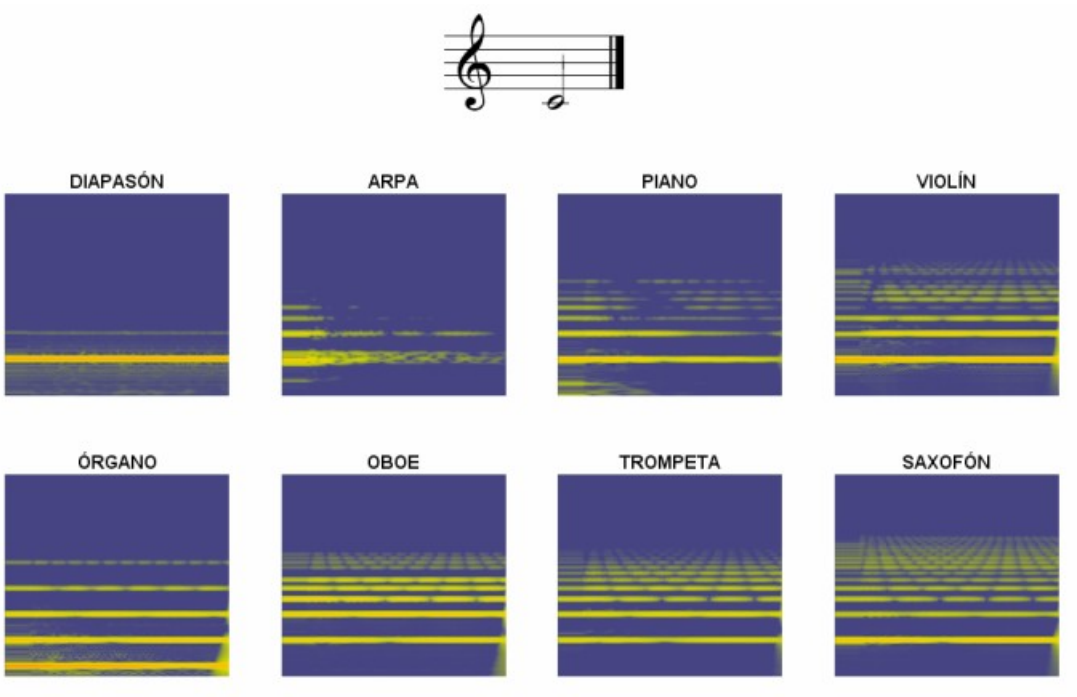


Figura 29: Respuesta en frecuencia de diversos instrumentos a la nota Do4 expresadas como espectrograma. De esta forma se resaltan los armónicos que determinan su timbre [37].

3.Estado del arte

En los sucesivos apartados de este capítulo se recoge una descripción de las tecnologías y desarrollos involucrados en este proyecto, así como los empleados en otros contextos con fines similares. En primer lugar se describirá MIDI, el formato de audio sintetizado más extendido en el ámbito musical. Posteriormente, se realizará una introducción acerca del tratamiento del sonido en Java, y más en concreto de la API jMusic enfocada a la música. En el apartado siguiente, se explicarán los principios aplicables para estructurar cualquier tipo de datos en XML. A continuación, se detallan los procedimientos de Recuperación de Información que ofrece la API Lucene para la indexación y búsqueda de datos. En último lugar, se expone el caso particular de Musipedia, un buscador de características similares al desarrollado en este proyecto, si bien se asentará sobre una estructura diferente.

3.1. Audio sintetizado MIDI

La evolución en los últimos años de los formatos de grabación de audio digital ha sido muy importante, sin embargo, las prestaciones que ofrecen sistemas de audio sintetizado, como MIDI principalmente, siguen siendo muy necesarias; tal y como lo han sido desde la aparición de los primeros sistemas informáticos con sonido. En primer lugar es necesario aclarar un hecho fundamental: el audio digital y el audio sintetizado tienen naturalezas muy diferentes entre sí.

El tratamiento de audio digital [19] implica un proceso mediante el cual una señal de audio -procedente de instrumentos musicales, voces o cualquier elemento sonoro- se muestrea de forma constante y se convierte a continuación en una serie de palabras en binario que representan el valor de la señal sonora. El archivo digital consistirá en una secuencia de datos de esta índole; y para ser reproducido requerirá un proceso inverso al anterior que tendrá lugar en un conversor digital/analógico, que volverá a transformar la información digital en una onda sonora. MIDI [40], sin embargo, maneja una información digital cuya función es controlar el proceso de generación del sonido. Los datos representan en este último caso lo que se define como eventos: un evento puede consistir tanto en la pulsación de una nota como en la variación de un ajuste o la conmutación de algún interruptor. Cuando se realiza una "grabación" musical mediante un secuenciador MIDI, lo que se almacena son datos que corresponden a eventos. La reproducción se podrá realizar mediante un dispositivo controlado por MIDI –ya sea un

instrumento musical específico o un ordenador personal- tras recibir estos datos, y serán los propios dispositivos los que generen un sonido, determinado por esta información pero no contenido en ella. La primera ventaja notable que se desprende de sus características es la reducción drástica de tamaños, ya que no se almacenan ondas completas sino lo que se puede considerar el equivalente a las clásicas notas musicales. La tasa binaria de MIDI es de 31.25 kbps [19] .

MIDI aparece a principios de la década de los ochenta, más como un convenio entre fabricantes que como un estándar al uso. De hecho, aún en la actualidad se arrastran pequeñas diferencias entre fabricantes, que no impiden, sin embargo, la interoperabilidad prácticamente completa. MIDI en realidad se define de partida como una interfaz para intercambiar mensajes entre instrumentos musicales, de ahí sus siglas: *Musical Instrument Digital Interface*. Posteriormente surgen los archivos MIDI, que se valen de la misma codificación para almacenar los sonidos musicales.

La estructura básica de mensajes y ficheros consiste en series de tramas de 24 bits. El primer byte de la trama, identificado con el primer bit a 1, codifica el estado en una combinación de 7 bits; y los dos siguientes bytes, con el primer bit a 0, dan cabida a datos de 7 bits.

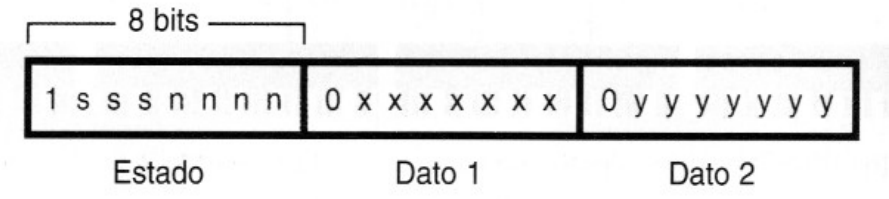


Figura 30: Trama binaria de datos MIDI.

En el byte de estado, los tres primeros bits de información (bits 's') indican el estado propiamente dicho, y los cuatro siguientes (bits 'n') señalan a cuál de los 16 canales posibles hace referencia. La excepción a esta regla son los mensajes de sistema, que presentan los 4 primeros bits a uno, y que no hacen referencia a ningún canal concreto. Los bytes de datos se interpretan de acuerdo a una norma fija establecida en la especificación MIDI [40].

Tabla 6: Ejemplos de codificación de algunos mensajes MIDI [40].

Mensaje	Estado [hexadecimal]	Dato 1	Dato 2
Note off	&8n	Núm. de nota	Velocidad
Note on	&9n	Núm. de nota	Velocidad
Polyphonic aftertouch	&An	Núm. de nota	Presión
Control change	&Bn	Núm. de controlador	Datos
Program change	&Cn	Núm. de programa	-
Channel aftertouch	&Dn	Presión	-
Pitch Wheel	&En	LSByte	MSByte
Mensajes de sistema exclusivo:			
System exclusive	&F0	ID de fabricante	Datos, (datos), (datos)
End of SysEx	&F7	-	
Mensajes comunes:			
Quarter trame	&F1	Datos	-
Song pointer	&F2	LSByte	MSByte
Song select	&F3	Núm. de canción	-
Tune request	&F6	-	-

En general, un fichero MIDI contiene datos que representan eventos sobre determinadas pistas de un secuenciador. Referencia mediante su código correspondiente a los instrumentos que pertenezcan a cada pista e incluye ciertas marcas temporales. Se pretende que estos archivos sean totalmente compatibles entre los diferentes sistemas, y que suenen de forma muy similar en todos los casos. Para ello se definió el llamado *general MIDI* que define el código exacto de cada instrumento. Los ficheros MIDI estándar pueden clasificarse en 3 tipos: el tipo 0 es el más sencillo y se utiliza para datos referentes a una única pista; el tipo 1 sirve para varias pistas sincronizadas entre sí verticalmente; y el tipo 2 contiene varias pistas sin relación temporal directa entre sí, que por lo tanto pueden ser asíncronas, y se suele usar para transferir ficheros con varias canciones secuenciadas, cada una de ellas de estructura multipista.

Tabla 7: Números de notas MIDI -valor introducido en el primer byte de dato para referenciarlas-.

Nota musical (Convención general)	Nota musical (Convención Yamaha)	Número de nota MIDI
C-1	C-2	0
C0	C-1	12
C1	C0	24
C2	C1	36
C3	C2	48
C4 (DO central)	C3 (DO central)	60
C5	C4	72
C6	C5	84
C7	C6	96
C8	C7	108
C9	C8	120
G9	G8	127

Tras proporcionar una sistematización para que cualquier plataforma sea capaz de interpretar el formato de MIDI, es necesario un mecanismo que permita sintetizar los sonidos, labor que corresponde al secuenciador MIDI. El funcionamiento de un secuenciador está relacionado estrechamente con las bases matemáticas de la música: las notas musicales consisten en ondas sonoras complejas con una frecuencia fundamental concreta. Para sintetizar audio en un secuenciador, los procedimientos primitivos se valían de una sinusoide de tono puro a la frecuencia determinada de la nota; evidentemente la música generada era bastante rudimentaria y antinatural. Por este motivo surgió la síntesis basada en *samples* o muestras de sonido reales para cada instrumento; su fundamento es sencillo: se parte del fragmento sonoro que corresponde a la interpretación de una determinada nota por un instrumento y se obtendrán a partir de él el resto de las notas, multiplicando sus componentes en frecuencia por la relación que corresponda –a razón de $\sqrt[12]{2}$ por cada intervalo de semitono- [19].

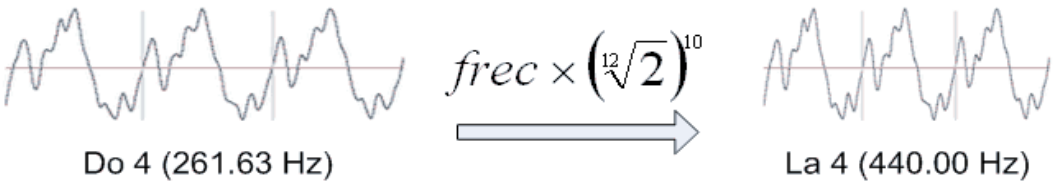


Figura 31: Ejemplo de síntesis de diferentes notas a partir de una misma muestra.

Las muestras que emplea el secuenciador se albergan en la llamada *wave table*, y usualmente almacenan varias muestras de notas por instrumento, en número tanto mayor cuanto más realista se pretenda que sea la síntesis. Estas muestras no sólo servirán para generar variaciones en frecuencia, también se empleará sobre ellas la amplificación o atenuación para dar lugar a diferentes intensidades.

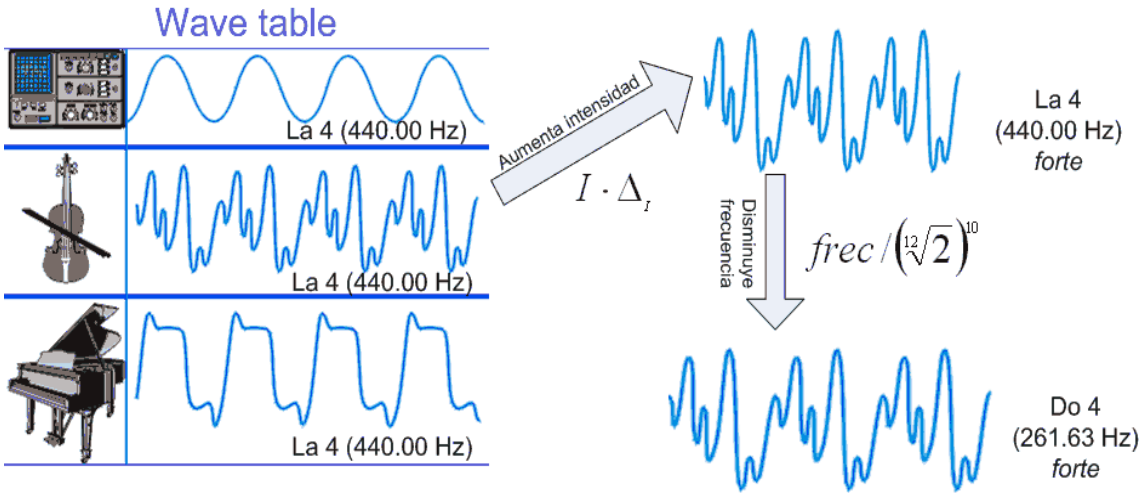


Figura 32: Ejemplo de síntesis musical a partir de la *wave table*.

3.2. El sonido en Java y JMusic

El tratamiento de fuentes de naturaleza multimedia, y sobre todo del sonido, de un modo especialmente adaptado a sus características propias, constituyó enseguida un objetivo en el entorno de desarrollo Java. Así, en la versión J2SE 1.3 [48], lanzada en mayo de 2000, se incluyeron los paquetes de ***javax.sound***, que permiten, entre otras posibilidades, leer y escribir ficheros MIDI, y controlar flujos de entrada y salida de audio.

De igual forma surgieron complementos como el *Java Media Framework* (**JMF**) [49], una API para la manipulación y procesamiento de medios en Java, que además permite la captación (tanto por medios locales, como a través de la red), almacenamiento, reproducción y difusión. La versión inicial, aún sólo con funciones de reproducción, vio la luz antes de que Java incorporara funciones básicas de sonido: JMF 1.0 apareció en 1997 desarrollado por Sun Microsystems, Silicon Graphics, e Intel. La versión JMF 2.0, que se hizo pública en 1999, fue desarrollada por Sun e IBM, incorporando gran parte de las funciones actuales.

Dentro de las funcionalidades que implementa JMF existe un especial interés por cumplir los requisitos temporales que requieren estos medios (principalmente audio y vídeo). Pretende ofrecer una herramienta a alto nivel que exija el conocimiento de un número reducido de clases y métodos para los usos habituales, relegando a un segundo plano las tareas necesarias a bajo nivel, que permanecen ocultas.

Prácticamente de manera simultánea a la aparición estas tecnologías, un grupo de investigadores (cuyo estudio se centraba en la tecnología musical) se planteó la necesidad de desarrollar sobre el lenguaje Java una serie de utilidades relacionadas con la composición musical; así nació **jMusic**. Con el paso del tiempo, el desarrollo que llevaron a cabo se sigue considerando de interés, ya que a pesar de existir funciones en el kit básico que leen y escriben MIDI u otros formatos sobre los que se pueden representar partituras, no permiten un nivel de abstracción tan próximo al lenguaje musical como el de éste. A continuación, se detallan las características de la librería jMusic.

3.2.1. jMusic

jMusic [43] es fruto del trabajo de investigación del departamento de música de la *Queensland University of Technology*, en Brisbane (Australia). Fue creada por Andrew Sorensen y Andrew Brown, y publicada oficialmente en 1998. Se trata de

una librería Java enfocada a las necesidades de los músicos, con la pretensión de ser suficientemente sencilla para los programadores noveles pero con la suficiente complejidad como para permitir realizar las tareas que la música requiere. A pesar de ser su objetivo principal servir de medio compositivo para músicos y no tanto para programadores, con el paso de los años, ha sido empleada en numerosas ocasiones como una API para el desarrollo de software musical, sobre todo para la implementación de instrumentos digitales.

Esta librería es Open Source, distribuida con licencia GNU General Public License. En sus principios básicos está ser extensible y, de hecho, se fomenta tanto su uso como su rediseño para la creación de aplicaciones. El paquete básico ha sido desarrollado y extendido como un trabajo colectivo, por personas interesadas en la música digital para sus propias creaciones y para otros creadores de música.

jMusic proporciona una estructura Java para la generación de música, construcción de sonidos de instrumentos, interpretaciones interactivas y análisis musical. Soporta compositores con su propia estructura musical de datos siempre que estén basados en eventos de nota/sonido (al estilo MIDI), y provee métodos para la organización, manipulación y análisis de los datos musicales. Es capaz de modelar partituras que puede retornar como archivos MIDI o como archivos de audio digital, para su almacenamiento y posterior procesamiento o reproducción en tiempo real. jMusic puede leer y escribir archivos MIDI, archivos de audio, archivos XML y su formato propio de archivos ".jm"; y permite soporte en tiempo real para JavaSound, QuickTime y MIDIShare.

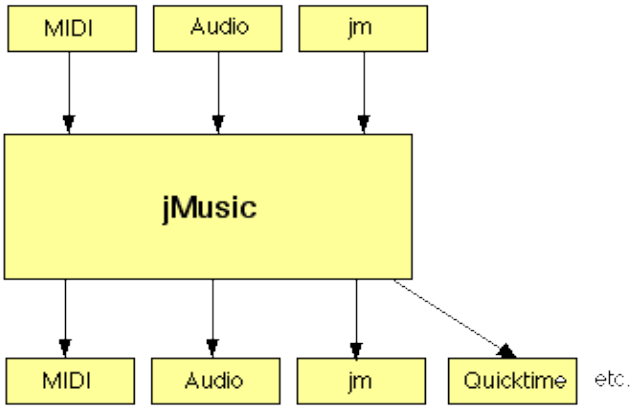


Figura 33: Posibles entradas y salidas de funciones jMusic.

jMusic también proporciona varias herramientas de visualización y audio como ayuda para ver y oír el estado actual de la composición. Gracias a que jMusic es un paquete Java, ésta se puede servir de los componentes jMusic para escribir aplicaciones.

Una de las principales potencialidades de jMusic consiste en que se construye sobre convenciones de la música tradicional occidental. Es posible almacenar la información musical siguiendo una jerarquía basada en las clásicas partituras.

La **nota** *(Note)* es la unidad más pequeña. Cada nota se irá almacenando junto con una serie de notas en la siguiente unidad superior, la **frase** *(Phrase).* Varias frases formarán una **parte** *(Part) y* varias partes una **partitura** (*Score*).

La API jMusic será importante en el desarrollo de este proyecto, debido a ello y por aportar mayor claridad, se incidirá más a fondo en su estructura en el capítulo de implementación.

3.3. Datos estructurados en XML

XML [51] es un lenguaje de marcado extensible (*eXtensible Markup Language*) diseñado para **almacenar** y **transportar** datos. Su fin primordial no es la visualización, a diferencia de otros lenguajes de marcado –como HTML.

El lenguaje XML describe una clase de objetos de datos llamados documentos XML y parcialmente describe el comportamiento de aplicaciones destinadas a procesarlos. XML es un perfil de aplicación, una simplificación, de SGML (*Standard Generalized Markup Language*) [10] . Por construcción, todo documento conforme con XML es conforme con SGML.

XML se constituyó en una recomendación oficial del consorcio de la *World Wide Web* (W3C) en febrero de 1998. Fue desarrollado por un Grupo de Trabajo de XML (originalmente conocido como el comité de revisión editorial de SGML) formado bajo el auspicio del W3C en 1996 [51]. Estaba presidido por Jon Bosak de Sun Microsystems con la participación activa de un Grupo Especial de Interés en XML también organizado por el consorcio. Dan Connolly sirvió como el contacto del grupo con la W3C.

Algunos de los objetivos de diseño para XML más relevantes son:

- XML debe ser utilizable directamente sobre internet.
- XML debe soportar una amplia variedad de aplicaciones.
- Debe ser fácil escribir programas que procesen documentos XML.

- El número de características opcionales en XML debe ser mantenido en un mínimo, idealmente cero.
- Los documentos XML deben ser legibles por un humano y claros (se diseñaron como documentos autodescriptivos).
- El diseño de XML debe ser confeccionable rápidamente
- El diseño de XML debe ser formal y conciso.
- Los documentos XML deben ser fáciles de crear.

Los documentos XML están compuestos por unidades de almacenamiento definidas por etiquetas, las cuales dan cabida a contenido en forma de datos procesados (*parsed*) o sin procesar. Las etiquetas tienen siempre formato textual limitado a caracteres ASCII y en el caso de los datos, para hacer referencia a algunos caracteres Unicode, se emplea un sistema de codificación conocido como entidad. No existen nombres de etiquetas ni datos predefinidos, son éstos aspectos que quedan totalmente a elección del usuario en cada implementación.

La estructura anterior obedece a dos tipos de reglas que se han de cumplir: comunes (para todos los documentos en XML) y específicas (de aplicación para un conjunto concreto). Para que un documento esté bien formado se han de cumplir las reglas comunes, que tienen que ver con que existan etiquetas de apertura y cierre situadas adecuadamente, así como otros aspectos relacionados con la sintaxis. Para que un documento sea válido se han de respetar las reglas específicas establecidas en un DTD o *XML Schema* asociado a éste, que será el encargado de definir tanto los nombres y estructura de las etiquetas como los contenidos válidos para su caso concreto.

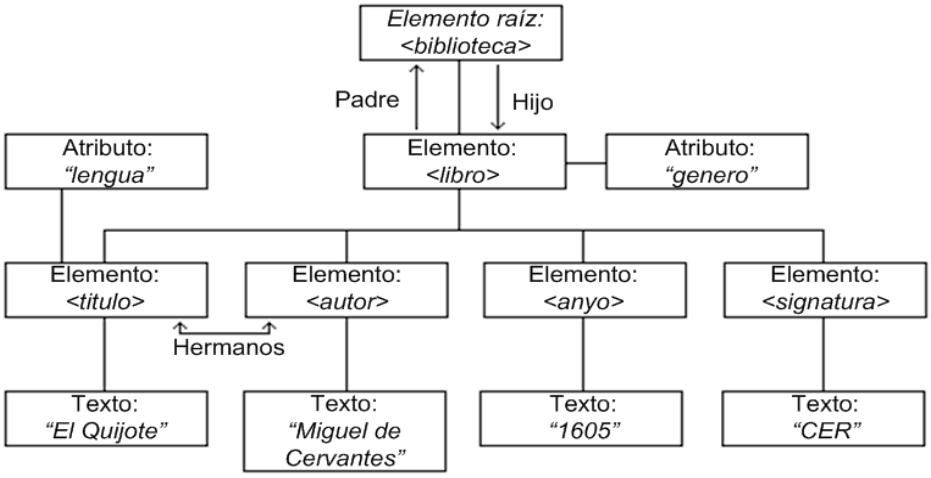


Figura 34: Ejemplo de diseño en XML.

Las unidades constructoras más importantes de XML son los elementos, formados por una etiqueta de comienzo, un contenido (que puede ser otro elemento o datos), y una etiqueta de fin. También puede existir algún dato que no se incluya en el contenido del elemento, sino que directamente se defina en la etiqueta de comienzo, éstos son los atributos. Todos los documentos XML requieren un elemento raíz, a partir del cual se incluirán los demás elementos en su interior; no existe ninguna limitación fijada sobre cuántos elementos se pueden incluir o anidar, solamente lo que en el DTD o *Schema* se defina.

A continuación se detalla un ejemplo concreto de documento XML creado según las directrices de un DTD o de su *Schema* equivalente. Como se puede observar, la sintaxis de un DTD (*Document Type Definition*) es específica, mientras que la de un *Schema* es en sí misma XML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Mensaje.dtd">
<mensaje>
<para>Charles</para>
<de>Robin</de>
<asunto>Recordatorio</asunto>
<contenido>No olvides el plan previsto para el fin de semana.</contenido>
</mensaje>
```

Figura 35: Ejemplo de documento XML.

```
<!DOCTYPE mensaje
[
<!ELEMENT mensaje (para,de,asunto,contenido)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
]>
```

Figura 36: Definición de la estructura permitida mediante DTD (archivo: "Mensaje.dtd").

```
<xs:element name="mensaje">

<xs:complexType>
  <xs:sequence>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="asunto" type="xs:string"/>
    <xs:element name="contenido" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Figura 37: Definición de la estructura permitida mediante un Schema (archivo: "mensaje.xsd").

La especificación del W3C recoge la definición de un módulo de software llamado **procesador de XML**, su fin residirá en ser usado para leer documentos XML y proveer acceso a su contenido y estructura.

Uno de los usos más extendidos de XML en la actualidad se ha dado en Internet y más concretamente en el entorno del desarrollo Web, sobre todo porque facilita el almacenamiento de información, así como su transporte e intercambio. Dada la sencilla naturaleza textual que presenta es altamente compatible con una gran mayoría de sistemas y aplicaciones. No es infrecuente encontrarse determinados tipos de bases de datos implementados con estructuras XML [44].

En la actualidad, XML ha sido la base para desarrollar un número importante de nuevos lenguajes, como por ejemplo:

- XHTML, la versión mejorada de HTML
- WSDL, empleado para describir servicios Web
- WAP y WML, lenguajes de marcado para dispositivos móviles
- RSS, destinado a las famosas suscripciones a canales de contenido
- RDF y OWL, utilizados para describir recursos y ontologías
- SMIL, empleado para describir multimedia con destino a la Web
- XML Music, que posibilita la construcción de partituras musicales

También se puede pronosticar que el XML será clave en el desarrollo de la web semántica.

3.4. Recuperación de Información con Lucene

3.4.1. Necesidad de nuevas herramientas para la recuperación de información. Aparición de Lucene.

Con el objetivo de dar sentido a la complejidad percibida en el mundo que los rodea, las personas crean categorías, clasificaciones, géneros, especies y otros múltiples tipos de esquemas de organización jerárquicos. El sistema decimal de Dewey [5] para realizar las signaturas de los libros en las bibliotecas en un clásico

ejemplo de esquema con organización jerárquica. La explosión de Internet y los repositorios electrónicos de información han traído consigo cantidades ingentes de información al alcance de la mano. Algunas compañías, como Yahoo!, han hecho de la organización y clasificación de la información *online* su oportunidad de negocio. Sin embargo, con el paso del tiempo, la cantidad de datos disponible ha llegado a ser de tan vasta magnitud que se han hecho necesarios nuevos métodos más dinámicos para recuperar información. A pesar de que sea factible y conveniente clasificar los datos, rastrear entre cientos o miles de categorías y subcategorías ya no es un método eficiente para encontrar información.

La necesidad de encontrar rápidamente información entre conjuntos extremadamente grandes de datos no se limita al mundo de Internet, porque, aun en mucha menor medida, la capacidad de los ordenadores personales es cada vez mayor y con ello la cantidad de datos. La opción de cambiar directorios y expandir hasta saturar las jerarquías de ficheros, no es una vía efectiva para acceder a los documentos almacenados. De hecho, cada vez los ordenadores se usan menos por sus capacidades de computación y más como reproductores multimedia o dispositivos de almacenamiento multimedia; y este cambio en los usos requiere nuevas herramientas que permitan encontrar rápidamente un elemento específico. Es necesario conseguir que los medios más ricos -imágenes, vídeos y archivos de audio, en múltiples formatos- sean fáciles de localizar [6].

Por tanto, la evolución de las tecnologías de la información y el uso que de ellas se hace, ha revelado paulatinamente que la simple organización y estructuración no es suficiente para recuperar muchos documentos de interés. Así es como surge el estudio de la **Recuperación de Información**, constituida como una disciplina de amplia investigación de utilidad en diversos ámbitos, como el académico, científico y comercial. En concreto, el problema que se desea atajar con su aplicación es el de recuperar de entre toda la información disponible sólo la información relevante [2].

Con la abundancia de información y el tiempo como un bien muy preciado para la mayoría de las personas usuarias, se requiere la capacidad de hacer consultas más flexibles, de forma variable y *ad-hoc*, que puedan atajar rápidamente los límites de la rígida categorización y encontrar en un instante lo que el usuario busca con el mínimo esfuerzo para él o ella.

Para ilustrar la permanente presencia de las búsquedas, baste señalar que hoy en día los principales navegadores, las interfaces gráficas para la gestión de ficheros de los sistemas operativos, e incluso los reproductores multimedia,

incluyen la aplicación de búsqueda incrustada.

En el momento actual, la compañía Google [35] se ha colocado en una posición de dominio gracias a un buscador que comenzó a funcionar de manera diferente al resto [21]. En una situación en la que sus desarrollos adquieren importancia incluso en el ámbito de búsquedas para aplicaciones, sus competidores intentan presentar soluciones alternativas. Microsoft adquirió *Lookout*, un producto que se apoya en *Lucene.Net*, una versión portada de Lucene para indexar y buscar en el correo de *Outlook;* y en sus actuación más reciente ha tratado de mejorar la funcionalidad de MSN con su buscador *Bing* [38]. Yahoo! ha realizado la compra de *Overture* [53] como herramienta de marketing; y un acuerdo reciente, firmado a finales de julio de 2009, entre Microsoft y Yahoo! establece la utilización de la herramienta *Bing* de Microsoft para los buscadores de las dos compañías, así como de *Overture* y otras soluciones para anunciantes, de Yahoo! [39].

3.4.2. Lucene

3.4.2.1. *Descripción de Lucene*

Lucene [26] es una herramienta de alto rendimiento y escalable para la recuperación de información (**IR**, de *Information Retrieval* en inglés). Lucene constituye un proyecto de larga trayectoria, gratuito y de código abierto -*Open Source*-, que proporciona la capacidad de añadir indexación y búsqueda a muchas aplicaciones. Forma parte de la familia de proyectos de Apache – Yakarta, y de hecho tiene licencia Apache de software libre. Actualmente, se considera como una de las librerías de Java para IR más populares.

Lucene proporciona un simple y potente núcleo de API que hace innecesario un conocimiento profundo acerca de la indexación de un texto y los procesos de búsqueda. Con sólo aprender acerca de algunas de sus clases es posible comenzar a integrar Lucene en una aplicación. Debido a que es simplemente una librería Java, no existen asunciones a priori sobre qué se indexa o se busca, lo cual supone una ventaja frente a algunas herramientas de búsqueda, cuyas premisas pueden ser contraproducentes para según qué casuísticas.

No se debe confundir la librería de Lucene con una aplicación lista para el uso, como un programa de búsqueda de archivos, un buscador tipo *"crawler"*, o una herramienta para búsqueda Web. No en vano, se trata exclusivamente de una librería software, un conjunto de utilidades o funciones al servicio de las aplicaciones que se conciban para hacer uso de ellas, conjunta o separadamente.

Las funciones se centran en la indexación de texto y en la búsqueda, y se pretende que sea con máxima eficiencia, de manera que la aplicación sólo tenga que enfrentarse con los problemas propios de su dominio mientras la complejidad de la implementación de indexación o búsquedas queda oculta tras una API de uso sencillo.

Un número cuantioso de aplicaciones de búsqueda se construyen *sobre Lucene*, o indirectamente sobre *frameworks* elaborados a partir de Lucene. Tal es el caso de muchas aplicaciones ejecutables en ordenadores personales y alojadas en sitios Web. Entre ellas se pueden citar: *Zilverline*, *SearchBlox*, *Nutch*, etc. Además, con la ayuda de Lucene se han creado herramientas de búsqueda para *Eclipse IDE*, la Enciclopedia Británica, *FedEx*, la clínica *Mayo*, *Netflix*, *Linked In*, *Hewlett-Packard*, *New Scientist magazine*, *Salesforce.com*, *Atlassian (Jira)*, *Epiphany*, *MIT OpenCourseware*, *DSpace*, la plataforma *Akamai's EdgeComputing*, *Digg*, y una larga lista de ejemplos [7].

3.4.2.2.*Historia de Lucene*

Lucene fue escrito originalmente por Doug Cutting, quien lo hizo disponible para descargar desde su domicilio a través del espacio Web de *SourceForge*. Contaba el autor con una significativa experiencia teórica y práctica en el área de IR, con varias publicaciones previas acerca de la materia, y en su currículo constaban experiencias en Excite y Apple, entre otras compañías. El proyecto se unió a la familia Jakarta de la *Apache Software Foundation* en 2001, y desde entonces ha seguido creciendo con su respaldo, como atestigua el lanzamiento ininterrumpido de versiones mejoradas a lo largo de los años.

En la actualidad, el equipo "*core"* de Lucene se encuentra formado, además de por su creador original, por una docena de desarrolladores activos de forma permanente. Aparte de los desarrolladores oficiales, existe una comunidad técnica de usuarios activa y de gran magnitud, que frecuentemente contribuye en la creación de parches, reparación de errores y adición de nuevas funcionalidades.

Por su parte, Doug Cutting se ha involucrado también en diversos proyectos de forma paralela a su participación en Lucene. En el año 2004, guiado por su preocupación ante el descenso del número de buscadores Web, y el monopolio que en su opinión se podía producir en este ámbito, crea *Nutch* [27], el primer buscador WWW íntegramente *Open Source*, diseñado para rastrear (hacer *crawling*), indexar y buscar entre varios miles de millones de páginas que se actualizan con

frecuencia. En el núcleo de *Nutch* se encuentra precisamente Lucene. Por otro lado, también se ha encontrado inmerso en el desarrollo de *Hadoop*, un proyecto encaminado a ofrecer herramientas para la computación y almacenamiento distribuidos [25].

Tabla 8: Historial de las diferentes versiones de Lucene [7].

Versión	Fecha de lanzamiento	Hitos
0.01	Marzo 2000	Primer lanzamiento *open source* (*SourceForge*).
1.0	Octubre 2000	
1.01b	Julio 2001	Último lanzamiento en *SourceForge*.
1.2	Junio 2002	Primer lanzamiento en Apache Jakarta.
1.3	Diciembre 2003	Formato de índice compuesto, mejoras en el *QueryParser*, búsqueda remota, posicionamiento de *tokens*, API de puntuación extensible.
1.4	Julio 2004	Clasificación, extensión de *queries*, *term vectors*.
1.4.1	Agosto 2004	Corrección de errores para la mejora de la clasificación.
1.4.2	Octubre 2004	Optimización del *IndexSearcher* y mejoras varias.
1.4.3	Noviembre 2004	Mejoras varias.
1.9.0	Febrero 2006	Campos almacenados en binario, herramientas de fecha, herramientas numéricas, filtro de rango, *RegexQuery*. Requiere Java 1.4.
1.9.1	Marzo 2006	Corrección de errores en *BufferedIndexOutput*.
2.0	Mayo 2006	Supresión de métodos obsoletos
2.1	Febrero 2007	Borrar/actualizar documento en *IndexWriter*, cierre de simplificaciones, mejoras en el *QueryParser*, contrib/benchmark.
2.2	Junio 2007	Mejoras de rendimiento, *Function queries*, Payloads, campos pre-analizados, políticas de borrado a medida
2.3.0	Enero 2008	Mejoras de rendimiento, políticas de fusión a medida y planificadores de fusión, fusiones en segundo plano por defecto, herramienta para detectar índices corruptos, *IndexReader.reopen*.
2.3.1	Febrero 2008	Reparación de ligeros defectos de la versión 2.3.0
2.3.2	Mayo 2008	Reparación de ligeros defectos de la versión 2.3.1
2.4.0	Octubre 2008	Mejoras profundas de rendimiento, semántica transaccional (*rollback*, *commit*), método *expungeDeletes* para cancelar borrados, supresión por *query* en *IndexWriter*
2.4.1	Marzo 2009	Reparación de ligeros defectos de la versión 2.4.0
2.9	Próximamente	
3.0	Próximamente	

3.4.2.3.*Versiones portadas de Lucene*

A pesar de que Lucene está escrito en Java, existen versiones portadas a otros lenguajes vinculadas al original, incluyendo Perl, Python, Ruby, C/C++, PHP, y C# (.NET). Esto ha permitido un fácil acceso a muchas funcionalidades de Lucene desde aplicaciones escritas en diversos lenguajes de programación.

3.4.3. Indexación y búsqueda

En el contexto de los grandes sistemas con un número importante de ficheros en los que hallar información, es imprescindible una opción de búsqueda óptima. No es válida una solución sencilla que consista en escanear de forma secuencial todos y cada uno de los archivos hasta encontrar el elemento deseado. Generalmente se hace necesario un proceso previo para facilitar la búsqueda, conocido como indexación (anglicismo derivado de la palabra *index*) que, tal y como su propio nombre sugiere, consiste en crear un índice. La indexación de ficheros de texto genera una estructura de datos, el índice, que posibilita un rápido acceso aleatorio a cada una de las palabras que se almacenan en ella. Estos términos hacen referencia a los presentes en los documentos, de forma análoga a como se referencian los tópicos presentes en un libro -citando todas las páginas en las que aparecen-, en este caso se indica en qué documentos se encuentran. La estructura de datos de este índice se diseña expresamente para este fin, y generalmente se guarda dentro del sistema de ficheros del equipo que ejecuta la aplicación como un conjunto de archivos de tipo *index*.

Con Lucene se puede indexar y preparar para la búsqueda cualquier dato que pueda ser convertido a formato textual. No es relevante la fuente de datos, como tampoco lo es su formato ni su lenguaje, siempre que sea convertible a texto. Por ello, es posible emplear Lucene para indexar y buscar datos en ficheros de páginas Web en servidores remotos, mensajes de correo electrónico, conversaciones de chat, documentos de Microsoft Word, ficheros PDF, HTML o cualquier formato del que se pueda extraer información textual. Mediante el empleo de sus funciones, es posible indexar el contenido de una base de datos en forma tal que se le conceda al usuario la posibilidad de una búsqueda textual de gran flexibilidad. En el siguiente ejemplo (Figura 38) se detalla una búsqueda íntegramente compatible con Lucene, en la cual se impone la presencia de un determinado término (signo '+'), se excluye la presencia en los resultados de unos términos concretos (signo '-'), se exige la presencia de varios a la vez ('AND') y se particularizan casos de interés concretos dentro de una categoría:

```
+Luis +Manzanas –comida –pudding, Apple –pie +Tiger, animal:mono AND comida:plátano
```

Figura 38: Ejemplo de sintaxis de búsqueda para Lucene.

Con el objetivo de entender como encaja Lucene en una aplicación de búsqueda, cuál es su cometido y cuál no, se puede situar en el contexto de una moderna aplicación de búsqueda, como en la siguiente figura.

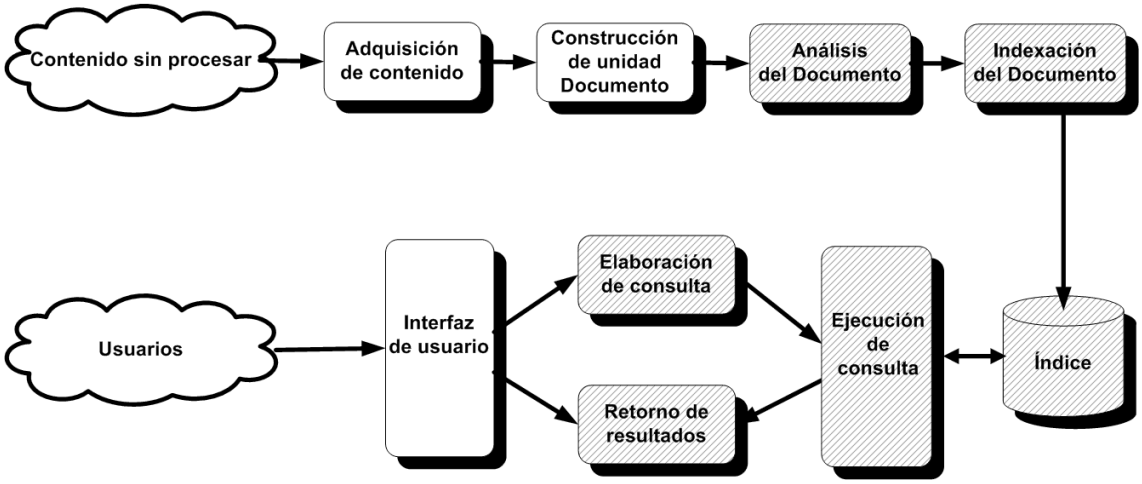


Figura 39: Componentes típicos de una aplicación de búsqueda. Los componentes sombreados indican los aspectos que son manejados por Lucene [7].

3.4.3.1.*Proceso de indexación*

A)Adquisición de contenido

El primer paso para realizar la indexación, como se puede observar en la Figura 39, expuesta por los propios desarrolladores de Lucene, es adquirir el contenido. Este proceso que frecuentemente se considera la labor de un crawler o una "araña", **agrupa** y **delimita** el contenido que requiere ser indexado. Esta labor será tanto más complicada cuanto mayor sea la dispersión de los contenidos.

Los derechos de acceso de los ficheros pueden dificultar la realización correcta de esta fase, su configuración puede ser tal que se impida que procesos de usuarios no autorizados lean su contenido. En estos casos sería necesario adquirir los derechos de acceso o *ACLs* antes de adquirir el contenido, y agregarlos como un campo adicional a los documentos, el cual pueda ser empleado durante la fase de búsqueda para preservar las restricciones.

Para grandes conjuntos de contenido es deseable que exista una eficiencia de tipo incremental, que sólo se acceda en cada ejecución a los documentos que han cambiado con respecto a la anterior. También puede tratarse de un servicio de los que se denominan "vivos", en los cuales existe una ejecución permanente encargada de detectar los contenidos nuevos o modificados de forma instantánea, con el fin de que su contenido sea agregado tan pronto como esté disponible.

Lucene no provee de ninguna funcionalidad específica para adquirir

contenido. Por tanto será una tarea que concierna por completo al desarrollador que implemente la aplicación global de búsqueda, pudiendo emplear éste algún módulo software ya existente en el mercado. De hecho, existen un buen número de crawlers de carácter *open source* disponibles:

- Nutch proporciona un crawler, altamente escalable adecuado para descubrir contenidos realizando el rastreo de sitios Web [27] .
- Grub, es una popular aplicación open source que funciona como crawler Web. Evidentemente, no tiene relación con el gestor de arranque dual de sistemas operativos de idéntico nombre [50] .
- Heritrix, un crawler desarrollado íntegramente en Java por Internet Archive [36] .
- Apache Droids es un proyecto que se encuentra en lo que se conoce como fase de incubación por parte de la fundación Apache; se trata de una herramienta con la que gestionar 'robots' de rastreo [24].
- Aperture ofrece el soporte necesario para rastrear lugares Web, sistemas de ficheros y buzones de correo electrónico, y a partir de ello extraer e indexar su contenido [31] .

Las herramientas que han sido citadas pueden revelarse como de gran utilidad en escenarios con gran dispersión de los contenidos, como es el caso de la Web, ya que el diseño de programas que cumplan esta función suele ser en extremo complicado. En otros ámbitos puede no ser necesario el empleo de aplicaciones tan complejas.

B)Construcción de documentos

El siguiente paso consiste en **organizar** el contenido adquirido. Los contenidos textuales extraídos, independientemente de su origen, se agrupan en torno a unos elementos básicos que generalmente se denominan documentos. Es necesario dejar claro que esta unidad *documento* es una agrupación realizada por el sistema analizador, no tiene por qué guardar relación con un documento tangible previamente. Un documento está constituido por **campos de contenido** con un nombre determinado, como por ejemplo: *autor*, *descripción*, *localización*, *título*, etc.

Una decisión importante en el diseño del sistema será el modo en que un flujo de contenido se divide en documentos. En ocasiones, la división puede ser trivial, así el contenido de una página Web, un mensaje de correo o un archivo PDF,

suele abarcarse dentro de una única unidad *documento* con unos determinados campos. En otros casos, la decisión sobre el procedimiento óptimo no resulta tan inmediata; a modo de ejemplo, los archivos adjuntos de un correo electrónico pueden tratarse como un único documento que reúna los contenidos de todos ellos, o cada uno de ellos como un documento que contase con una referencia a su origen común.

Tras las decisiones de diseño previas, la aplicación se encarga de extraer contenido textual a partir del contenido sin procesar que se ha adquirido previamente. La operación comporta diferentes niveles de dificultad según la naturaleza de los documentos tratados, ya que existen documentos que son de naturaleza textual para los que la transformación requerida es mínima, mientras otros muchos, de uso frecuente –como los documentos de *Open Office*, *Microsoft Office*, o archivos de audio y vídeo-, presentan naturaleza binaria y requerirán un filtro específico que extraiga el contenido en formato textual para introducirlo dentro de los campos correspondientes. Determinados documentos, fundamentalmente en HTML o XML, se estructuran en torno a un contenido textual con un marcado concreto, en esto casos el filtrado consistirá en eliminar estas marcas, que no obstante se considerarán para organizar el texto del contenido resultante en campos de documentos.

En este paso, es frecuente la aplicación de técnicas de optimización. De tal manera, en documentos con campos de textos muy extensos se pueden añadir campos adicionales con datos concretos extraídos de los anteriores, como nombres, lugares, fechas y tiempos, etc. Por otro lado, es habitual introducir consideraciones de relevancia para documentos y campos, por ejemplo que los más recientes tengan prioridad frente a los que lo sean menos, o cualquier otro criterio definido en el diseño. En la etapa posterior de indexado generalmente existen también otros mecanismos que conceden más importancia a un campo de un documento frente al mismo campo de un documento diferente; la mayoría de las aplicaciones introducen de forma automática una mayor relevancia a los campos de texto de menor tamaño frente a los de mayor (de manera intuitiva, no tiene la misma importancia que un término aparezca varias veces en un texto de miles de palabras que en uno con apenas una decena), y así se contempla en los métodos de Lucene.

Lucene provee de una API que permite elaborar las unidades documento con sus respectivos campos, pero no proporciona una lógica para construirlos, los criterios para este fin y las herramientas para conseguirlo han de fijarse en cada aplicación implementada. No existe ninguna clase de Java Lucene que proporcione

filtros documentales. En este punto se hacen necesarias herramientas como los llamados *parsers* o analizadores sintácticos (cuya función en este caso es interpretar la sintaxis de la estructura de cada tipo de archivo).

Existen *parsers* para formatos específicos de uso muy extendido. Es necesario mencionar Xerces [29], otro proyecto englobado dentro de Apache que tiene por finalidad proporcionar un amplio conjunto de *parsers* para documentos XML. Para ello se vale de SAX (*Simple API for XML*) [45] , una API elaborado inicialmente para Java que fue el primero en facilitar las tareas de manipulación de XML, y que hoy en día se puede considerar un estándar *de facto*; así como de DOM [52] (*Document Object Model*), una API que sirve para modelar el contenido de los documentos con el objetivo de manipularlo o extraerlo. Mediante su uso es posible llevar los textos contenidos entre etiquetas, en la estructura de un archivo, a los campos adecuados de una unidad documento, creada con los métodos de Lucene.

También existe un subproyecto dentro de la 'familia' software de Lucene, Tika [28], que ofrece instrumentos para gestionar el filtrado de un buen número de tipos documentos, y que sería por ello de gran utilidad en esta etapa. Tika posibilita la extracción de contenidos textuales de múltiples formatos de archivos. Permite extraer tanto propiedades como el texto contenido en archivos de *Microsoft Office* (para lo que emplea *Apache POI*, una API Java para acceder a archivos con la estructura propietaria de formato *Microsoft*). Igualmente se puede emplear para otros muchos formatos usados directa o indirectamente para recoger texto, y también para archivos de naturaleza radicalmente diferente de los que no se puede extraer los contenidos, pero sí la información textual contenida en los metadatos. Tika puede usarse, por ejemplo, para obtener datos asociados a archivos comprimidos (en formato *zip*, *tar*, y *gzip*, entre otros), así como metadatos de grabaciones de audio en formatos *WAV* y *MP3* (etiquetas ID3v1). En el caso de audio sintetizado *MIDI*, se vale del paquete básico javax.sound.midi para conseguir extraer valores de la secuencia contenida.

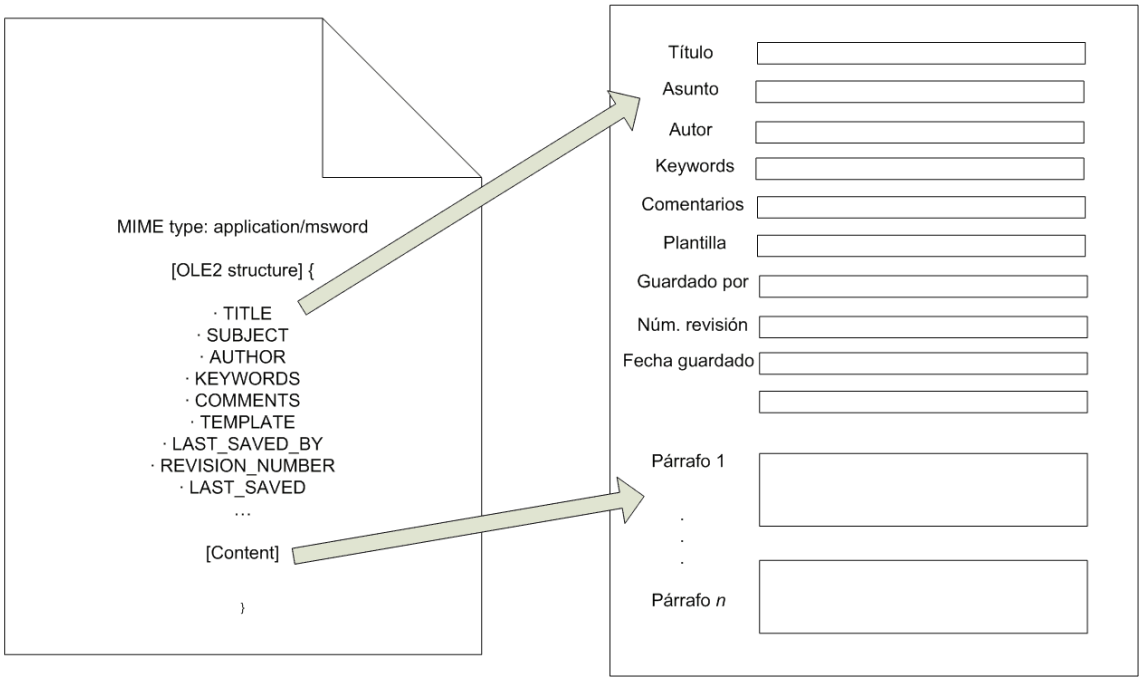


Figura 40: Ejemplo de extracción de contenido con Tika para un documento Lucene. El contenido es extraído como una serie de párrafos.

C)Análisis de documentos

El texto de los campos normalmente no se indexa de forma inmediata tras la elaboración del documento, sino que ha de ser analizado antes. El proceso de análisis implica la **división** en pequeñas porciones, más conocidas por su expresión en lengua inglesa como ***tokens***, y que si se trata de textos lingüísticos serán equivalentes a las palabras aisladas. Tras este proceso, el texto de un campo documental quedará definido como una serie de *tokens*.

En este punto se introducen muchas optimizaciones que se aplican a los *tokens* de forma individual, especialmente en el caso de los textos lingüísticos. Así, es habitual quedarse con la raíz de las palabras, de manera que los nombres y adjetivos sean singulares y neutros, los verbos presenten su forma en infinitivo, e incluso se busquen sinónimos de los términos para que la búsqueda sea más flexible y obtenga resultados más acertados. De igual modo, para otros textos que representen contenido no lingüístico se podrán emplear mecanismos equivalentes, en cada caso adecuados a la naturaleza del documento (el término textual de un *token* no corresponderá a una expresión del lenguaje, será una representación mediante texto de un valor, pero generalmente también se podrán establecer equivalencias entre ellos).

Lucene incorpora una amplia gama de analizadores integrados en sus métodos que permiten un control preciso de las diferentes opciones para este proceso. En cualquier caso, también se puede decidir crear un analizador personalizado para satisfacer unas necesidades específicas. Tras el proceso de análisis el contenido estará preparado para entrar en la fase de indexado.

D)Indexación de documentos

En el siguiente paso, la unidad documento es agregada a un **índice**. Se puede afirmar que la API de Lucene proporciona precisamente la mayor parte de lo necesario en este paso.

Lucene almacena el contenido en una estructura conocida como **índice inverso**. De este modo se consigue un uso más eficiente del espacio en disco además de permitir un rápida búsqueda de términos. Consiste exactamente en tomar los términos obtenidos tras el análisis, los *tokens*, y emplearlos como claves en la búsqueda en vez de utilizar los documentos que los contienen. En otras palabras, se responde a la cuestión: ”¿qué documentos contienen el término 'Z'?”, en vez de: “¿qué términos están contenidos en el documento 'A'?”.

Es extremadamente importante resaltar que todos estos pasos detallados antes y durante la indexación suelen tener un único objetivo principal: obtener los mejores resultados posibles en la búsqueda. Todas las mejoras realizadas en el proceso tendrán esa meta máxima.

3.4.3.2.*Proceso de búsqueda*

La búsqueda se puede definir como la acción que tiene por objetivo encontrar en un índice los documentos en los que aparece un determinado término.

Como ya se ha adelantado, este es el proceso clave que se busca optimizar, pero no sólo en términos de rapidez o precisión: existen múltiples factores a perfeccionar para mejorar la experiencia de usuario. De modo que se pueden citar la capacidad para introducir consultas con mayor número de términos, con comodines, términos confusos, así como la ordenación y/o ranking de resultados. Todas ellas son prestaciones que en Lucene se pretenden ofrecer.

La forma en que a continuación se detalla el proceso completo de búsqueda repara en la estructura desde la parte más externa al usuario, la interfaz, a la más interna, la elaboración y retorno de una consulta en el índice. Constituye básicamente un camino de ida y vuelta, el que va de la formulación de una consulta del lado del usuario, pasando por la realización de la consulta y la resolución en el

índice, hasta la presentación del resultado de nuevo hacia el usuario.

A) Interfaz de usuario de la búsqueda

La interfaz de usuario puede situarse en el entorno de una página web, una aplicación local en la máquina del usuario o en un dispositivo móvil, por citar algunos ejemplos. El factor común, entre cualquiera de sus modalidades, es la finalidad de servir como medio para la interacción con el usuario de la aplicación de búsqueda.

La importancia de la interfaz no es desdeñable, de hecho es frecuente considerarla la pieza clave en el éxito de una aplicación, así lo señalan en sus textos precisamente algunos de los desarrolladores más relevantes de Lucene [6] [7]. Tanto es así que se pueden encontrar casos en los que el funcionamiento del sistema sea realmente eficaz, con un diseño interno potente que se valga de los desarrollos más recientes como Lucene u otros similares, y que sin embargo presenten un pequeño fallo en su interfaz que provoque su absoluto fracaso cuando éste se presenta a los usuarios.

La API de Lucene no incluye ninguna interfaz por defecto, por tanto es algo que deberá ser implementado necesariamente para la aplicación concreta. Tras la interacción del usuario con la interfaz se genera una petición de búsqueda que se debe transformar en una determinada consulta para el sistema.

B) Elaboración de consultas

Tras la interacción con la interfaz por parte del usuario, se produce como resultado una **petición** de búsqueda. En el escenario más típico, tal hecho sucede tras el envío de un formulario web de un explorador hacia un servidor. La petición ha de ser traducida en una **consulta** para el sistema de búsqueda, más comúnmente llamada ***query***, y que en el caso de Lucene se constituirá en un objeto de la clase *Query*.

Los objetos *Query* son heterogéneos y pueden ser de considerable complejidad. Lucene provee del paquete *QueryParser* para procesar el texto introducido y traducirlo en un objeto *Query*. El texto admitido por el *QueryParser* sigue el formato de una sintaxis común, marcada por la especificación [23]. Una *query* puede contener frases (siempre entre comillas dobles), operadores booleanos, comodines, etc.

Muchas aplicaciones requerirán modificar la consulta para resaltar o filtrar los elementos más importantes, especialmente si no se ha ejecutado ningún

mecanismo para modificar la relevancia de términos en la fase de indexación.

A menudo, la funcionalidad que ofrece el *QueryParser* por defecto de Lucene cumple con las necesidades de la mayoría de aplicaciones. En otras ocasiones, será necesario tomar la salida del *QueryParser* y añadir la lógica propia necesaria para adecuar mejor el objeto *Query* a los fines deseados. Y, en cualquier caso, siempre es posible reelaborar la sintaxis de *QueryParser* y definir qué instancias de *Query* se crean, gracias a la característica *open source* de Lucene.

Tras la elaboración de la consulta en la forma adecuada, el siguiente paso es ejecutar la consulta sobre el índice.

C) Búsqueda de consultas

El proceso posterior consiste en buscar en el índice las coincidencias con la *Query*, que serán organizadas de acuerdo al orden que se defina. Éste constituye el trabajo interno más complejo para la herramienta de búsqueda, y en el caso de Lucene puede permanecer totalmente encapsulado (al modo de una 'caja negra') bajo su API de fácil manejo. No obstante, en este punto también es perfectamente posible la extensión y modificación de la implementación original de Lucene, para lograr una determinada forma de agrupación, filtrado u ordenamiento de los resultados.

Existen tres modelos principales de búsqueda, tal y como explica la teoría de la IR [2]:

- **Modelo booleano puro**: Los documentos reflejados en el índice se clasifican sólo en dos estados posibles, coincidencia o no coincidencia, pero no se establece ninguna puntuación que lleve a que un resultado esté mejor situado que otro. El resultado de la consulta es simplemente un subconjunto del total de elementos que reúne a todos los coincidentes.

- **Modelo de espacio vectorial**: Tanto la consulta como los documentos se modelan como vectores en un espacio multidimensional, en el cual cada término único constituye una dimensión. La relevancia o similitud entre una consulta y un documento concreto se calcula de acuerdo a la distancia existente entre estos vectores.

- **Modelo probabilístico**: En este caso, se calcula la probabilidad de que un documento sea relevante para una consulta de acuerdo a un modelo probabilístico. Para este modelo es habitual que exista una realimentación

que permita redefinir las probabilidades de relevancia, de acuerdo a las anteriores interacciones con el sistema.

Lucene emplea una aproximación basada en dos modelos: el **vectorial** y el **booleano puro**.

D) Presentación de los resultados

Tras obtener un conjunto de documentos considerados coincidentes con la consulta, ordenados según corresponda, es necesario presentarlos al usuario. Preferiblemente se presentarán en un formato que sea intuitivo y de utilidad, en muchos casos no se deberá tratar como un punto final, sino como otro elemento de interacción para que el usuario pueda redefinir la búsqueda, buscar documentos similares, etc. Lucene no ofrece ninguna interfaz por defecto, tampoco para la presentación de resultados, por tanto, en caso de hacer uso de sus prestaciones, será éste otro aspecto que concierna por completo al desarrollador.

3.5.Antecedentes de búsqueda musical: Musipedia

Existen en la actualidad un buen número de ejemplos de sistemas cuya finalidad es localizar y/o identificar melodías musicales. De entre todos ellos, se procede a detallar a continuación el que reúne mayor similitud con el que se ha pretendido desarrollar en este proyecto: ***Musipedia***.

Musipedia [42] es un buscador dirigido a identificar fragmentos de obras musicales. Para ello permite varios métodos de búsqueda: introducir una melodía en un teclado web virtual, marcar un determinado ritmo mediante el teclado del ordenador, introducir un código de Parson (que se basa en las subidas y bajadas de tono de la melodía) o, incluso, silbar la melodía. Cualquier persona puede modificar la colección de melodías e introducir los archivos MIDI, las imágenes con las partituras (que pueden ser generadas por el propio servidor del sistema al introducir código fuente de *Lilypond*[1] o *abc*[2]), las letras y texto acerca de la composición, así como el contorno melódico del código Parson; todo ello constituye la información asociada a cada elemento almacenado en su base de datos.

[1] *LilyPond* es un programa de software libre para elaborar partituras con carácter multiplataforma que permite la personalización y extensión por parte del usuario. Utiliza una sencilla notación de texto como entrada y funciona por línea de comandos.

[2] *Abc* es un lenguaje para notación musical que se vale para su fin del conjunto de caracteres ASCII, por ello se puede emplear en cualquier sistema con gran facilidad

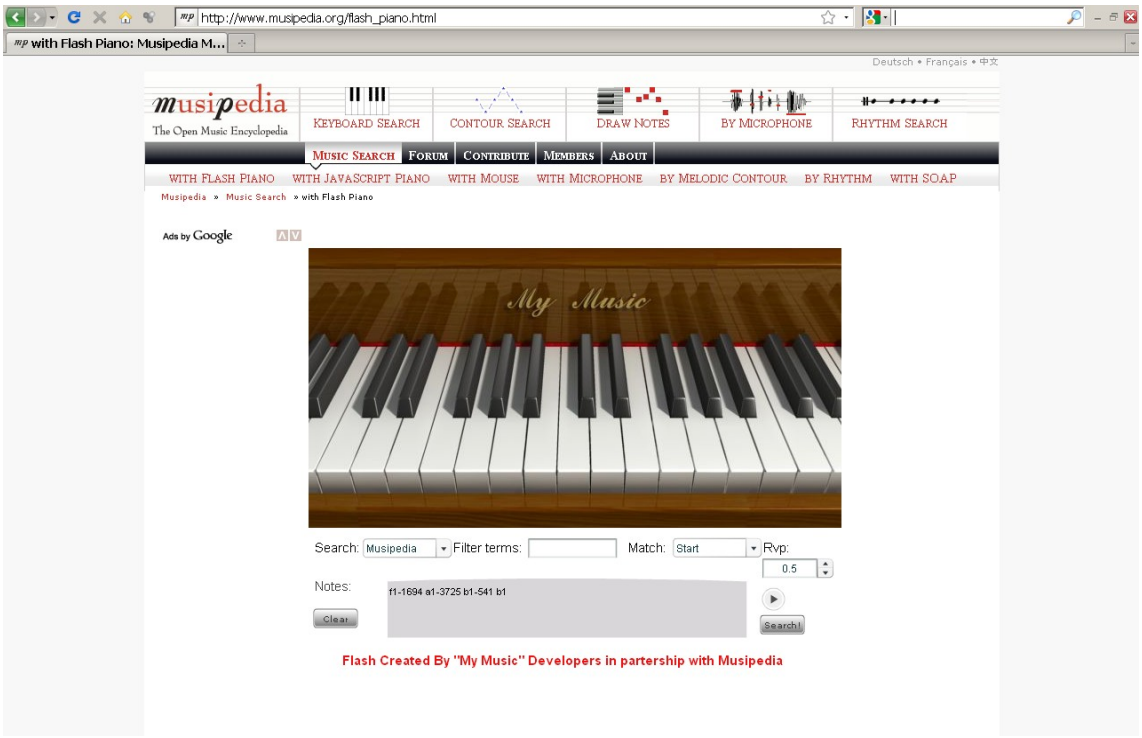


Figura 41: Página de búsqueda de Musipedia.

3.5.1.Evolución de Musipedia

Musipedia empezó de la mano de Rainer Typke en 1997, quien se ha encargado de su desarrollo desde entonces hasta el momento actual. La herramienta recibía el nombre de *Melodyhound* antes de que se introdujera la metodología de colaboración de tipo *wiki*, la cual se implantó en 2004, en pleno "despegue" de la *web 2.0*, para permitir así que cualquier usuario pudiera editar y eliminar las entradas existentes. Desde 2006 su potencia de búsqueda se ha ido incrementando gracias a la posibilidad de buscar ficheros MIDI en cualquier lugar de la *World Wide Web*. Para este fin se emplea el servicio que ofrece la plataforma *Alexa Web Search*, propiedad de la compañía *Amazon*, y que en esencia es lo que se denomina un *crawler* de la Web al que se pueden realizar peticiones de contenidos específicos [41].

3.5.2.Descripción

Musipedia ofrece tres posibilidades de búsqueda: basada en el contorno melódico, en las notas con su temporización respectiva, o en el ritmo exclusivamente.

La búsqueda por contorno melódico emplea una distancia flexible, por ello el mecanismo de búsqueda encuentra no sólo las entradas que corresponden con el

contorno exacto introducido, sino también aquellas que, sin coincidir rigurosamente, más se le parecen. La similitud se cuantifica determinando unos pasos de modificación (insertar, suprimir o reemplazar algún carácter) que se necesitan para convertir el contorno buscado en aquel que ha devuelto como resultado el sistema. Como en esta opción de búsqueda sólo es relevante el contorno melódico, se pueden encontrar resultados satisfactorios aunque se desconozcan los tonos, ritmos o intervalos exactos entre notas.

La búsqueda basada en tonos y duraciones tiene en cuenta más propiedades de la melodía. Se emplea por defecto y es invariante ante las transposiciones de alturas de las notas y modificación del tempo: sólo toma en consideración los intervalos y el ritmo.

Las posibilidades para introducir la melodía son varias, destaca a modo de ejemplo el teclado virtual web manejable mediante ratón. El mecanismo de búsqueda posteriormente segmenta la consulta, convirtiendo cada segmento en un conjunto de puntos en el espacio bidimensional formado por tono y duración. Después realiza el cálculo de lo que, coloquialmente, se conoce como "Distancia de la mudanza de la Tierra", una medida matemática de la distancia entre dos distribuciones sobre una determinada región. Según explica la teoría que la sustenta, si estas dos distribuciones se tratan como dos montones de materia, la distancia es el mínimo coste de llevar la totalidad de uno hacia el otro. Esta medida se emplea habitualmente en otros ámbitos, por ejemplo, para comparar archivos de imágenes deteriorados por el ruido y en general en la recuperación de imágenes basada en el contenido -para lo cual se compara la distancia entre histogramas-. En el caso de Musipedia, se obtiene la medida de distancia entre los segmentos introducidos y los de la melodía almacenada, y se logra de esta manera estimar y devolver los resultados óptimos con cierta tolerancia a errores.

La búsqueda por pulsaciones sólo toma en cuenta el ritmo, emplea el mismo algoritmo que en el anterior, con la particularidad de asumir que todos los tonos son iguales.

En el algoritmo de búsqueda empleado se logra rapidez mediante índices basados en objetos "aventajados". En vez de calcular la distancia entre los segmentos de la consulta y los de cada entrada de la base de datos, en primer término, Musipedia sólo calcula la distancia entre la consulta y cada uno de los elementos de un pequeño subconjunto de objetos a los que se ha concedido ventaja. En un segundo paso, el algoritmo realiza una observación más cercana sólo a aquellas entradas de la base de datos cuya distancia a los objetos

aventajados sea similar a la distancia existente entre estos últimos y la consulta.

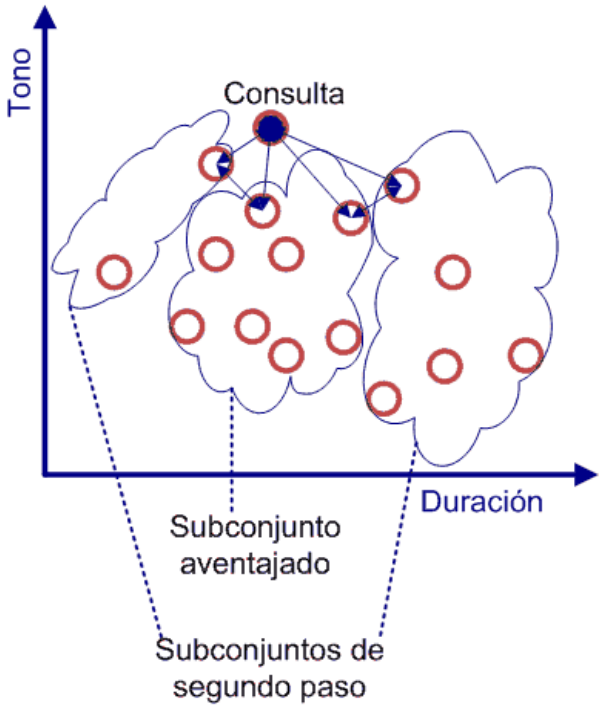


Figura 42: Representación gráfica del algoritmo de Musipedia.

3.5.3. Diferencias entre Musipedia y las herramientas de identificación de audio

La herramienta de búsqueda de Musipedia funciona de forma diferente a como lo hacen otras herramientas tales como *Shazam* [46], *LASSO* [11], *AudioID* [33], *Gracenote* [47] y un largo etcétera, a las que se podría denominar como herramientas de identificación. Las aplicaciones de este tipo pueden identificar cortos fragmentos de audio, unos pocos segundos procedentes de una grabación, incluso a través de una conexión telefónica. Emplean una técnica que se denomina *Audio Fingerprinting* (huellas digitales de audio), que permite a los usuarios llamar a un número de teléfono con el objetivo de que el sistema identifique la canción que en ese momento están escuchando. Estas aplicaciones cuentan con una base de datos en la que almacenan las "huellas digitales", una especie de firma que se extrae de los aspectos más relevantes de una grabación de audio. Mediante el uso de un algoritmo concreto, los valores físicos de un segmento de sonido se transforman en un valor numérico -huella- exclusivo para ese fragmento. Algunas de las características que se toman en cuenta para constituir esta marca unívoca

son: energía, sonoridad, centroide espectral, tasa de cruces por cero, *pitch* o tono fundamental, armónicos, monotonía espectral, coeficientes *cepstrales* de frecuencias de *Mel* –MFCC-, etc. Con cierto conocimiento acerca de la ciencia psicoacústica, se puede afirmar que la mayoría de las cualidades que se extraen son además relevantes desde el punto de vista perceptual. A partir de la pieza musical consultada, se extraen estas características y mediante el mismo algoritmo se genera su "huella" para compararla con las almacenadas en la base de datos.

Musipedia se diferencia de los sistemas de identificación en que encuentra obras musicales que contengan una melodía dada, mientras que los sistemas de identificación encuentran una determinada grabación exacta que contiene un fragmento dado, pero no son capaces de encontrar otras grabaciones que contengan este fragmento, aunque pertenezcan a la misma obra.

4.Diseño del sistema

En el presente sistema se pretende proporcionar una aplicación de búsqueda de contenidos melódicos musicales. En su diseño se abarcan tanto las etapas previas necesarias para dotar de contenidos, como las correspondientes a la realización de búsquedas sobre el conjunto total de disponibles.

4.1.Funcionalidad del sistema

El sistema se divide en dos grandes bloques de acuerdo a los procesos que se llevan a cabo, tal y como se puede observar en la figura de la página siguiente: **indexación** y **búsqueda**. Los bloques se definen de acuerdo a una estructura modular, concebida para la fácil integración con nuevos componentes futuros.

El bloque de indexación agrupa toda la funcionalidad que tiene por objetivo agregar contenido al índice. Este bloque incluye un módulo de **Extracción de contenidos** a través del cual recibe las entradas al sistema, consistentes en archivos musicales en formato MIDI junto a sus textos descriptivos asociados. A partir de las entradas, el módulo extraerá la información que empleará para generar como salida un documento estructurado. Tras este proceso, comienzan las labores necesarias para agregar el contenido al índice, desempeñadas por el **Constructor** y el **Indexador**.

El bloque de búsqueda da sustento a la aplicación de búsqueda del usuario, gobernada por una interfaz que permitirá al usuario definir los términos deseados. El elemento **Buscador** tendrá encomendada la misión de servir de intercomunicador e intérprete entre el usuario y el índice, permitiendo la realización de consultas sobre este último y la devolución de resultados al primero.

El planteamiento ha pretendido respetar las etapas recomendadas para un sistema de IR basado en Lucene, recogidas en los apartados 3.4.3.1. y 3.4.3.2. Una particularización, de hecho, de los procedimientos generales aplicables a cualquier sistema de IR.

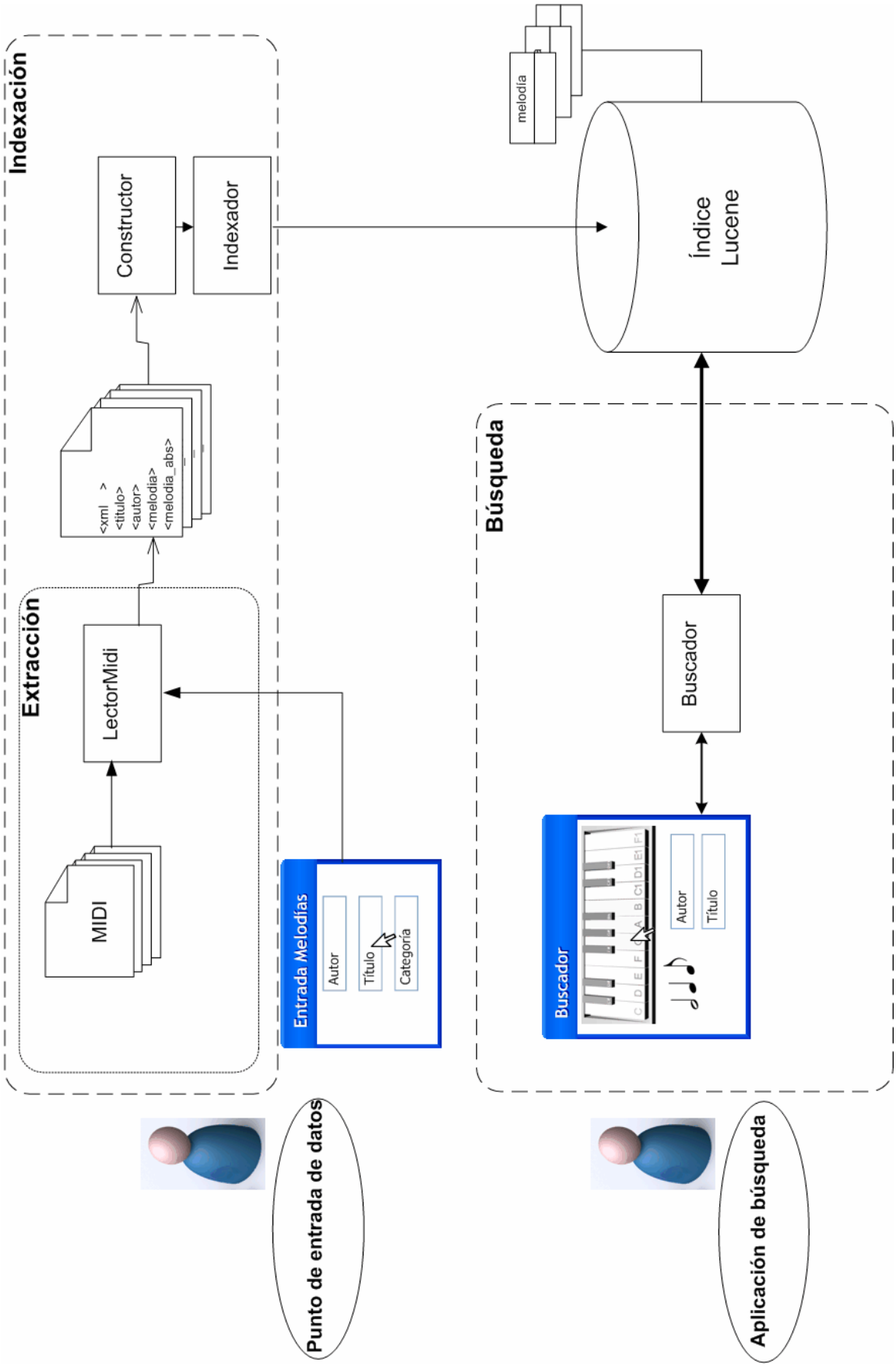


Figura 43: Diagrama de bloques del sistema.

4.2. Bloque de indexación

En este proyecto se ha pretendido conseguir un diseño efectivo de un sistema de búsqueda de fragmentos musicales. Los dos puntos de partida son por tanto una información musical almacenada y una información musical consultada, que deberán relacionarse adecuadamente.

Según las teorías de la Recuperación de Información (IR), a la hora de almacenar los documentos se ha de buscar maximizar la semejanza entre informaciones que de forma veraz sean parecidas o equivalentes. Si un elemento A y un elemento B se considera que tienen gran parecido, será porque tendrán algunas características, alguna información, en común; lo mismo puede suceder entre el elemento B y un elemento C, con otra serie de características en común. La representación que se haga de cada uno de los elementos recogerá datos que reflejen las mismas características para todos, pero deberá ser tal que entre esas características tengan cabida: las que asemejan a A con B, las que los diferencian entre sí, las comunes de B con C y las diferentes, y así sucesivamente para todos los elementos. De esta manera, la representación de los datos ha de facilitar la identificación de similitudes entre elementos [2].

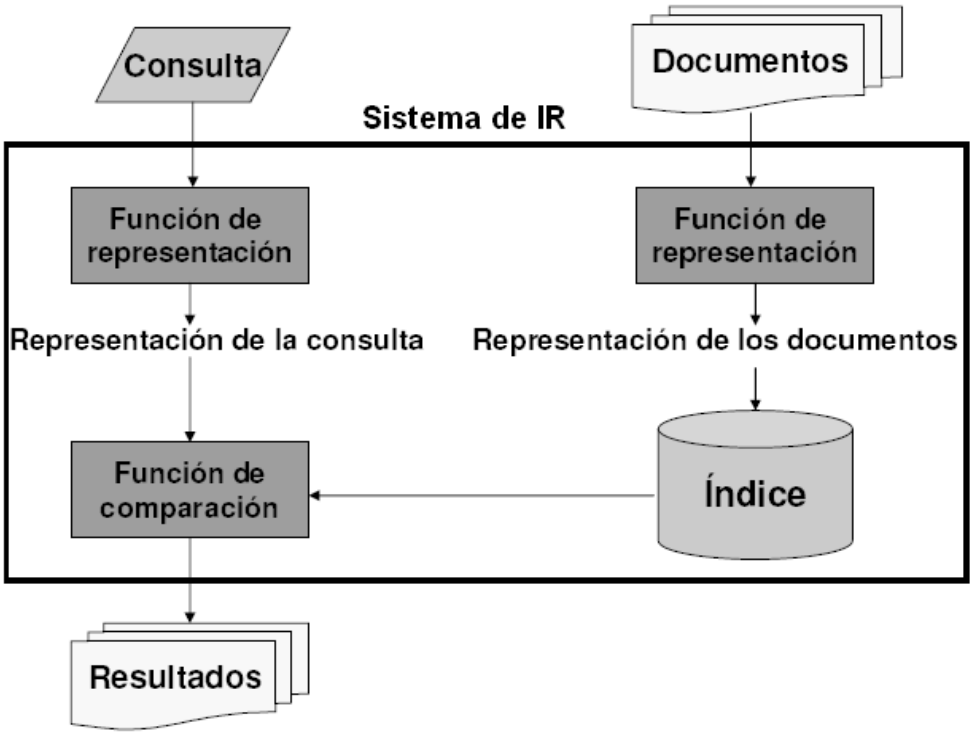


Figura 44: Diagrama funcional genérico de un sistema de IR.

El acierto en el modelo de representación de la información impactará de manera decisiva en la efectividad del sistema completo, ya que facilitará la identificación de elementos coincidentes. De ahí la importancia de definir una representación adecuada de forma previa a la indexación, obtenida tras identificar correctamente las características a resaltar.

Los contenidos musicales que se tratarán en este sistema tienen una naturaleza característica que determinará de manera fundamental las decisiones de diseño del sistema. A pesar de que desde el punto de vista físico se puedan caracterizar por su comportamiento ondulatorio, de acuerdo a una serie de componentes en frecuencia fundamentales, las composiciones musicales sólo se pueden identificar unívocamente mediante su partitura.

Según la teoría de IR, es necesario definir una función de representación que dé lugar a una representación óptima de los documentos que figurarán en el índice. La función de representación será ejecutada por el módulo de extracción, por medio del *LectorMidi*, éste se encargará de representar los fragmentos introducidos de acuerdo a sus notas musicales. Dicha función deberá tener su equivalente en el módulo Buscador para así poder establecer una función de comparación válida entre las consultas y los elementos del índice, que permita encontrar los contenidos coincidentes.

La representación de los contenidos mediante sus notas, en lugar de mediante parámetros físicos, resulta ser la manera más lógica, además de posibilitar la independencia de las circunstancias físicas que afectan a una interpretación concreta (diferentes instrumentos, recintos o sistemas de captación que producen componentes frecuenciales distintas). Por tanto, la tradicional representación en forma de figuras musicales, constituye en sí misma una representación normalizada. Sin embargo, aún es necesario ir un paso más allá y decidir un modo concreto de parametrizar las notas para dotar de máxima expresividad a las búsquedas.

4.2.1. Modelo de representación de melodías

Se debe diseñar una representación de las notas de tal modo que los elementos equivalentes queden representados de igual forma. La equivalencia en términos musicales queda definida atendiendo a las cuatro características fundamentales que se detallan a continuación.

A)Tono

El tono o altura de las notas es uno de los parámetros más relevantes para el significado musical. Permite establecer diferencias claras entre una composición y el resto. No obstante, la percepción del oyente puede diferir del original, considerando éste que las notas están algunos semitonos por encima o por debajo del tono registrado en la partitura, y no por ello se puede considerar errónea. De hecho, es frecuente que para interpretar se realicen las llamadas transposiciones, que consisten en elevar o disminuir por igual la altura de todas las notas, de forma que la composición suene varios semitonos más aguda o más grave.

Éste es el primer aspecto en el que se pueden establecer equivalencias basadas en la percepción auditiva y que se justifican por las relaciones matemáticas entre sonidos. La verdadera regla de identificación se basa en que se cumplan las mismas relaciones de altura entre las notas. De tal modo, se van a tratar como equivalentes dos fragmentos que hayan sufrido distintas transposiciones, porque la diferencia en semitonos entre las notas permanece constante.

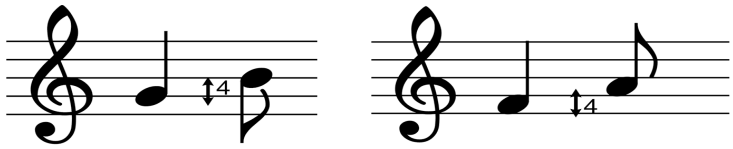


Figura 45: Ejemplo de fragmentos equivalentes en tono. Los sonidos de las notas están separados en ambos por un intervalo de 4 semitonos.

La representación, por tanto, habrá de incluir los tonos de las notas de forma relativa, es decir, reflejando los intervalos de semitonos existentes entre las notas; lo cual permitirá una identificación más efectiva que con los tonos absolutos.

B)Duración

Si el tono de las notas era clave para establecer características diferenciales entre composiciones, tanto o más lo es la duración. Es relativamente habitual que se pueda identificar inequívocamente una melodía sólo por su ritmo, marcado por las duraciones de sus sucesivos sonidos. Al igual que sucedía con las alturas tonales, la percepción del oyente se debe considerar en términos relativos, ya que aunque la interpretación de cada figura puede estar definida de una forma más o menos estricta por el tempo, el oyente generalmente lo desconoce.

A modo de ejemplo, una negra se puede interpretar en una ocasión con una duración de 1 segundo y en otra con una duración de 0,60 segundos. Este hecho puede suceder incluso para la misma composición, ya que es común que el tempo esté definido sólo de forma cualitativa con los movimientos (*allegro*, *moderato*, etc.). De hecho, las figuras de duración se definen siempre en términos relativos (la duración de una negra es la mitad de la de una blanca, el doble que la de una corchea, etc.) y son las proporciones entre notas la única norma que siempre se debe respetar. Por consiguiente, se llega a un criterio de equivalencia según el cual se consideran iguales en cuanto a su rítmica dos fragmentos si presentan las mismas proporciones entre la duración de las notas.

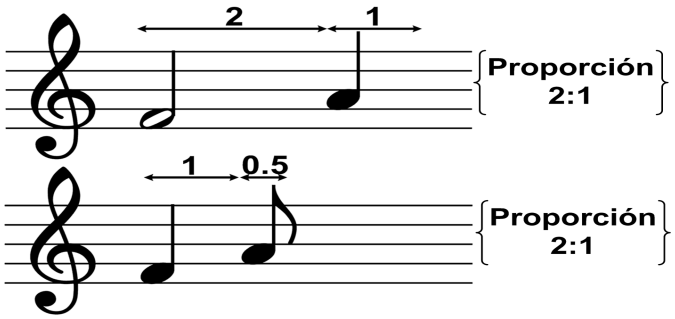


Figura 46: Ejemplo de fragmentos equivalentes en rítmica. La duración de la segunda nota es en ambos la mitad de la anterior.

La representación melódica reflejará la duración de las notas de forma proporcional, según la fracción existente entre el tiempo de cada nota y el de las anteriores. A fin de obtener mayor homogeneidad, se prescindirá de la medida de tiempo del compás, que es de gran importancia conceptual pero no facilita la identificación.

C)Timbre

La información musical almacenada respecto a los fragmentos no incluirá el timbre. A efectos de este sistema, se considerará irrelevante qué instrumento interprete una composición: se consideran equivalentes dos composiciones idénticas tocadas por distintos instrumentos. Por tanto, su timbre característico se omitirá, a pesar de que se disponga de esa información.

D)Intensidad

La intensidad tiene plena importancia en la interpretación musical. Sin embargo, no se trata de un característica suficientemente identificativa, como pueden ser el tono o la duración; es decir, un músico podría equivocarse e interpretar una composición sin respetar las intensidades marcadas, pero no se trataría de una obra diferente, sino de una interpretación errónea de la original.

En este sistema se va a considerar que todos los sonidos presentan la misma intensidad con el objetivo de hacer la búsqueda más efectiva. Desde el punto de vista del usuario que efectúa la consulta, esta simplificación facilita su tarea sin eliminar con ello un factor significativo.

La representación final de las melodías que se empleará en el índice incluirá por cada nota los tonos y duraciones en la forma descrita anteriormente. La función de representación se aplicará de la misma manera a las consultas realizadas, para así poder establecer la función de comparación entre el elemento consulta y los elementos en el índice.

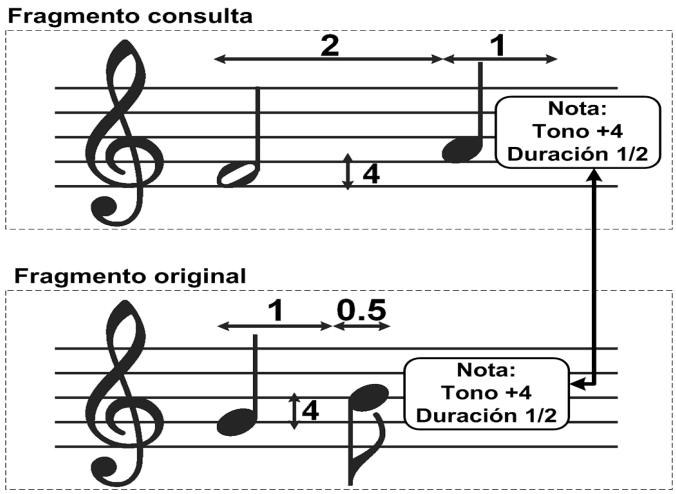


Figura 47: Ejemplo gráfico de función de comparación entre fragmento de consulta y fragmento original almacenado.

En primer lugar, se va permitir la búsqueda de melodías siguiendo criterios de correspondencia exacta de alturas tonales y figuras de duración. Para ello se guardarán también los valores originales (Do 4 negra, Si 4 corchea, etc.) codificados del modo que se explicará más adelante. Cuando se deseen ampliar las posibilidades de búsqueda, siguiendo los criterios de equivalencia expuestos anteriormente, se deberá recurrir a la representación proporcional o relativa.

Existe la posibilidad de realizar la representación relativa descrita de dos modos diferentes:

- Duraciones y tonos relativos a la **nota anterior**. En este caso se almacenaría la diferencia en altura de cada nota respecto a su nota anterior, y la razón existente entre la duración de cada una y la anterior. Presenta la ventaja de ser el método más sencillo, así como de permitir encontrar coincidencias en una melodía a partir de fragmentos más pequeños de la

misma (los fragmentos se localizarían independientemente de su posición). Como desventaja, un error en una nota afectaría a la codificación de la nota siguiente, aunque sólo a ésta; este error tendría incidencia si se produjese con varias notas y en fragmentos cortos.

- Duraciones y tonos relativos a la **primera nota**. En tal supuesto se guardaría el intervalo de diferencia existente entre el tono de cada nota y la primera, y la fracción resultante de dividir la figura de duración entre la correspondiente a la primera. Ofrece como ventaja una mayor tolerancia a errores intermedios en un número reducido de notas; como contrapartida presenta una dependencia insalvable respecto a la primera nota, lo cual, por ejemplo, haría imposible localizar un fragmento pequeño incluido en uno de mayor tamaño si éste no se encontrara al principio.

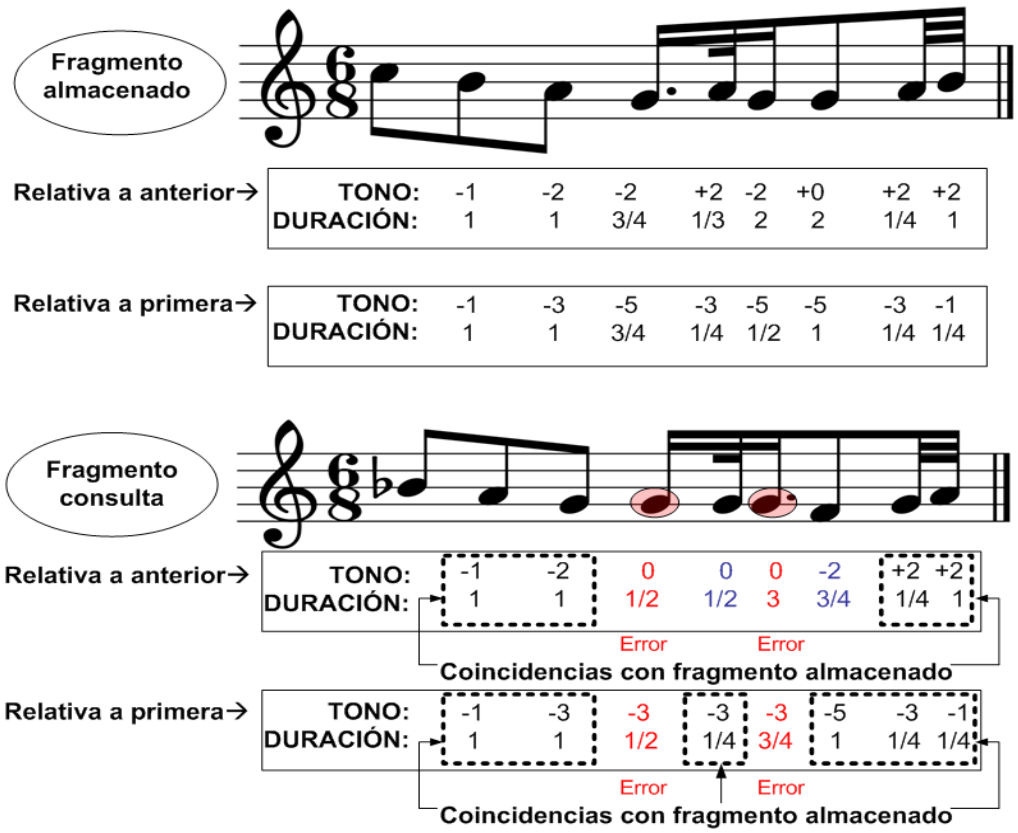


Figura 48.a: Ejemplo de las dos descripciones posibles aplicadas a un fragmento almacenado y un fragmento de consulta, este último parte de errores en dos notas intermedias.

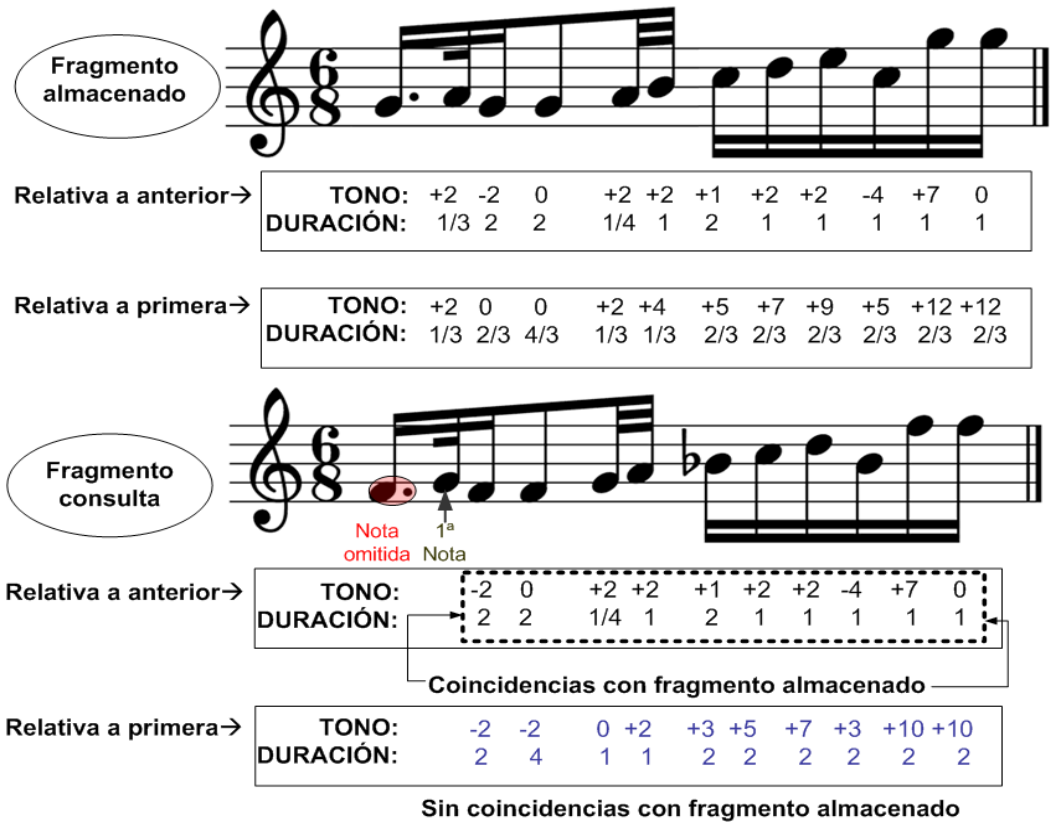


Figura 48.b: Ejemplo de las dos descripciones posibles aplicadas a un fragmento almacenado y un fragmento de consulta, este último es un subconjunto que no incluye la primera nota.

Entre las dos posibilidades es necesario optar por la que provoque un menor impacto negativo. De tal manera, se ha decidido que la notación relativa se haga respecto a las notas precedentes, así se consigue el efecto de hacer independiente los fragmentos de su situación en las melodías, y en consecuencia se podrá localizar secuencias de notas contenidas en cualquier ubicación dentro de una melodía indexada.

Esta elección se puede justificar desde los principios clásicos de la IR documental, que por naturaleza incluyen la posibilidad de encontrar los términos de una consulta independientemente de su posición en un documento. Además, se debe tener en cuenta las tendencias más habituales que un usuario puede presentar en las búsquedas: será un hecho bastante frecuente que el usuario no introduzca al completo la melodía indexada; mientras que es relativamente menos común que el usuario introduzca erróneamente tantas notas intermedias como para provocar el fracaso de la búsqueda. Y, en cualquier caso, una melodía con demasiadas notas alteradas ya no se puede considerar la misma melodía, toda vez que un fragmento incompleto sí. Por tanto, los casos en que una representación relativa a las notas precedentes es preferible se producirán de manera mucho más habitual que los casos en que puede resultar perjudicial.

4.2.1.1.*La representación de los silencios*

Los silencios son un caso particular de nota que carece de tono (por tratarse de la ausencia de sonido), pero que cuentan con una duración característica del mismo modo que las notas sonoras. En la secuencia que representa las notas de la melodía se reflejarán los silencios en la posición que corresponda, detallando la duración relativa que proceda, y que afectará como siempre a las notas posteriores. El tono no se codificará con valores numéricos como '0', porque esto provocaría un error conceptual consistente en que la nota de silencio estaría más próxima a unas notas que a otras, cuando en realidad se encuentra igualmente alejado de todas (es un sonido inexistente). No se tendrá en cuenta la presencia de silencios en lo que al tono se refiere, y así las notas anteriores y posteriores a ellos presentarán el mismo tono relativo que tendrían si no existiesen; es decir, si acontece la secuencia de notas de la siguiente figura, el tono relativo de la segunda nota sería '0' y su duración relativa '1/2':

Nota(tono=60)(duración=1) – Silencio(duración=2) – Nota(tono=60)(duración=1)

Figura 49: Ejemplo de secuencia de valores de notas con silencio.

En determinadas melodías resulta complejo recordar la presencia de silencios, sobre todo para oyentes que no estén especialmente entrenados. No en vano, por la naturaleza misma de la aplicación, existe un cierto desconocimiento acerca de la melodía buscada, desconocimiento que puede manifestarse en la omisión o modificación previsible de algunos aspectos como los silencios. Por ello, se ha decidido que la aplicación también permita la búsqueda de fragmentos sin considerar los silencios; para este fin se almacenará adicionalmente la secuencia de notas relativas sin incluirlos.

4.2.1.2.*Representaciones alternativas*

Es posible definir otras representaciones alternativas que sirvan para identificar melodías. El código de Parson [15] constituye una notación sencilla empleada para describir contornos melódicos. La primera nota se declara con un asterisco o se omite y a continuación se detallan las siguientes, representadas por una sigla:

- *U*: *up*, si la nota es más aguda que la anterior.
- *D*: *down,* si la nota es más grave que la anterior.
- *R: repeat,* si la nota presenta el mismo tono.

Su autor defiende esta representación como suficientemente identificativa, sin embargo la ausencia de indicaciones rítmicas y la escasa precisión tonal (sólo se indica subida, bajada o permanencia del tono), provocan que una misma representación en este código pueda pertenecer a diferentes composiciones; tal indeterminación no sucede cuando se emplea la codificación numérica de tonos y duraciones descrita en apartados previos.

Al mismo tiempo, su principal causa de desventaja puede convertirse en una virtud, ya que al codificar las diferencias tonales en una forma menos estricta, puede facilitar la recuperación de más fragmentos coincidentes, aunque siempre existirá el riesgo de obtener resultados con baja precisión.

En el contexto de este sistema se ha valorado como insuficiente una identificación basada en dicho código, por este motivo el mecanismo de búsqueda principal, en cuyo desarrollo se ha focalizado el esfuerzo, ha sido el fundamentado en intervalos de tonos y fracciones de duración. No obstante, se permite incluir este código con miras a efectuar búsquedas más simplificadas.

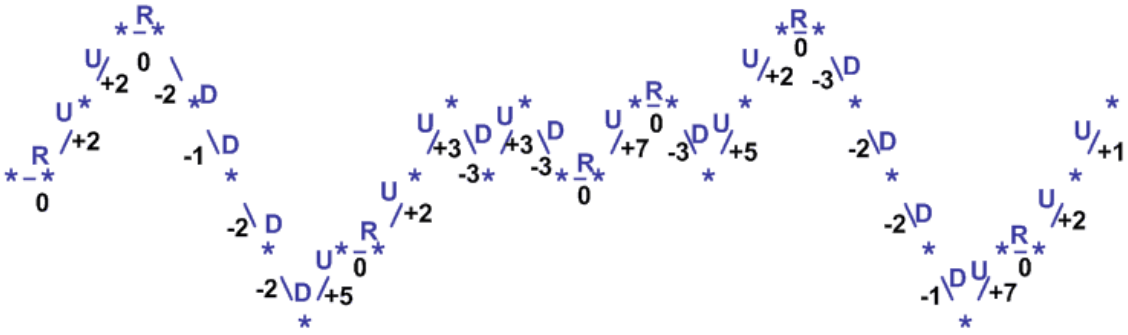


Figura 50: Comparación entre el contorno melódico descrito con código de Parson y con intervalos tonales

4.2.2.Módulos del bloque de indexación

A continuación se detallan las decisiones de diseño relacionadas con los módulos necesarios para la indexación, destinados a proporcionar las unidades de información constituyentes del índice. En concreto, se conciben el Módulo de Extracción y los módulos Constructor e Indexador.

4.2.2.1. *Módulo de extracción*

El módulo de extracción tendrá por objetivo adquirir y transformar en la forma adecuada la información destinada a ser agregada al índice del sistema. Se ha de adaptar el contenido para obtener una representación identificativa en formato textual.

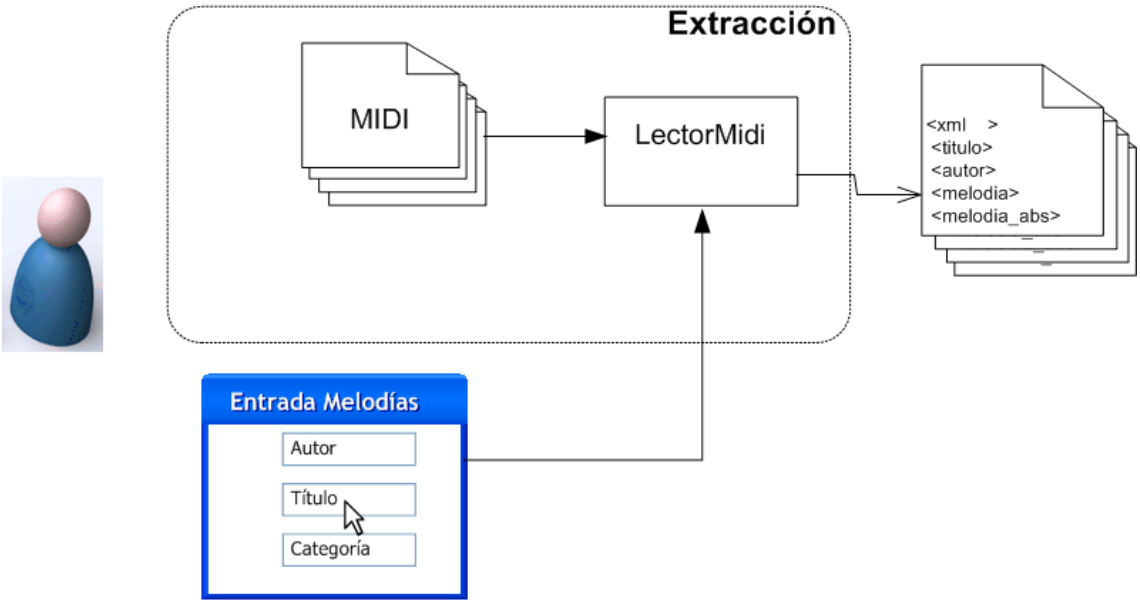


Figura 51: Bloque de indexación: módulo de extracción

El módulo de extracción está ligado a un determinado formato de entrada admitido. En este caso se tratará de archivos MIDI, los cuales presentan la ventaja de encontrarse fácilmente disponibles, además de posibilitar la fácil extracción de sonidos musicales independientes. En futuras ampliaciones del sistema se podría considerar la opción de añadir módulos para la extracción a partir de otros formatos.

El elemento central del módulo es el *LectorMidi*, al que llegarán los archivos MIDI junto a un texto descriptivo. En un escenario típico para esta aplicación, los contenidos de entrada provendrían de formularios web, en los que un usuario adjunta el archivo y consigna algún texto descriptivo -los denominados metadatos-. Se permite agregar la siguiente información:

- Autor
- Título
- Categoría o género

- Letra o descripción

- Código de contorno melódico

Se deberán tomar estos elementos junto con la representación de la melodía, que extraerá el *LectorMidi* a partir del fichero de audio sintetizado de la entrada. Es lo que recibirá el nombre de proceso de análisis melódico.

I. Proceso de análisis melódico

La información del sonido de las notas se extraerá a partir de los eventos almacenados en el fichero MIDI, los cuales presentan una estructura basada en pistas. Haciendo uso de métodos de la API jMusic, el lector traducirá los sonidos recogidos en esta estructura a notas situadas dentro de una estructura musical. En concreto, las notas se insertan en el nivel más bajo dentro de una estructura jerárquica de tres niveles; siendo el más bajo la frase, a continuación la voz, y por último la partitura.

El objetivo es obtener melodías monofónicas, por eso el *LectorMidi* realizará un proceso de serialización de las notas de manera que todas aparezcan de forma sucesiva. De estas notas se tomarán exclusivamente los valores del tono y la duración que se almacenarán de forma ordenada en una estructura de datos matricial. Aunque en principio se introducirán melodías individuales, esta decisión de diseño permite tratar composiciones con polifonía como un conjunto de melodías independientes.

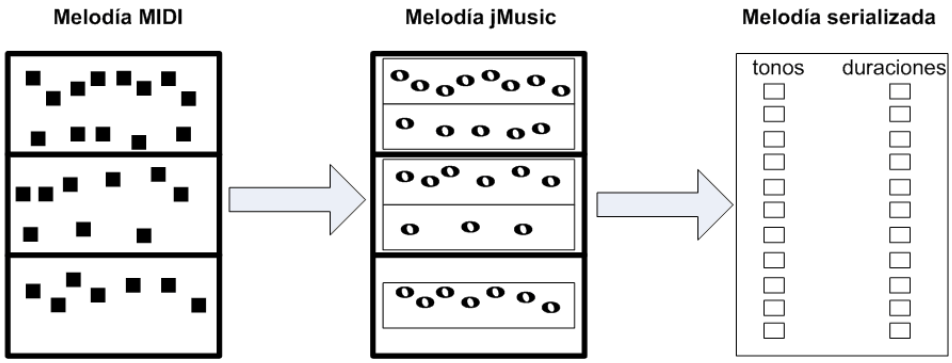


Figura 52: Esquema básico de la serialización de melodías.

El conjunto de valores de tono y duración servirá para generar el campo de información *melodía absoluta* y se empleará en el siguiente proceso, invocado de manera inmediata a éste. Será el denominado como proceso de normalización.

II. Proceso de normalización

El objetivo central del proceso será obtener las funciones de representación principales descritas con anterioridad para las melodías, en definitiva, una representación normalizada de las melodías.

Para realizar este proceso se tomarán cada uno de los valores de los tonos y se establecerá su relación con el tono inmediatamente anterior como se ha explicado en este mismo capítulo. Esta misma operación se realizará con cada una de las duraciones de las notas. Como resultado se obtendrá la representación relativa de la melodía, la cual constituirá un elemento identificativo esencial, recogida en el campo de información *melodía*.

El sistema de búsqueda permitirá realizar consultas simplificadas, basadas sólo en la rítmica -por tanto sólo consideran las relaciones de duración y son independientes del tono-, o bien sólo en los tonos -se consideran los intervalos tonales de forma independiente a la duración-. La búsqueda exclusivamente rítmica encuentra su fundamento en la constatación de que las relaciones de duración son un elemento suficientemente identificativo en muchos casos, a la par que fácilmente recordable. La búsqueda exclusiva por alturas tonales ha de fundamentarse más bien como una ayuda para el usuario, ya que en esencia carece de sentido musical prescindir de las relaciones de duración. Estas representaciones simplificadas podrían obtenerse a partir del campo *melodía* evadiendo así la necesidad de un campo de información aparte; sin embargo, tal decisión chocaría de frente con los principios que rigen un sistema de IR basado en índice: la información ha de organizarse de un modo rápidamente accesible. No es admisible el coste de calcular en la fase de búsqueda esta información, y es considerablemente más factible añadir en este proceso previo la capacidad de generar otros dos campos adicionales, uno que recoja la información sólo rítmica y otro para la que es sólo tonal. Recibirán respectivamente los nombres de *melodía sin tonos* y *melodía sin duraciones*.

Hay que tener en cuenta, además, una serie de peculiaridades que se presentan en este proceso y que esencialmente tienen que ver con los silencios.

- Debido a que el valor del tono de un silencio se ha definido como un número negativo muy inferior a cualquier otro tono de la nota, si se realizara la misma operación que en el resto de los casos, se producirían valores de tono y diferencias irreales. Es por eso que en el lugar del tono de la nota, tanto en la notación absoluta como en la relativa, se almacena una cadena vacía.

En este proceso se debe detectar la presencia de silencios para realizar la codificación adecuada, que deberá calcular el tono relativo de la nota siguiente a un silencio a partir de la última nota sonora anterior. En el lugar de su duración, sin embargo, sí se almacena la diferencia con la anterior, ya que siguen una relación de duraciones idéntica e igualmente importante que las figuras musicales.

- Se toma la decisión de posibilitar también al usuario la búsqueda sin considerar los silencios, opción que se ha de considerar en esta fase previa a la indexación para que sea efectiva. De tal forma, se tomará el conjunto de valores absolutos (tonos y duraciones) de la melodía, se omitirán los pertenecientes a los silencios y se realizará un nuevo cálculo de duraciones y tonos relativos que dará origen al campo de información identificado como *melodía sin silencios*.

Al final de este proceso, terminada la labor del módulo *LectorMidi*, se obtendrán los siguientes campos de información para representar la melodía:

- Melodía: representación relativa de tonos y duraciones.
- Melodía absoluta: representación absoluta de tonos y duraciones.
- Melodía sin tonos: representación relativa de duraciones.
- Melodía sin duraciones: representación relativa de tonos.
- Melodía sin silencios: representación relativa de tonos y duraciones, omitiendo los silencios.

A los que se añaden los campos obtenidos como descripción, de origen no automático, también llamados metadatos:

- Autor
- Título
- Categoría o género
- Letra o descripción
- Código de contorno melódico

Por cada fichero MIDI se generará un documento XML que incluirá todos los campos citados. De este modo será como se permita a los módulos posteriores de indexación acceder a la información. El XML será un medio eficaz de comunicación, que facilitará la integración en el caso de decidir incorporar nuevos módulos de extracción. Se logra así una independencia deseable con respecto a la estructura existente, mediante una estructura de información fácilmente manejable como es XML.

La estructura de estos documentos quedará definida por un documento DTD, común a todos ellos. Éste establecerá los elementos que los constituirán y el contenido que tendrá cabida dentro de ellos.

Tabla 9: Elementos del XML definidos en el DTD.

Nombre de elemento	Contenido
cancion	*autor, titulo, categoria, letra, parson, melodia, melodia_abs, melodia_st, melodia_sd, melodia_sin*
autor	Cadena de texto
titulo	Cadena de texto
categoria	Cadena de texto
letra	Cadena de texto
parson	Cadena de texto
melodia	Cadena de texto
melodia_abs	Cadena de texto
melodia_st	Cadena de texto
melodia_sd	Cadena de texto
melodia_sin	Cadena de texto

4.2.2.2. *Módulos Constructor e Indexador*

Tras los procesos necesarios de extracción, el sistema puede realizar los siguientes procesos, relacionados con la elaboración de las unidades documento que representarán a cada composición en el índice. Éstos serán, por orden, los procesos de: construcción, análisis e indexación de documentos.

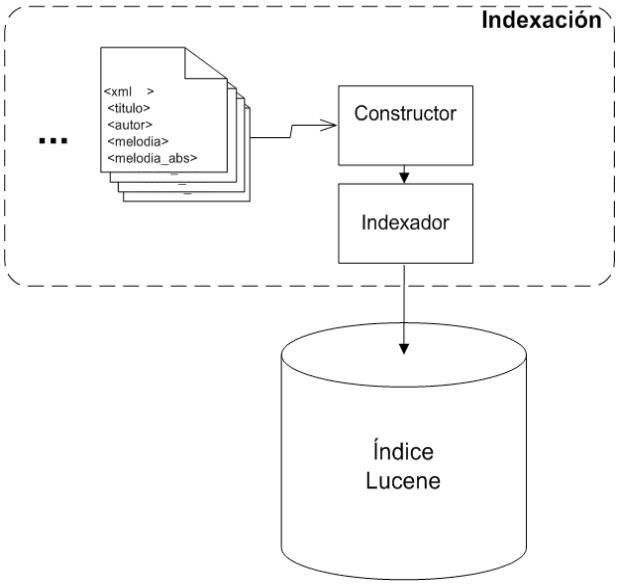


Figura 53: Bloque de indexación: módulos Constructor e Indexador.

Estas funciones se llevarán a cabo desde dos módulos que trabajan de manera conjunta: *Constructor* e *Indexador.* Se trata de dos elementos funcionalmente diferentes, que tendrán una estrecha relación entre sí, y debido a ello se implementarán en un mismo programa.

El módulo *Constructor* desarrollará el proceso de construcción, en el que se crean los campos y unidades documento que los albergarán.

El módulo *Indexador* será el encargado de los procesos de análisis e indexación. El análisis se realizará sobre los campos pertinentes justo antes de que los documentos se agreguen al índice. Después de realizar el análisis, se añaden al índice los términos de la unidad documento resultantes, para de esta forma permitir la localización de fragmentos.

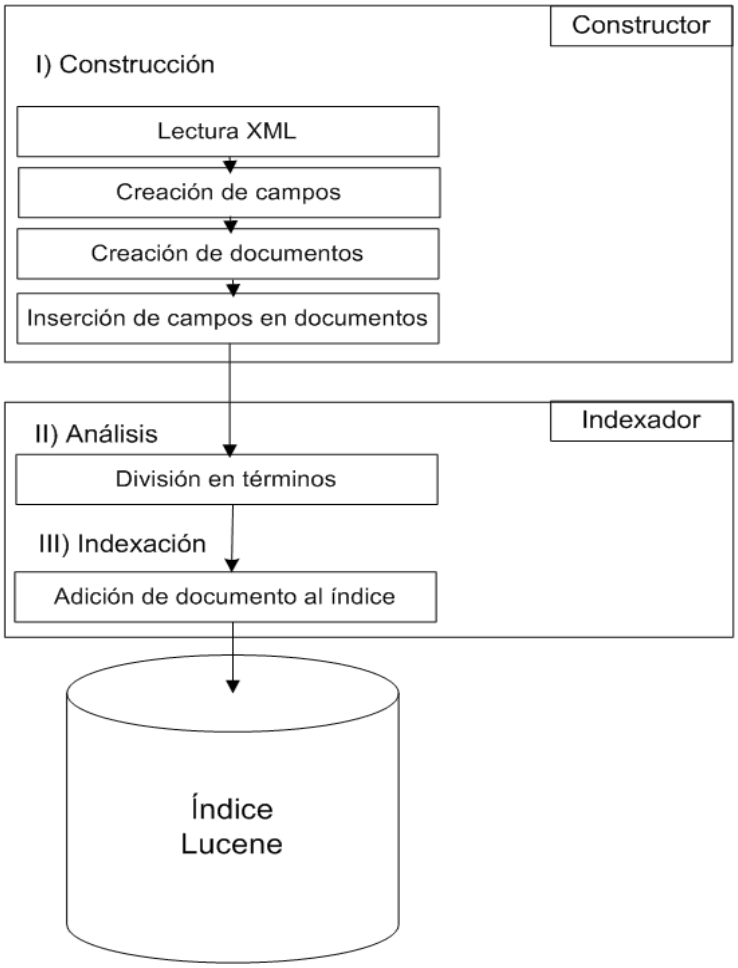


Figura 54: Diagrama de acciones desempeñadas por los módulos *Constructor* e *Indexador*.

I. Proceso de construcción

En primer lugar, el *parser* XML incluido en el módulo *Constructor* se ha de emplear para acceder a la información generada en el módulo previo de extracción. Tras la lectura del documento XML, esta información -organizada en campos como se explicó en el apartado 4.2.2.1.- se almacenará para dar origen en el siguiente paso a los campos definitivos empleados en el índice.

A continuación, deberá tener lugar la creación de los campos que formarán parte de la unidad documento dirigida al índice. En total, once campos que se identificarán como se detalla en la siguiente tabla.

Tabla 10: Campos de la unidad documento.

Nombre de campo	Definición
nombre	Nombre del archivo de audio musical
autor	Autor
titulo	Título de la composición
categoria	Categoría o género
letra	Letra del canto o texto alternativo de descripción
parson	Código de contorno melódico de Parsons descrito por el usuario
melodia	Representación relativa de tonos y duraciones de la melodía
melodia_abs	Representación absoluta de tonos y duraciones.
melodia_st	Representación relativa de duraciones.
melodia_sd	Representación relativa de tonos.
melodia_sin	Representación relativa de tonos y duraciones, omitidos silencios.

Sobre el contenido de los campos individuales será efectuada posteriormente la división en términos, salvo en aquellos que se indique lo contrario; en este sistema el único campo que no se tendrá que dividir en términos o *tokens* es el relativo al nombre de archivo.

Posteriormente, se crea la unidad documento a la que se agregarán los campos para realizar su indexación.

II. Proceso de análisis

El análisis se llevará a cabo mediante las funciones de la API Lucene en el módulo *Indexador* antes de añadir la información al índice. Es el proceso por el que las cadenas de texto contenidas en los campos se dividen en pequeños porciones, llamadas *tokens* o términos. Así, por ejemplo, cada una de las duraciones y tonos contenidos en los distintos campos de melodía quedarán identificados como términos separados. Y del mismo modo, las palabras que forman los campos de información propiamente textual. La posición de los términos es un factor esencial que deberá ser parametrizado para identificar correctamente las coincidencias en la búsqueda.

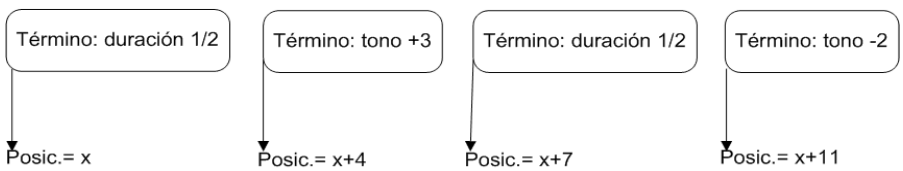


Figura 55: División de melodía en términos.

III. Proceso de indexación

El último paso necesario consiste en agregar la unidad documento al índice, operación que será realizada por el modulo *Indexador*, empleando igualmente las funciones de la API. Éste realizará una acción de escritura (sin lectura) sobre el índice, en la que incluirá las referencias a los documentos de forma inversa; es decir, por cada término de un campo debe constar en qué documentos aparece y en qué posiciones.

Por tanto, de forma conceptual se puede considerar el índice como una tabla cuyas entradas son los términos hallados en los diferentes campos. Además del documento y posición en la que se encuentran, también se hará constar otro tipo de atributos tales como la extensión del campo para ese documento concreto.

		Documento	Posición
Campo <melodia>	Término: duración 1/2	xxx	xxx
	Término: duración 1/2	xxx	xxx
	Término: tono +3	xxx	xxx
	Término: tono -2	xxx	xxx
	Término: tono -2	xxx	xxx

Figura 56: Esquema conceptual de un índice para el campo melodía.

Posteriormente, la correspondencia de una búsqueda con un documento del índice se evaluará de acuerdo a todos los indicadores disponibles. Así, será más optimo un resultado cuanto más similares sean las posiciones relativas de los términos entre sí, y para una misma secuencia tendrá mayor relevancia la que ha sido localizada en un campo de menor extensión. Tales criterios serán de aplicación para todos los posibles campos de búsqueda (melodía, autor, título, etc.).

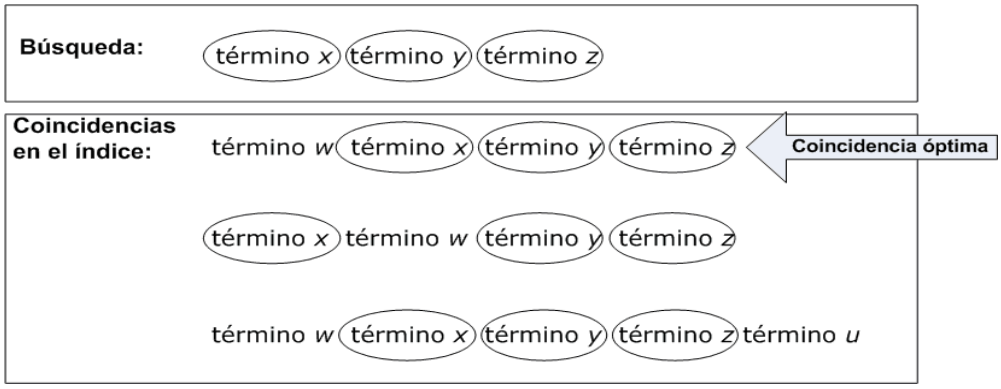


Figura 57: Ejemplo de criterios de coincidencia basados en información del índice.

Si bien el esquema conceptual detallado es totalmente válido, el índice es en realidad una estructura compleja proporcionada por la API de Lucene, que se describirá a más bajo nivel en el siguiente capítulo. Puede residir en memoria principal o en la secundaria como parte del sistema de ficheros; por simplicidad y dado que no existía exigencias extremas de rendimiento, el índice de este sistema se almacenará en memoria secundaria.

4.3.Bloque de aplicación de búsqueda

La aplicación de búsqueda será la encargada de consultar el índice para satisfacer las peticiones de búsqueda de los usuarios, con los cuales se establecerá una comunicación a través de la interfaz.

4.3.1. Módulos del bloque de búsqueda

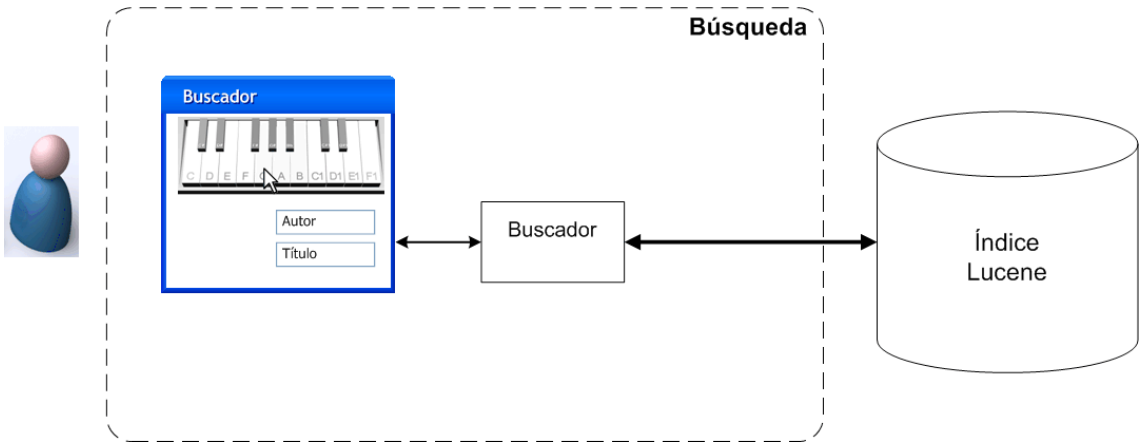


Figura 58: Bloque de aplicación de búsqueda: buscador e interfaz de usuario.

La estructura de diseño consiste en un módulo principal *Buscador* que por sí solo soporta todas las funciones de búsqueda: introducción de consultas, realización de las mismas mediante lectura del índice, y a partir de ellas la obtención y ordenación de resultados. Sin embargo para facilitar la introducción de consultas y la presentación de resultados, se debe añadir una interfaz de usuario.

Las funciones del *Buscador* tienen su complemento previo necesario en las realizadas en la indexación. No en vano, su cometido principal es servir como lector de la información del índice, en base a ella y junto con la lógica añadida, se estiman los resultados.

A pesar de su importancia, se debe considerar el módulo *Buscador* como el más ligero de todo el sistema, así lo corroboran las teorías de la IR. La mayor complejidad ha de recaer en los procesos previos a la agregación de contenidos al índice.

Las opciones permitidas para la búsqueda estarán directamente relacionadas con los campos existentes en el índice. En primer lugar, existen seis cadenas de entrada que se podrán emplear en combinación:

- Melodía
- Título
- Autor
- Categoría
- Letra
- Código de Parson

Todas ellas se relacionarán como intersección lógica (*AND* lógico) entre sí, a excepción de la melodía. En otras palabras, si se realiza la búsqueda de un texto de título junto con el nombre de un determinado autor que no corresponda, el sistema no retornará ningún resultado, aunque sí existan coincidencias para ellos individualmente en el índice. A su vez, si existen varios términos en una misma entrada (p. ej. en la entrada título), éstos se relacionaran entre sí como unión lógica (*OR* lógico).

En el caso de la melodía se aplicará un criterio menos restrictivo en relación a las demás entradas: se tomará en consideración como un *OR* respecto al resto de

ellas. Es decir, si no existen coincidencias para una determinada melodía junto con un determinado autor, pero sí se encuentran la melodía y el autor por separado en distintos documentos, se retornarán éstos como resultados.

La entrada de melodía se considera el elemento principal de búsqueda, y se permiten diferentes opciones de utilización, que llevan implícitas la búsqueda en un determinado campo del índice:

- Buscar melodías relativas: la búsqueda se realiza mediante la comparación de la consulta -trasladada a tonos y duraciones relativas- con el campo por defecto, que es el de melodías relativas.

- Buscar melodías exactas: se toman los tonos y duraciones exactos de la consulta y se efectúa su búsqueda en el campo de melodía absoluta.

- Ignorar silencios: conlleva la transformación de la consulta introducida a tonos y duraciones relativas, con la supresión de los posibles silencios, tras lo cual se realiza la búsqueda en el campo sin silencios.

- Buscar melodías independientes de la altura: implica que de la consulta sólo se toman las duraciones trasladadas a magnitud relativa, desechando los tonos, por ello la búsqueda se lleva a cabo en el campo sin tonos.

- Buscar melodías independientes de la duración: a partir de la consulta sólo se emplearán los tonos considerados en relativo, por tanto la búsqueda se realizará en el campo sin duraciones.

A continuación, tras introducir las informaciones pertinentes y definir las opciones, toda la información en formato textual se emplea en los procesos del *Buscador*. Se consideran en total tres procesos: normalización, elaboración de consultas y ejecución de consultas.

I. Proceso de normalización

Es necesario aplicar la misma función de representación que en la indexación. Por ello, las melodías que se introducen como consultas siguen un proceso de normalización análogo al realizado en el módulo *Lector*. La única diferencia es que el punto de partida en este caso será una secuencia de notas para la búsqueda que se encontrarán en un formato textual, en vez de extraerse del fichero de audio.

Tras el proceso de normalización, se habrán obtenido las secuencias de valores de las notas relativos o absolutos según el caso, las cuales servirán para elaborar la consulta junto con las entradas de texto en el siguiente proceso.

II. Proceso de elaboración de consulta

Todos las operaciones realizadas en la indexación han de tener su reflejo en la fase de búsqueda. Si se desea realizar una consulta sobre campos en el índice, el contenido de ésta ha de quedar dividido en términos, del mismo modo que lo hicieron los módulos de indexación.

Un analizador con los mismos criterios que el empleado en la fase de indexación será el encargado de fraccionar los contenidos de búsqueda en términos. Con los términos individuales obtenidos, se elaborará una consulta para ejecutar en el índice sobre los campos indicados.

III. Proceso de ejecución de consulta

La consulta se ejecutará por medio de un lector del índice, proporcionado por el mecanismo de búsqueda empleado, y se generan una serie de resultados que han de ser ordenados de acuerdo a unos criterios cerrados.

Los pasos a considerar por las funciones de búsqueda son:

1) Se encuentra alguno de los términos de la búsqueda (ya que la búsqueda es en primer término booleana, no se considera ningún documento que no contenga al menos uno de los términos exactos).
2) La comparación vectorial, que toma en cuenta las posiciones relativas de los términos, evalúa cuantitativamente el grado de similitud entre la consulta y las diferentes coincidencias.
3) Aplicación de factores de ponderación. Los campos de menor extensión cuentan con mayor relevancia que los mismos campos con mayor extensión. También existen factores de ponderación que se aplican a un campo para otorgarle de partida una mayor relevancia frente a otros; en este sistema se aplicará de partida un factor de mayor relevancia a los campos melódicos.

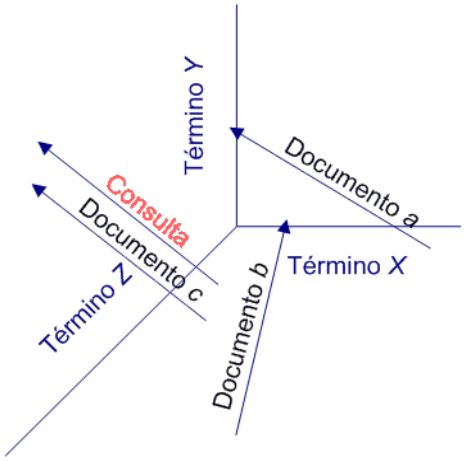


Figura 59: Método vectorial de evaluación de resultados para tres términos (espacio tridimensional).

Por último, la interfaz que interactúa con este módulo *Buscador* se tomó de un desarrollo previo realizado por el director de este proyecto, basado en el código abierto de Musipedia. La interfaz incluirá un teclado gráfico para la introducción de la consulta melódica. Si bien, no se implementarán todas las opciones, sólo se incluirán las figuras de duración de notas y silencios más habituales, así como los tonos.

Se pretenderá que la integración de la interfaz sea sencilla, de una forma ligera. Simplemente se invocará desde ella un ejecutable externo, al cual se pasarán los parámetros adecuados y los resultados serán devueltos mediante un fichero de intercambio de texto.

5.Implementación

En el presente capítulo se detallan los aspectos más relevantes de la implementación del sistema. En concreto éste ha sido desarrollado en su totalidad en Java con en el empleo de dos librerías externas principales: jMusic y, especialmente, Lucene.

5.1. Módulos del bloque de indexación

5.1.1. Módulo de extracción

El módulo de extracción de este sistema está constituido por el *LectorMidi*, implementado como una clase Java que hace uso de las clases externas de la API jMusic, necesarias para realizar la abstracción de la partitura a partir del fichero MIDI. Este módulo, básicamente, llama a la función de apertura del MIDI, y mediante un bucle va extrayendo la información serializada de todas las notas contenidas en la melodía.

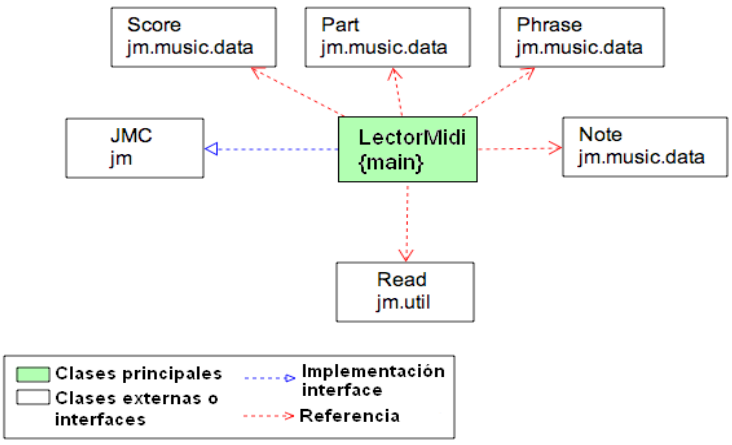


Figura 60: Diagrama de clases del módulo de extracción.

5.1.1.1.*Clases de la librería jMusic*

El uso de jMusic permitirá realizar un representación inicial de los fragmentos más próxima al lenguaje musical tradicional. Dicha representación seguirá una jerarquía basada en las clásicas partituras.

I. Estructura de datos

En la estructura jMusic, la **nota** *(Note)* es la unidad menor. Cada nota se almacenará junto con otras notas en la siguiente unidad superior, la **frase** *(Phrase).* Una o varias frases formarán una **parte** *(Part) y* una o varias partes una **partitura** (*Score*).

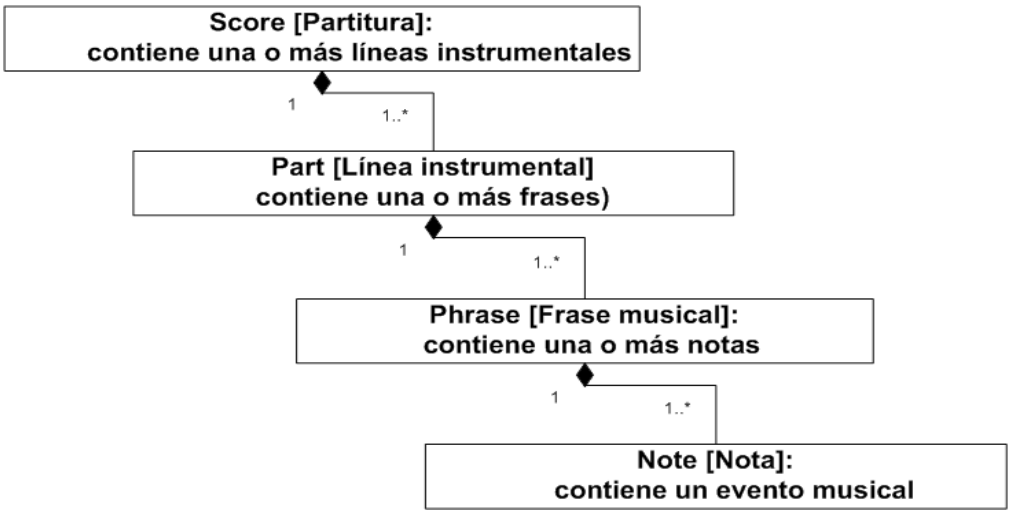


Figura 61: Estructura jerárquica de jMusic

A)Note

Contiene la información relevante acerca de un único evento musical, que consta de los atributos definidos en la tabla 11.

Tabla 11: Algunos parámetros de la nota en jMusic

Nombre	Descripción
Pitch	Tono o altura de la nota Ejemplo: C4 (La 4), 440.0 Hz, 69 (valor MIDI).
Dynamic	Intensidad. Valores para *piano, forte*, etc…
Rhythm Value	Duración o longitud de la nota, figura. Blanca, negra, corchea, redonda…
Pan	Panorama, posición de las notas en la imagen sonora estéreo: izquierda, derecha, centro…
Duration	Duración de la nota en milisegundos.
Offset	Tiempo de retraso desde el comienzo normal de una nota (anacrusa)

Las notas están contenidas en frases (*Phrase*). Al igual que en la notación musical, las notas son reproducidas una tras otra según hayan sido incluidas en la frase.

B)Phrase

Las frases, se pueden equiparar desde el punto de vista musical a las distintas voces de una partitura. Los objetos de clase *Phrase* sólo contienen un único atributo importante, una lista de notas que pueden ser añadidas, eliminadas o cambiadas de posición dentro de la frase. Existe un tipo especial de *Phrase*, el *CPhrase*, que modula estructuras musicales homófonas, normalmente combinaciones de notas simultáneas como acordes.

C)Part

Contiene las notas pertenecientes a un instrumento. Cada parte contiene un vector que recoge varias *Phrase*. Estas frases corresponden todas a sonidos similares, como por ejemplo las distintas voces de los clarinetes en una obra. Una parte tiene asociados un título, un "canal" y un instrumento.

D)Score

La clase *Score* representa el nivel superior de la estructura de datos: la partitura. Contiene un vector de partes y un título.

El objeto de clase *Score*, contiene uno o más de la clase *Part*, que a su vez puede incluir uno o varios *Phrase*, que darán cabida a una o varias *Note*. Las notas musicales, modeladas por esta última, puede contener diferentes atributos.

II. Constantes

Las constantes se definen en jMusic para poder referirse a distintos atributos musicales. Existen constantes para el tono o frecuencia, figuras rítmicas, dinámica, timbre, balance, duración articulada, escala, etc.

A)Tono (Pitch)

Las notas se escriben en notación de tono y octava, de tal manera que el Do central es denominado C4, el Re de esa misma escala, D4 y el Si de la escala anterior B3. Las alteraciones tonales se distinguen con S para indicar un sostenido y F para indicar un bemol. Por ejemplo, el Do sostenido central (Do# 4) sería CS4.

Los tonos se codifican como números enteros para hacer referencia a su valor al igual que en el estándar MIDI. El rango posible se encuentra entre CN1 (Do-1) y G9 (Sol 9).

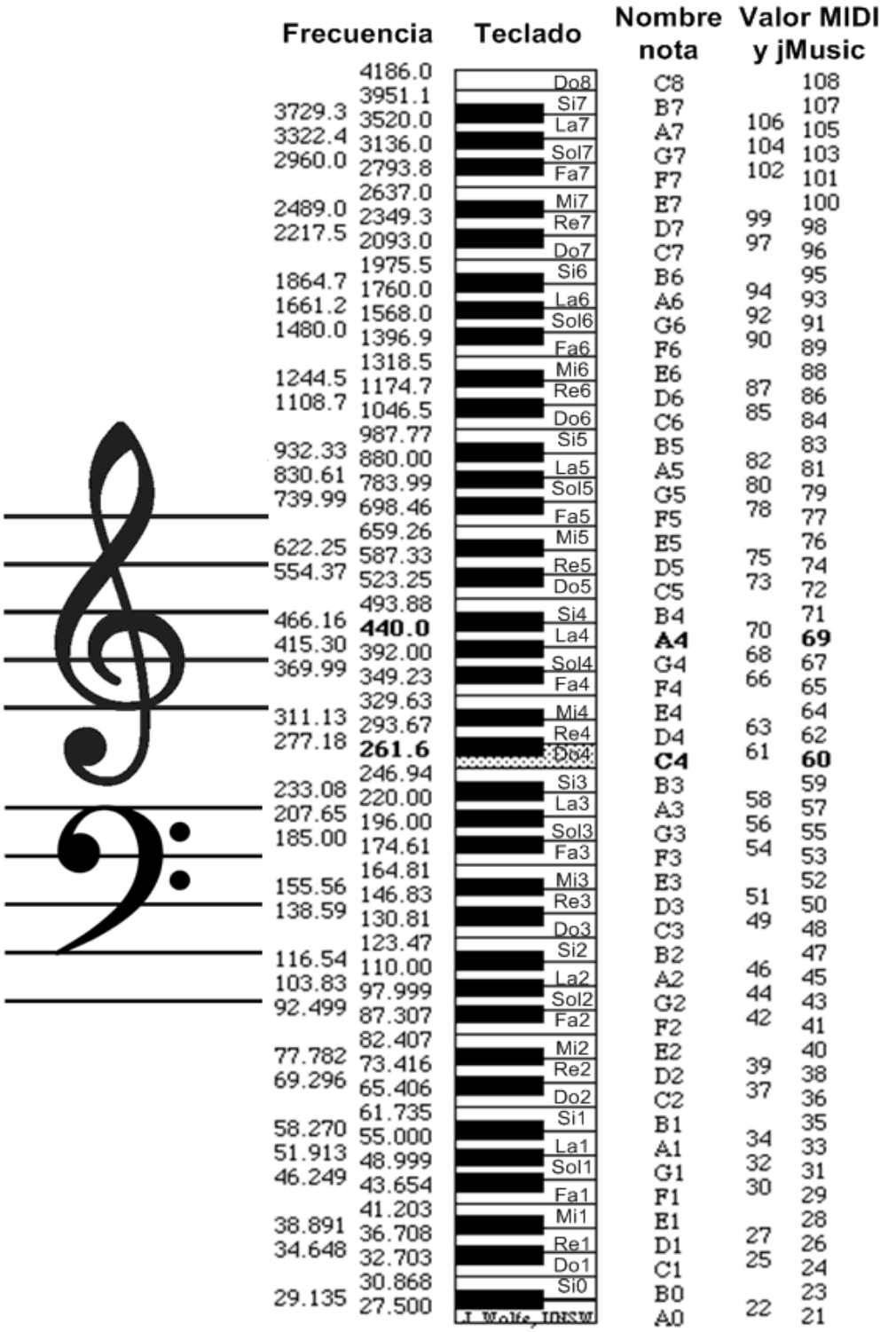


Figura 62: Correspondencia entre las notas musicales y los valores de tonos empleados en jMusic.

El silencio musical es un tono especial en jMusic. La constante para la ausencia de sonido es REST, codificado como el número entero más bajo negativo, -2147483648, e indica que no debe existir sonido de ninguna nota.

B)Figuras de duración o rítmicas (Rhythmic Value)

jMusic está basada en pulsos de valor 1.0, que es el correspondiente a la figura de una negra. El resto de ritmos son relativos a este valor.

Las constantes que representan las figuras rítmicas pueden tener varios nombres para denominar una misma duración. En general, se puede emplear el nombre británico, el americano y sus abreviaturas.

Tabla 12: Valores de las figuras de duración en jMusic y nombres para referenciarlos.

Valor		Nombre Español	Nombre Británico	Nombre Americano	Abreviat.
0,0833		Fusa en Tresillo	DEMI_SEMI_QUAVER_TRIPLET	THIRTYSECOND_NOTE_TRIPLET	DSQT, TSNT
0,125		Fusa	DEMI_SEMI_QUAVER	THIRTYSECOND_NOTE	DSQ, TSN
0,1667		Semicorchea en Tresillo	SEMI_QUAVER_TRIPLET	SIXTEENTH_NOTE_TRIPLET	SQT, SNT
0,25		Semicorchea	SEMI_QUAVER	SIXTEENTH_NOTE	SQ, SN
0,3333		Corchea en Tresillo	QUAVER_TRIPLET	EIGHTH_NOTE_TRIPLET	QT, ENT
0,375		Semicorchea con Puntillo	DOTTED_SEMI_QUAVER	DOTTED_SIXTEENTH_NOTE	SQD, DSN
0,5		Corchea	QUAVER	EIGHTH_NOTE	Q, EN
0,6667		Negra en Tresillo	CROTCHET_TRIPLET	QUARTER_NOTE_TRIPLET	CT, QNT
0,75		Corchea con Puntillo	DOTTED_QUAVER	DOTTED_EIGHTH_NOTE	DQ, DEN
0,875		Corchea con Doble Puntillo	DOUBLE_DOTTED_QUAVER	DOUBLE_DOTTED_EIGHTH_NOTE	DDQ, DDEN
1		Negra	CROTCHET	QUARTER_NOTE	C, QN
1,3333		Blanca en Tresillo	MINIM_TRIPLET	HALF_NOTE_TRIPLET	MT, HNT
1,5		Negra con Puntillo	DOTTED_CROTCHET	DOTTED_QUARTER_NOTE	DC, DQN
1,75		Negra con Doble Puntillo	DOUBLE_DOTTED_CROTCHET	DOUBLE_DOTTED_QUARTER_NOTE	DDC, DDQN
2		Blanca	MINIM	HALF_NOTE	M, HN
3		Blanca con Puntillo	DOTTED_MINIM	DOTTED_HALF_NOTE	DM, DHN
3,5		Blanca con Doble Puntillo	DOUBLE_DOTTED_MINIM	DOUBLE_DOTTED_HALF_NOTE	DDM,D DHN
4		Redonda	SEMIBREVE	WHOLE_NOTE	SB, WN

C)Dinámica (Dynamic)

La dinámica es una magnitud proporcional a la intensidad en la música. En jMusic se emplean al menos ocho niveles de dinámica cuantitativa como aproximación a la clásica cualitativa. Desde el sonido muy suave hasta el sonido muy fuerte, permite establecer la dinámica mediante un valor entero entre 0 y 127; y ofrece una serie de constantes que especifican los ocho niveles habituales antes citados.

Tabla 13: Constantes de dinámica.

Abreviatura	Valor
ppp	10
pp	25
p	50
mp	60
mf	70
f	85
ff	100
fff	120
Silencio	0

D)Otras constantes

Además de las constantes habituales, detalladas anteriormente, existen otras que se pueden emplear en funcionalidades musicales:

- ***Panoramización*** (*Panning*): es la posición de una señal monoaural en el canal estéreo. El panning define la panorámica de la imagen sonora estéreo: todo el sonido en un solo canal (izquierdo o derecho) o en ambos canales a la vez, centrada.

- ***Duración Articulada*** (*Duration Articulation*): por defecto, el tiempo de duración de las notas es el 85% del valor de la figura. No obstante, se puede especificar una duración diferente multiplicando el valor de la figura por una constante.

- ***Timbre*** *(ídem)*: de acuerdo con el estándar MIDI, se asigna un número en el rango de 1 a 127 para definir los distintos instrumentos o voces.

- ***Escala*** (*Scale*): define las escalas musicales más habituales.

5.1.1.2.*Implementación de procesos del módulo de extracción*

I. Proceso de análisis melódico

Durante este primer proceso, el método *analiza()* se encarga de capturar la información textual descriptiva introducida por el usuario (metadatos); y sobre todo, de obtener una representación del audio MIDI en la estructura jMusic, consistente en agrupaciones de objetos *Note* (notas musicales). El método implementado trasladará estas notas de la estructura jerárquica a una estructura lineal donde las notas formarán una serie, el vector *NoteList*.

Es necesario para los fines descritos acceder al objeto *Score* (partitura) a fin de extraer los objetos *Part* incluidos dentro de ella. A su vez se toman de éstos los objetos *Phrase*, que finalmente contienen los *Note*.

```
void analiza(){

        Read.midi(partitura, file[i].getAbsolutePath());

        Enumeration enum1= partitura.getPartList().elements();
        //enumeración de las partes

        while(enum1.hasMoreElements()){

          Part nextPt = (Part)enum1.nextElement();

          Enumeration enum2 = nextPt.getPhraseList().elements();
           //enumeración de las frases

                  while(enum2.hasMoreElements()){

                    Phrase nextPhr = (Phrase)enum2.nextElement();

                    Enumeration enum3 = nextPhr.getNoteList().elements();
                     //enumeración de las notas

                            while(enum3.hasMoreElements()){

                              Note nextNote = (Note)enum3.nextElement();

                              noteList.add(nextNote);

                            }
                  }
        } (…)
}
```

Figura 63: Método de análisis melódico: lectura y serialización.

A continuación, de entre todos los atributos de cada objeto *Note* sólo se tomarán los relativos al tono (*Pitch*) y a la duración (*RythmValue*).

```
for(int j=0; j< contadorNotas; j++){

  notaJ= (Note)(noteList.elementAt(j));

  tonos[j]= notaJ.getPitch();
  duraciones[j]= notaJ.getRhythmValue();
}
```

Figura 64: Almacenamiento de tonos y duraciones para cada nota de la melodía.

Tras finalizar este proceso, ya se dispone de los valores de las notas (tonos y duraciones) que constituirán el campo de información de la melodía en su notación absoluta (no relativa), el campo denominado como *melodía_abs*. La única salvedad la constituyen los silencios; como ya se ha expuesto con anterioridad, el valor del tono de un silencio es el menor de los números enteros negativos (para 32 bits: -2147483648), y consignar este valor proporcionaría una representación irreal (no proporciona la idea de una ausencia de sonido, sino de una bajada superlativa del tono) a la par que ineficiente. Por este motivo, los valores del tono para los silencios se cambiarán por una cadena vacía en el paso previo a la generación del documento de salida, momento en el cual toda la información es gestionada como cadenas de texto (*String*).

II. Proceso de normalización

Para ejecutar este proceso se recurre al método *normaliza()*, éste se encarga de realizar las operaciones necesarias con el objetivo de obtener las representaciones que se emplearán posteriormente en la indexación.

En el proceso anterior, la obtención de los valores para la melodía en notación absoluta es prácticamente inmediata. Por el contrario, el método de representación principal de este sistema, la notación relativa, requiere de los correspondientes cálculos entre los valores de cada nota y los de su precedente, que serán realizados en este proceso.

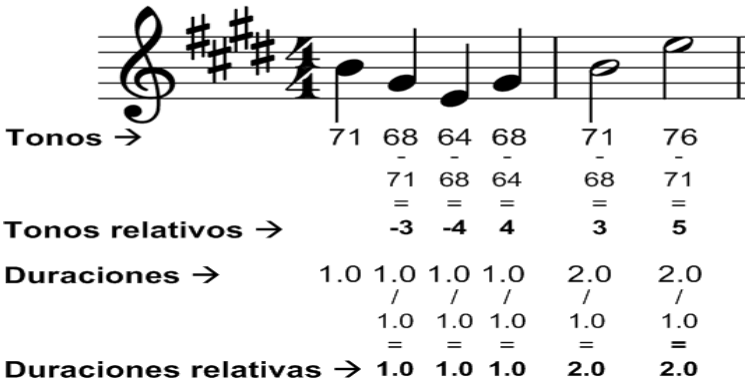


Figura 65: Obtención de tonos y duraciones relativas de las notas

De nuevo, es necesario contemplar el caso especial de los silencios. Su valor de tono no debe ser tenido en cuenta para los cálculos de tono relativo de las notas. Dos son las razones en este caso para decidirlo: originaría diferencias irreales, por los motivos que ya se han explicado, y por otro lado, las operaciones con estos valores resultarían con demasiada frecuencia en desbordamiento del rango permitido para los enteros. En el espacio que se destina al tono relativo para esta nota se emplazará un cadena vacía.

En relación a los silencios, también se decidió en el planteamiento del diseño proporcionar la posibilidad de realizar búsquedas que los omitieran. La notación relativa queda afectada cuando tal hecho sucede, y por este motivo se decidió que esta representación sin silencios tuviera cabida en un campo aparte. Para obtener esta serie de notas alternativas es necesario realizar un cálculo paralelo que queda detenido cuando se encuentra un silencio, hasta que exista una nota sonora, y cuyos resultados divergirán de la secuencia de valores relativos completa (cuando existan uno o más silencios tendrá diferentes valores y la cantidad de elementos será menor).

Como se puede observar en la figura 66, los valores de la secuencia relativa completa se almacenan directamente en formato textual en el array *notasNormalizadas* (columnas 0 y 1).

```
for(int x=1,y=1;x<tam;x++){

   if(tono[x]>=0){   //Si no es silencio
        tonoNormalizado=Integer.toString(tono[x]-tono[x-1]);
        duraNormalizada=Double.toString(duracion[x]/duracion[x-1]);
        (...)

    }else{
        tonoNormalizado="  ";
        tono[x]=tono[x-1];  //No se anota si es un silencio: se provoca que se tome en el tono
                            //siguiente la altura relativa a la anterior nota (la que no era silencio).
        duraNormalizada=Double.toString(duracion[x]/duracion[x-1]);
         }
    notasNormalizadas[x+1][0]= tonoNormalizado ;
    notasNormalizadas[x+1][1]= duraNormalizada ;
}
```

Figura 66: Código de normalización para la obtención de representaciones melódicas relativas.

A fin de almacenar todos los valores requeridos se elabora una sencilla estructura matricial, un array denominado *notasNormalizadas*. Éste contendrá una columna para los valores de tonos relativos y otra para las duraciones relativas, así como otras dos columnas que albergarán los tonos y duraciones relativas de la representación sin silencios. El espacio ocupado en estas dos últimas columnas será

el mismo que en las dos primeras cuando el fragmento no contenga silencios (porque, de hecho, la representación será idéntica), pero será menor cuantos más silencios contenga. Por este motivo, se reserva la primera posición de cada columna en el array para indicar la longitud de cada serie de valores.

A partir de la estructura *notasNormalizadas* se obtendrán los valores necesarios para todos los campos de representación melódica restantes:

- Melodía en notación relativa (*melodía*)
- Melodía en notación relativa sin silencios (*melodía_sin*)
- Melodía en notación relativa de duraciones, independiente de alturas o tonos (*melodía_st*)
- Melodía en notación relativa de tonos, independiente de duraciones (*melodía_sd*).

Los valores para el campo de melodía en notación absoluta (*melodía_abs*), como ya se ha mencionado, se encuentran disponibles tras la serialización ejecutada en el método de análisis, concretamente en los arrays de *tonos[]* y *duraciones[]*.

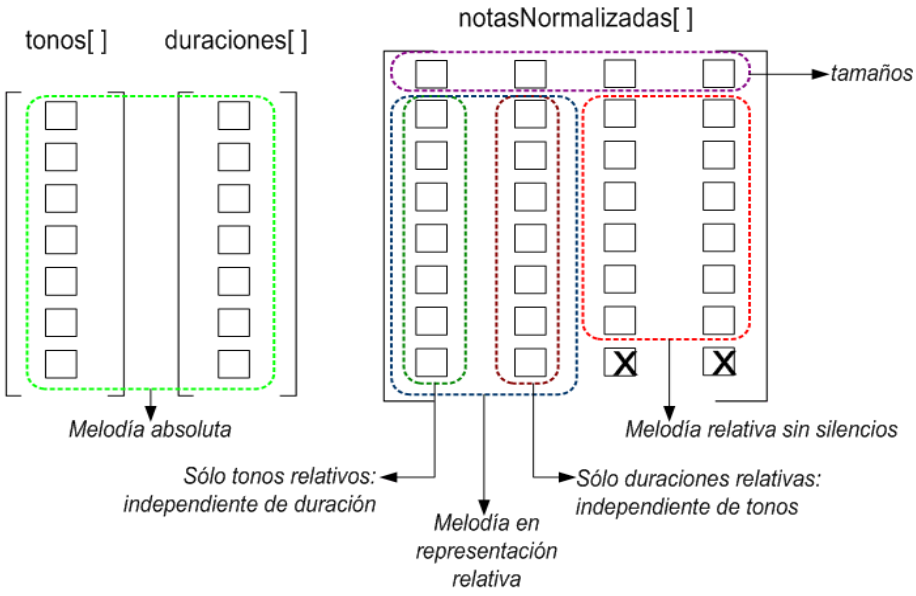


Figura 67: Extracción de los campos de información melódica.

Como es lógico, la notación relativa se puede aplicar a todas las notas salvo a la primera. Existe la opción de omitir esta nota, sin embargo se ha estimado conveniente mantenerla con su notación original (el tono y la duración absolutos) a

pesar de que las demás se encuentren en una notación relativa, para una mayor consistencia. De este modo, con una decisión sin apenas coste computacional, en cualquier momento se puede obtener la melodía original, partiendo de la primera y realizando los cálculos necesarios sin tener que consultar el campo melodía absoluta.

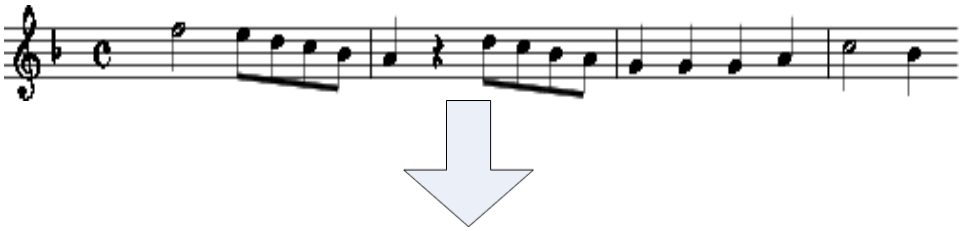
En los campos de información melódica que contienen tonos y duraciones (*melodía*, *melodía_abs* y *melodía_sin*), se ha concebido, por motivos de eficiencia, que dichos valores tono y duración de cada nota aparezcan juntos. Por cada nota se encontrará en primer lugar el valor del tono relativo y a continuación, con la separación de un espacio en blanco, el valor de la duración relativa. Para permitir una rápida diferenciación en la búsqueda, el valor de tono estará precedido por el carácter *T* y el de la duración por el carácter *D*. Es la forma que va a permitir un tratamiento más cercano a la fuente original (la partitura), aprovechando al máximo la potencialidad de un sistema de indexación textual.

Tras la finalización de los procesos descritos, el programa abre un flujo de escritura para generar el documento XML representativo de la melodía. Este documento se denominará de acuerdo al nombre del archivo MIDI originario. En la figura 69, se recoge un ejemplo de documento generado para una melodía concreta, de acuerdo al DTD común detallado en la figura 68.

```
<!ELEMENT cancion (autor, titulo, categoría, letra, parson, melodía, melodia_abs,
melodia_st, melodia_sd, melodia_sin)>

<!ELEMENT autor (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT categoria (#PCDATA)>
<!ELEMENT letra (#PCDATA)>
<!ELEMENT parson (#PCDATA)>
<!ELEMENT melodia (#PCDATA)>
<!ELEMENT melodia_abs (#PCDATA)>
<!ELEMENT melodia_st (#PCDATA)>
<!ELEMENT melodia_sd (#PCDATA)>
<!ELEMENT melodia_sin (#PCDATA)>
```

Figura 68: DTD común para la generación de los documentos XML.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cancion SYSTEM "musica.dtd">
<cancion>
        <autor>Beethoven, Ludwig van</autor>
        <titulo>piano concerto 2 in Bb opus 19, 1st movement, 2nd theme</titulo>
        <categoria>Classical</categoria>
        <letra></letra>
        <parson>DDDDDUDDDDRRUUD</parson>
        <melodia> T77 D2.0 T-1 D0.25 T-2 D1.0 T-2 D1.0 T-2 D1.0 T-1 D2.0 T   D1.0 T5
                  D0.5 T-2 D1.0 T-2 D1.0 T-1 D1.0 T-2 D2.0 T0 D1.0 T0 D1.0 T2 D1.0 T3
                  D2.0 T-2 D0.5
        </melodia>
        <melodia_abs> T77 D2.0 T76 D0.5 T74 D0.5 T72 D0.5 T70 D0.5 T69 D1.0 T  D1.0
                      T74 D0.5 T72 D0.5 T70 D0.5 T69 D0.5 T67 D1.0 T67 D1.0 T67
                      D1.0 T69 D1.0 T72 D2.0 T70 D1.0
        </melodia_abs>
        <melodia_st> D2.0 D0.25 D1.0 D1.0 D1.0 D2.0 D1.0 D0.5 D1.0 D1.0 D1.0 D2.0
                     D1.0 D1.0 D1.0 D2.0 D0.5
        </melodia_st>
        <melodia_sd> T77 T-1 T-2 T-2 T-2 T-1 T   T5 T-2 T-2 T-1 T-2 T0 T0 T2 T3 T-2
        </melodia_sd>
        <melodia_sin> T77 D2.0 T-1 D0.25 T-2 D1.0 T-2 D1.0 T-2 D1.0 T-1 D2.0 T5 D0.5 T-
                      2 D1.0 T-2 D1.0 T-1 D1.0 T-2 D2.0 T0 D1.0 T0 D1.0 T2 D1.0 T3
                      D2.0 T-2 D0.5
        </melodia_sin>
</cancion>
```

Figura 69: Ejemplo de documento XML generado para una melodía.

5.1.2. Módulos Constructor e Indexador

Los módulos Constructor e Indexador se constituyen en una única clase Java de nombre *IndexXML*. Los objetos de esta clase implementarán el proceso de construcción documental con la ayuda de un *parser* del paquete Xerces [29] basado en la API de SAX [45] (mencionado en el apartado 3.4.3.1.B), que permitirá mediante la clase interna *MelodyHandler* leer los documentos generados en XML. Para los procesos de análisis y de agregación al índice se requerirán una serie de clases de la librería Lucene.

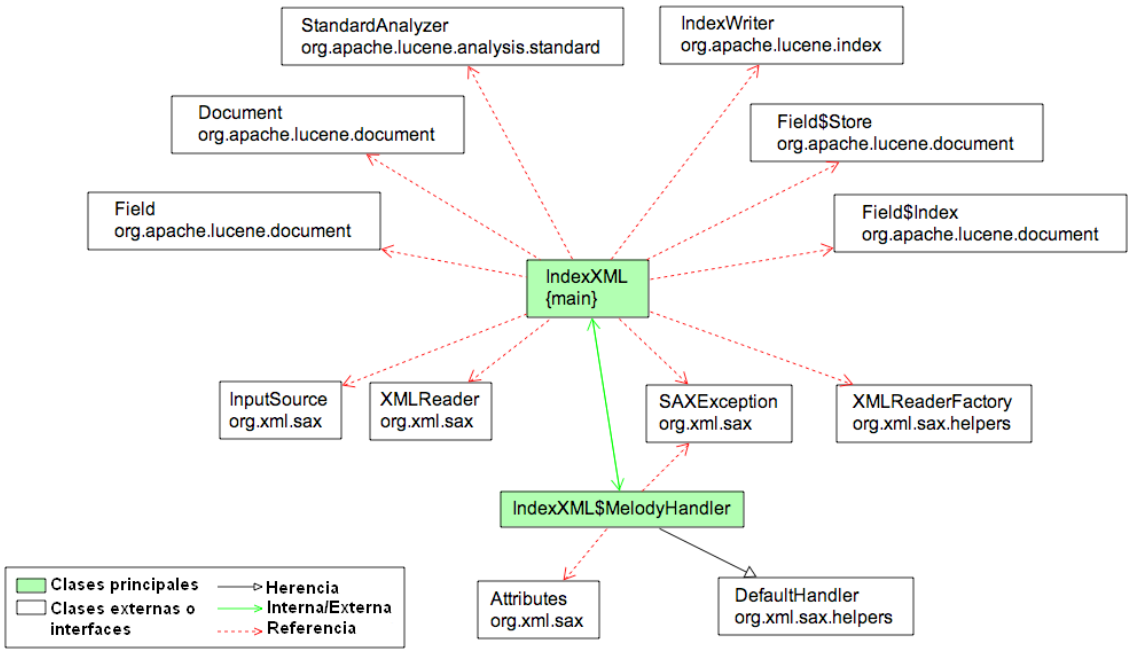


Figura 70: Diagrama de clases del módulo Constructor e Indexador.

5.1.2.1.*Clases principales de la librería Lucene para la indexación*

En un procedimiento de indexación mediante los métodos de la API Lucene intervienen las siguientes clases:

- *IndexWriter*
- *Directory*
- *Analyzer*
- *Document*
- *Field*

A continuación se detalla la función de cada una de estas clases, cuya intervención en el proceso se describe en la figura 71.

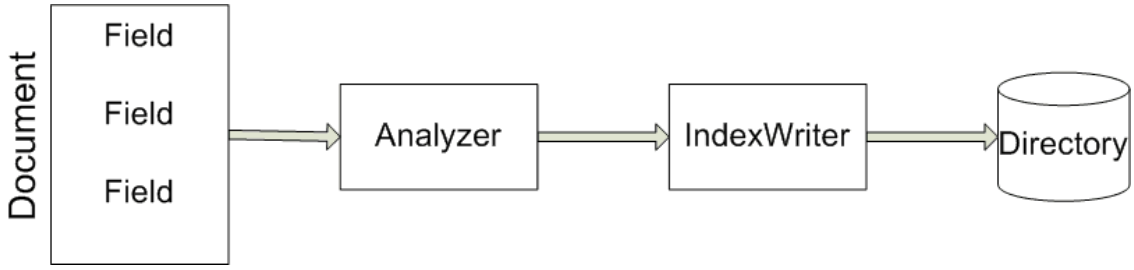


Figura 71: Clases participantes en el proceso de indexación.

A)IndexWriter

Mediante esta clase se crea un nuevo índice o se abre uno ya existente, para a continuación añadir, actualizar o eliminar documentos en él. Los objetos de la clase *IndexWriter* tiene acceso a los índices para su escritura, pero no permiten leer ni hacer búsquedas. El índice sobre el que escriben ha de ser guardado en un lugar físico, de ahí surge la necesidad de la clase *Directory.*

B)Directory

La clase abstracta *Directory* representa la localización de un índice Lucene. Las subclases que heredan de ella permitirán almacenar el índice de la forma concreta que se juzgue más conveniente. En concreto, la subclase *FSDirectory,* almacena archivos reales en el sistema de ficheros. Mientras que la subclase *RAMDirectory* se emplea para mantener todos los datos en memoria principal y destruirlos tras el término de la aplicación. Esta última opción se empleará cuando se requiera un rendimiento óptimo en cuanto a rapidez, tanto para indexar como para buscar (es decir, tanto para escribir como para leer del índice), ya que se garantiza un acceso más rápido que el de disco; lógicamente el tamaño del índice ha de ser tan pequeño como para permitirlo.

El objeto de la clase *Directory*, cualquiera que sea la clase hija a la que pertenezca, se debe introducir como argumento en el constructor de *IndexWriter.*

C)Analyzer

El objeto *IndexWriter* escribirá un texto en el índice, pero previamente éste debe separarse en diferentes términos. Por ello, el texto de un documento debe ser analizado por un objeto de la clase *Analyzer* antes de ser indexado, el cual se encargará de extraer los términos que han de ser indexados y eliminará el resto. Al tratarse *Analyzer* de una clase abstracta, sus diferentes implementaciones son

realmente subclases de ella. En la API de Lucene existen varios tipos de analizadores, constituidos como subclases, que pretenden adecuarse en cada caso a diferentes escenarios. La mayoría de ellos separan los términos entre espacios en blanco; un buen número de ellos filtran las *stop words* o palabras irrelevantes (que no ayudan a distinguir entre un documento y otro), tales como los determinantes, preposiciones y conjunciones en inglés; y también es habitual la conversión de términos a letra minúscula, para conseguir que en las búsquedas no se distinga entre mayúsculas y minúsculas. Debido a su impacto en la fase de búsqueda, el tipo de analizador a emplear es habitualmente un factor clave en el diseño.

El análisis toma como entrada un objeto documento (*Document*) constituido por una serie de campos (*Field*) a indexar.

D)Document

Un objeto *Document* modela un conjunto de campos, los cuales pueden representar en formato textual el contenido o los metadatos de un documento físico, sea cual sea su naturaleza original (cabe recordar que incluso un archivo multimedia se traduce de algún modo a información textual). Los campos, de la clase *Field*, se deben añadir al *Document* que será la unidad referenciada en el índice.

E)Field

Los campos contenidos en una unidad documento, constituidos siempre como objetos de la clase *Field*, cuentan con un nombre identificativo, su valor en cadena de texto que corresponda y una serie de opciones que controlan cómo se debe indexar la información del campo. Los campos de un mismo documento pueden crearse con nombres repetidos, en ese caso, al realizar la indexación, se irán concatenando los valores de los campos en el mismo orden en que se fueron añadiendo a la unidad documento.

5.1.2.2.*Implementación de procesos del módulo de construcción e indexación*

I. Proceso de construcción

I.1. Lectura de documentos XML

En un primer término es necesario adquirir la información extraída en el módulo previo, que se encontrará almacenada en formato XML. Para ello se implementará un clase interna *MelodyHandler*, que define el proceso de manejo de la información y que será empleada por un objeto *parser* en la clase principal para obtener los valores de cada campo.

```
static class MelodyHandler extends org.xml.sax.helpers.DefaultHandler {
        (...)

   public void startElement(String namespaceURI,
                String localName, String qualifiedName, Attributes atts) {

     if (qualifiedName.equals("autor")) {
        buffer = new StringBuffer();
     }
      (...)
  }

   public void endElement(String namespaceURI, String localName,
                String qualifiedName) {

     if (qualifiedName.equals("autor")) {
        autor = buffer.toString();
        buffer = null;
     }
      (...)
  }

   public void characters(char[] text, int start, int length)
   throws SAXException {

     if (buffer != null) {
        buffer.append(text, start, length);
     }
      (...)
  }
(...)
}
```

Figura 72: Extractos de código de la clase interna *MelodyHandler*.

En la clase principal, un objeto *parser* (instancia de la clase externa *XMLReader* proporcionada por el paquete *org.apache.xerces)* empleará un objeto de la clase *melodyHandler* como gestor de contenido (*ContentHandler*). Tal hecho es lo que permitirá que se puedan obtener los elementos almacenados que constituirán la información de los campos.

```
XMLReader parser =
    XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser");

            MelodyHandler handler = new MelodyHandler();

            parser.setContentHandler(handler);

            InputStream in = new FileInputStream(fDoc);
            InputSource source = new InputSource(in);
            parser.parse(source);

            in.close();
```

Figura 73: Creación de un *parser* para la lectura de un documento.

I.2. Creación de campos del documento

Como ya se ha expuesto con antelación, todos los elementos con contenido creados en el documento XML, se constituirán en campos de información de la unidad documento a indexar. Además, se requiere añadir un campo para el nombre de archivo de audio, que será útil en la fase de búsqueda (por ejemplo para insertar hiperenlaces). La forma de obtención de este último es trivial: se toma el nombre del archivo a partir del nombre del propio documento XML.

En la tabla 14, se recogen los once campos de los que constará una unidad documento. En el momento de su creación, se señala si un campo debe ser analizado para ser separado en términos (*tokens*) de cara a la indexación, así se indica en la tercera columna de la citada tabla.

Tabla 14: Campos de una unidad documento en el sistema.

Nombre de campo	Definición	Analizable
nombre	Nombre del archivo de audio musical	No
autor	Autor	Sí
titulo	Título	Sí
categoria	Categoría o género	Sí
letra	Letra o descripción	Sí
parson	Código de contorno melódico de Parsons	Sí
melodia	Representación relativa de tonos y duraciones de la melodía	Sí
melodia_abs	Representación absoluta de tonos y duraciones.	Sí
melodia_st	Representación relativa de duraciones.	Sí
melodia_sd	Representación relativa de tonos.	Sí
melodia_sin	Representación relativa de tonos y duraciones, omitidos silencios.	Sí

A través de los atributos del objeto *handler* se accederá al contenido de los campos extraídos del XML.

```
String campoMelodia=handler.melodia;
```

Figura 74: Obtención de cada campo mediante atributos del *handler*.

Tras la obtención del contenido, es necesario crear los objetos que modelan los campos (*Field*). En la misma operación se ha de indicar el contenido junto con el nombre del campo que aparecerá en el índice, así como si se le aplica el análisis para dividirlo en términos (campo *tokenized*).

En la fase de búsqueda, se aplican unos criterios muy definidos para establecer la relevancia de unos resultados frente a otros. Uno de estos criterios está influido precisamente por un posible factor de ponderación o de realce (*boost*) que puede afectar a un campo concreto, de forma que se le pueda conceder mayor o menor importancia a las coincidencias con sus términos en el cómputo total. Se mantendrá el factor por defecto, 1, en todos los campos de metadatos; y dada su importancia en el planteamiento del sistema, se aplicará un factor de 2 sobre todos los campos de representación melódica (*melodia*, *melodia_abs*, etc.)

```
Field md=new Field("melodia", campoMelodia, Field.Store.YES,
                                Field.Index.TOKENIZED);

                md.setBoost(2.0f);
```

Figura 75: Creación de un campo y aplicación de un factor de realce.

I.3. Creación de unidades documento

Cada unidad documento corresponderá unívocamente a una composición musical, por tanto se creará una por cada entrada en el sistema. Una vez creado un objeto *Document*, es posible añadirle cualquier campo, Lucene no impone ninguna restricción al respecto; las restricciones se establecen de acuerdo al diseño de la aplicación, como se describen en el siguiente punto.

I.4. Inserción de campos del documento

En un mismo índice de Lucene se pueden incluir documentos con campos totalmente diferentes, en cuanto a su nomenclatura y su definición. Esta característica es muy útil en documentos de texto lingüístico, pero no en sistemas de texto representativo como es este caso. En este diseño se han estipulado uno campos concretos para aportar homogeneidad al sistema (*autor*, *título*, *melodía*, etc.), y son los únicos que se podrán agregar a las correspondientes unidades

documento. De forma consecutiva, se agregan sobre cada objeto *Document*, los objetos *Field* definidos.

II. Proceso de análisis

El proceso de análisis afectará a todos los campos que hayan sido declarados como *tokenized* y conllevará su división en términos. Como ya se ha indicado, éste será el caso de todos los campos salvo el de *nombre*.

La división puede realizarse atendiendo a diferentes criterios según el tipo de analizador que sea definido. Existe una limitación a tener en cuenta consistente en que el analizador se asocia al escritor del índice (*IndexWriter*), de forma que el mismo analizador se aplica a todos los documentos que éste agrega al índice, y por tanto todos los campos serán analizados y divididos bajo un mismo criterio. A consecuencia de ello, se debe definir un tipo de analizador cuyo desempeño sea óptimo para todos los campos. La API Lucene ofrece varios posibilidades:

Tabla 15: Clases de analizadores disponibles en Lucene.

Analizador	Efectos
WhiteSpaceAnalyzer	Separa el texto entre espacios en blanco.
SimpleAnalyzer	Separa el texto en torno a caracteres que no sean letras (eliminando tales caracteres) y normaliza todos los términos resultantes a letras minúsculas.
StopAnalyzer	Separa el texto en torno a caracteres que no sean letras (eliminando tales caracteres), normaliza todo el texto a letras minúsculas y elimina las *stop words* (como por ejemplo los determinantes, conjunciones, etc. citados en el apartado 5.1.2.1.C)
StandardAnalyzer	Separación basada en un gramática propia compleja que, entre otras cosas, permite mantener íntegras direcciones de email, acrónimos, palabras apostrofadas, palabras con caracteres especiales en diversas lenguas, etc. Además aplica una normalización a letras minúsculas y elimina *stop words*.

Tras una comparación no sólo teórica, sino también empírica, se decidió emplear el *StandardAnalyzer;* por ofrecer un mejor tratamiento del texto contenido en los campos de metadatos y un tratamiento correcto de las series numéricas contenidas en los campos melódicos. En el apartado 5.1.3., relativo al índice, se puede observar una captura de pantalla (figura 79) que permite visualizar algunos de los términos presentes en el índice, los cuales fueron creados tras la aplicación del analizador.

```
Analyzer a = new StandardAnalyzer();
IndexWriter writer;

try {
   writer = new IndexWriter(index, a, false);
}
    catch (Exception e) {
      writer = new IndexWriter(index, a, true);
    }
```

Figura 76: Código que asocia el proceso de escritura en el índice con el *StandardAnalyzer*.

III. Proceso de indexación

Tras la ejecución del análisis, el contenido está dividido en términos para aquellos campos en que así fuera procedente, y en su totalidad listo para ser agregado al índice. El procedimiento de indexación en esencia consistirá en tomar las unidades mínimas de contenido -términos o, en el caso de campos no analizados, el campo completo- e insertar cada una de ellas como una entrada en el índice. Se referenciará el documento en el cual se encuentra y la posición dentro de él.

El índice es una estructura en realidad ligeramente compleja, cuyo único objetivo es optimizar la fase de búsqueda. Además de la información básica antes citada, en el proceso de indexación se llevan a cabo múltiples operaciones para consignar diversos datos por cada término en el índice. Por ejemplo, se incluye la frecuencia del término en el documento, un factor de normalización del término de acuerdo a la longitud total del campo, el factor de *boost* (realce) para el campo del término, etc.

Una vez terminado el proceso de indexación para todos los documentos, se ejecuta una operación de optimización, proporcionada por el método *optimize()* de *IndexWriter*, que proporcionará al índice un formato más efectivo. Como último paso, es necesario cerrar el *IndexWriter* con la ejecución del método *close()*.

```
writer.addDocument(nDoc);
        (...)
//Tras agregar todos los documentos al índice:
writer.optimize();
writer.close();
```

Figura 77: Operación de escritura de documentos en el índice.

5.1.3. Índice

La localización de un índice se modela desde la clase abstracta *Directory*, y se implementa concretamente como un objeto de la subclase *FSDirectory* o *RAMDirectory*. En este caso, se ha decidido que sea *FSDirectory*, por tanto el índice se albergará como un conjunto de archivos en el sistema de ficheros.

El índice empleado por este sistema, al igual que cualquiera que se encuentre implementado con esta API, no es por principio una estructura monolítica. Un índice esta formado por una o varias subunidades llamadas segmentos, y cada uno de ellas contiene el resultado de la indexación de un subconjunto de documentos. Los segmentos se almacenan como archivos individuales en el sistema de ficheros, con una nomenclatura característica: comienza con el carácter "_" y su extensión es ".cfs".

Durante el proceso de indexación de documentos, la información se agrega al índice de forma incremental, en concreto se suelen generar varios segmentos y la adición de documentos se hará de forma alternada sobre ellos. Éste es un principio de diseño de los métodos de indexación ofrecidos por Lucene, con el objetivo de minimizar el impacto de las repetidas modificaciones físicas del índice. Sin embargo, al finalizar de añadir todos los documentos, se puede aplicar una optimización sobre el índice, como de hecho se realiza en este sistema (método *optimize()*), con el fin de obtener una estructura más eficiente para la búsqueda. Tras esta optimización final, sólo los índices que contienen un número elevado de documentos se reparten en varios segmentos. En la gran mayoría de las pruebas realizadas con esta aplicación se generó un único segmento.

La existencia o no de segmentos en el sistema y sus respectivos ficheros asociados, son especificados en los ficheros *segments.*

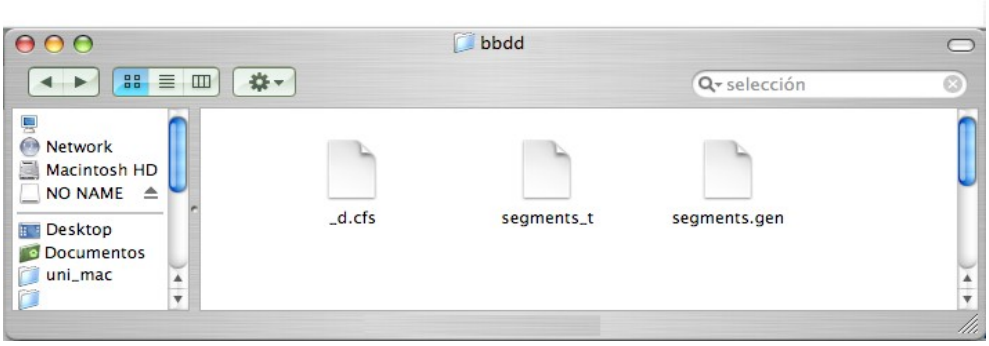


Figura 78: Archivos del índice de la aplicación en el sistema de ficheros.

En la figura 79, es posible visualizar el contenido del índice tal y como lo ofrece la aplicación Luke [34] (herramienta elaborada como parte del desarrollo adicional colaborativo de Lucene).

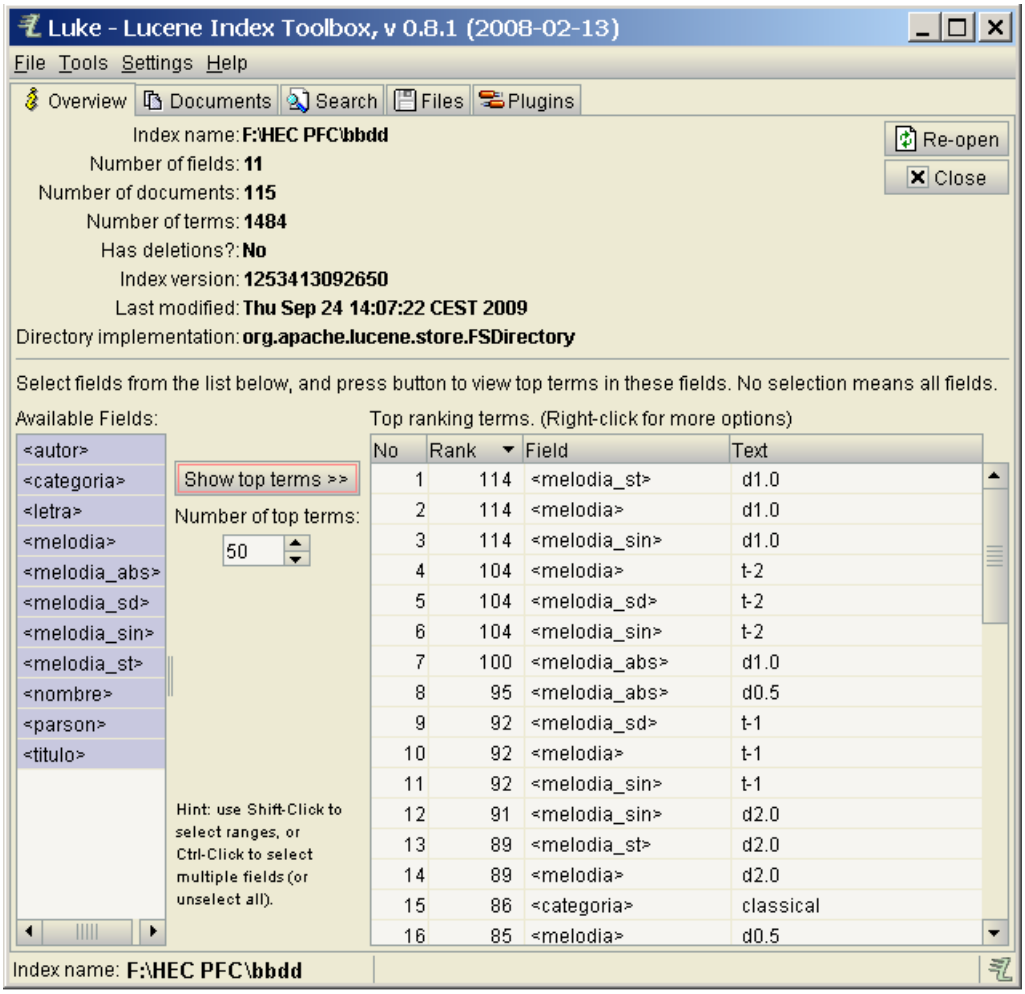


Figura 79: Captura de pantalla de la aplicación Luke - Lucene que permite visualizar el contendido del índice del sistema.

5.2. Bloque de aplicación de búsqueda

5.2.1. Módulo Buscador

El módulo se implementa en una clase Java denominada *Buscador*. Los objetos de esta clase ejecutarán tres procesos definidos: normalización, elaboración de consultas y ejecución de consultas. Para el proceso de normalización se recurrirá en primer lugar a determinados métodos de la clase *Traductor,* que permitirán traducir la consulta melódica del usuario. Posteriormente, para los procesos de consulta se requerirá emplear una serie de clases de la librería Lucene.

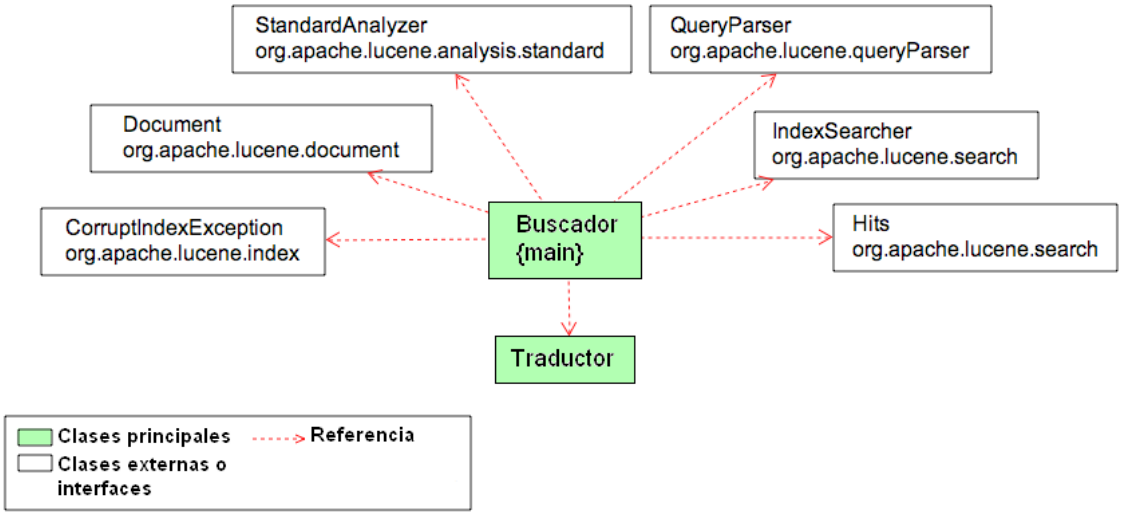


Figura 80: Diagrama de clases del módulo Buscador.

5.2.1.1.*Clases principales de la librería Lucene para la búsqueda*

El proceso de búsqueda se puede manejar de forma eficiente con un número reducido de clases de la API Lucene. Éstas son principalmente:

- *IndexSearcher*
- *Term*
- *Query*
- *TermQuery*
- *Hits*

A)IndexSearcher

El objeto *IndexSearcher* tiene una función dual a la del *IndexWriter* de la fase de indexado: es el encargado de servir de enlace con el índice para buscar en él mediante métodos. La clase *IndexSearcher* tiene por objetivo abrir un índice en modo de sólo lectura y ofrece una serie de métodos de búsqueda, en su mayoría ya implementados en su clase madre abstracta S*earcher*.

B)Term

Un término, *Term*, es la unidad básica de búsqueda. De un modo simétrico a como se constituían los objetos *Field*, un objeto *Term* se compone de dos elementos *strings o* cadenas de texto, uno para detallar el nombre del campo en el que se desea buscar, y el otro para el término que se desea encontrar en él. Mediante un objeto *Term* se puede construir un objeto de la clase consulta: *Query.*

C)Query

Query es una clase abstracta, a partir de la cual la API de Lucene recoge varias subclases que corresponden a diferentes tipos de consultas. Una de las más conocidas es *TermQuery,* pero no menos importantes son *BooleanQuery*, *PhraseQuery*, *PrefixQuery*, *PhrasePrefixQuery*, *RangeQuery*, *FilteredQuery* y *SpanQuery*.

D)TermQuery

Como ya se ha adelantado, esta es una de las clases hijas de *Query* más empleadas. Sirve para realizar uno de los tipos de consulta más sencillos: encontrar los documentos que contienen un término, un valor determinado en un campo concreto.

E)Hits

La clase *Hits* se constituye como un contenedor de punteros para ordenar los resultados de la búsqueda (los documentos coincidentes con la consulta) de acuerdo a la puntuación otorgada. Por razones de rendimiento, las instancias de *Hits* no cargan todos los documentos coincidentes del índice, sólo una pequeña porción.

5.2.1.2.*Implementación de procesos del módulo Buscador*

I. Proceso de normalización

Se implementa principalmente dentro del método *normaliza()*, cuya función es equivalente a la de su homónimo en el módulo de extracción. Su diferencia fundamental reside en el origen de los datos melódicos, que en este caso se introducen en el sistema en formato textual, como resultado de una consulta de usuario. Por otro lado, si bien el método permite proporcionar todas las representaciones melódicas, tanto en forma relativa como absoluta, únicamente se obtiene la representación correspondiente a la opción de búsqueda que el usuario ha seleccionado (melodía absoluta para búsquedas exactas, o bien melodía relativa, sin silencios, sin tonos, etc.).

Las entradas posibles para el campo melódico pertenecen a un conjunto limitado de combinaciones, formadas por la nota en nomenclatura anglosajona, su octava y una representación numérica de la duración, del modo expresado en la figura 81:

[carácter nota][valor octava],[valor duración]

Figura 81: Sintaxis de las notas para la búsqueda.

Tanto el nombre de la nota como el la octava siguen la nomenclatura conocida (notas de A a G y octavas de -1 a 9); pero en el caso de la duración, se ha decidido no emplear los valores de MIDI y jMusic, sino otros más próximos a la tradicional nomenclatura musical. Algunos de los indicadores de duración empleados se encuentran en la tabla 16:

Tabla 16: Ejemplos de Valores de duración empleados en el Buscador.

Figura	Valor Buscador	Valor MIDI & jMusic
Blanca	2	2.0
Negra	4	1.0
Negra con puntillo	3	1.5
Corchea	8	0.5

Los métodos implementados en la clase *Traductor* contemplan la traducción de las notas con su respectiva octava y los indicadores de duración a los dos valores MIDI correspondientes. Esta operación es posible para todos el rango de tonos y duraciones permitido por jMusic. En la práctica los valores de entrada estarán limitados por las opciones que ofrezca la interfaz de usuario, lo cual garantiza que no se introduzcan valores no contemplados.

Tras obtener un campo de consulta melódico es necesario un paso previo consistente en contar el número total de notas. Este dato es relevante para la ejecución del proceso de normalización, pero también para eliminar consultas que no se pueden llevar a buen término por no contar con un número de notas suficientes.

```
int contadorNotas(String melodiaIntroducida){

  int contador=0;
  String[] notasIntroducidas;
                          //Cada nota es una letra de la A a la G, seguida de número,
                          seguida de coma y de otro número
  notasIntroducidas=melodiaIntroducida.split("[A-G]\\d,\\d");
  contador=notasIntroducidas.length;
  return contador;
}
```

Figura 82: Código contador de notas.

El proceso de normalización generará como resultado una cadena String que contenga la representación melódica correspondiente al tipo de búsqueda que se desea realizar. En el caso de las representaciones relativas, por motivos obvios ya evidenciados con anterioridad, se iniciará el cálculo a partir de la segunda nota introducida, y la consulta comenzará con la representación relativa de esta segunda nota.

```
for(int x=1;x<tam;x++){          //Comienzo en la segunda nota

        buffer.append("T");
          if(((Integer)tonos.elementAt(x)).intValue()<0){//Si esta nota es silencio
            buffer.append(" ");
          }
          else if(((Integer)tonos.elementAt(x-1)).intValue()<0){//Si esta nota no es silencio
pero la anterior lo fue
            int n=x-1;
            while(((Integer)tonos.elementAt(n)).intValue()<0){//entonces, busca hasta que
encuentre una nota precedente que NO sea un silencio
              n--;
            } buffer.append(((Integer)tonos.elementAt(x)).intValue()-
                                          ((Integer)tonos.elementAt(n)).intValue() );
          }else{
            buffer.append(((Integer)tonos.elementAt(x)).intValue()-
                                        ((Integer)tonos.elementAt(x-1)).intValue() );
          }
          buffer.append(" D");
          buffer.append(((Double)duraciones.elementAt(x)).doubleValue()/
                                  ((Double)duraciones.elementAt(x-1)).doubleValue() );
          buffer.append("\t");
        }
        melodiaNorma = buffer.toString();
        return melodiaNorma;          // devuelve las notas normalizadas
}
```

Figura 83: Código para la normalización de una consulta en representación relativa.

El número mínimo de notas que debe introducir el usuario es dos, y éste es un requisito que se comprueba antes de ejecutar el proceso de normalización. En caso de no alcanzar ese mínimo, la búsqueda contemplará consultas para el resto de campos introducidos (p. ej. autor, título, etc.), pero no lo hará para ningún campo melódico.

La representación melódica relativa para búsquedas sin silencios empleará el mismo cálculo descrito en la figura 83, previa eliminación de los silencios en la cadena introducida, si los hubiera. La diferencia estribará en que la consulta posterior con esta representación se hará sobre el campo sin silencios (*melodía_sin*).

Las representaciones relativas independientes de tono e independientes de duración, se obtienen tras eliminar de la cadena original los tonos y las duraciones, respectivamente. Para el cálculo de la primera se seguirá el mismo algoritmo descrito anteriormente sin la inclusión de lo referente a los tonos, y en el caso de la segunda será necesario omitir todo lo referente a duraciones.

Para la representación melódica absoluta, tan sólo es necesario leer los elementos *tonos* y *duraciones* y trasladar los valores a una cadena con las referencias pertinentes (*T* para tonos y *D* para duraciones); como siempre, en el caso de los silencios se introducirá un espacio vacío en el lugar del tono.

II. Proceso de elaboración de consulta

Para realizar una búsqueda sobre el índice es necesario construir primero el objeto que representa la consulta (el objeto *Query*). Una *query* está constituida por una serie de términos de búsqueda para unos determinados campos del índice y además incorpora las relaciones de los términos entre sí.

La *query* se construye a partir de un objeto *QueryParser* que es el encargado de fragmentar los campos de consulta en términos, del mismo modo que se hizo con los campos documentales en la indexación. Como los criterios de análisis para dividir en términos han de ser idénticos, se emplea el mismo analizador *StandardAnalyzer*.

Los *QueryParser* se asocian desde su inicialización con el analizador y un determinado campo del índice por defecto. El campo por defecto se aplicará para todo el contenido textual introducido que no contenga una etiqueta específica. Las etiquetas específicas, tal y como se describen en la sintaxis del *QueryParser*, se

aplicarán en este caso para los campos de metadatos (título, autor, etc.) y la consulta para el campo melódico será la parte del texto introducido que no lleve asociada ninguna etiqueta.

Sólo se podrá buscar a la vez en un campo melódico. Las cinco opciones diferentes de búsqueda por melodía que se le ofrecen al usuario son excluyentes entre sí, y son estrictamente las que corresponden a los campos existentes. Se inicializará el *QueryParser* con la indicación de campo por defecto adecuada respecto a la representación melódica que se desea buscar (*melodía*, *melodía_abs*, *melodía_st*, etc.).

```
        a = new StandardAnalyzer();  //analizador
        QueryParser parser;

if(absoluta)   parser = new QueryParser("melodia_abs",a);
else{   !absoluta
        if(conSilencios){

          if(independienteTonalidad){       parser = new QueryParser("melodia_st",a);
          }
          else if(independienteDuracion){ parser = new QueryParser("melodia_sd",a);
          }
          else{                    parser = new QueryParser("melodia",a);
          }
          (...)
        else{ //(!conSilencios)

       parser = new QueryParser("melodia_sin",a);
           (...)
       }   (...)
  }
  if(contadorNotas(querymelodia)>=2){
    querymelodiaNormalizada=Buscador.normaliza(querymelodia);
  }
  else descartaMelodia=true;
```

Figura 84: Inicialización de *QueryParser* con los campos por defecto correspondientes.

Al final, el *QueryParser* recibirá una cadena de texto con toda la información a buscar, tanto en el campo melódico, como en los campos de metadatos. Esta cadena de texto puede seguir unas determinadas reglas, marcadas por la sintaxis específica de *QueryParser* [23], para indicar relaciones entre términos. En el caso de este sistema, el formato en que se genera esta cadena es el ilustrado en la figura 85.

+autor:([términos] **) +titulo:(** [términos] **) +categoria:(** [términos] **) +letra:(** [términos]**)**
+parson:([términos] **)** [melodía]

Figura 85: Formato general del texto destinado al *QueryParser* en la aplicación.

Esta disposición permite establecer una relación *AND* lógica entre los campos que aparecen con el signo '+', lo cual quiere decir que al menos uno de los términos de cada campo deben coincidir en el mismo documento.

```
//según corresponda> 0:autor  1:titulo  2:categoria   3:letra  4:parsonsCode

if(querycampos[0]!=null && querycampos[0].length()>0){

  queriesCampos=queriesCampos+"+autor:(";
  queriesCampos=queriesCampos+querycampos[0]+" ";
  queriesCampos=queriesCampos+") ";
}
if(querycampos[1]!=null && querycampos[1].length()>0){

  queriesCampos=queriesCampos+"+titulo:(";
  queriesCampos=queriesCampos+querycampos[1]+" ";
  queriesCampos=queriesCampos+") ";

} (...)
```

Figura 86: Concatenación de elementos de consulta para formar la *Query*.

Completada la cadena de texto anterior, tan sólo es necesario ejecutar el método *parse()* sobre el objeto *QueryParser,* para obtener el objeto que va a permitir realizar la consulta sobre el índice (*Query*). De este modo la consulta ha quedado dividida en términos que se buscarán individualmente en el índice.

```
if(descartaMelodia)    query=queriesCampos;
else                   query=queriesCampos+"\t"+ querymelodiaNormalizada;

 if(query.length()>0){   //Sólo si se ha añadido algo a la cadena
Query q = parser.parse(query);
 }
```

Figura 87: Creación del objeto *Query*.

III. Proceso de ejecución de consulta

Para realizar la consulta sobre el índice se ha de instanciar un objeto de la clase *IndexSearcher;* éste se encargará de leer el índice para encontrar los términos de búsqueda que se le indiquen mediante el método *search()*.

```
IndexSearcher searcher = new IndexSearcher(index);
```

Figura 88: Creación de *IndexSearcher* para un índice concreto 'index'.

Los resultados de ejecutar el método *search()*, se recogen directamente en un objeto *Hits,* que los ofrece ordenados de acuerdo a la puntuación obtenida.

```
        Hits hits= searcher.search(q);


      //-RESULTADOS-
        resultados= new String[hits.length()];
        scores= new float[hits.length()];
        titulos= new String[hits.length()];
        autores= new String[hits.length()];


      //Arrrays para guardar los resultado en orden:
        for (int i = 0; i < hits.length(); i ++){

          resultados[i]=hits.doc(i).get("nombre") ;
          scores[i]=hits.score(i);
          titulos[i]=hits.doc(i).get("titulo");
          autores[i]=hits.doc(i).get("autor");
        }
      }
```

Figura 89: Código que recoge la ordenación de resultados mediante *Hits*.

Esta puntuación que sirve para ordenar los resultados de acuerdo a su grado de coincidencia, sigue unos pasos de cálculo estrictos, que ya han sido citados con anterioridad:

1) Se debe encontrar alguno de los términos de la búsqueda en el campo correspondiente del índice.

2) Se toman estos términos coincidentes y se establece una comparación vectorial (ente los términos de búsqueda y los hallados en el índice) que toma en cuenta las posiciones relativas de los términos. A menor distancia entre vectores, mayor grado de semejanza.

3) Aplicación de diversos factores de ponderación. En concreto Lucene, aplica una fórmula que considera por cada término coincidente una serie de factores influyentes en una mayor puntuación (véase en la figura 90 y tabla 17):

$$\sum_{t=first\ term}^{terms\ query} f_t(D) \cdot inv\_f_t(D) \cdot boost(field) \cdot lengthNorm(field)$$

Figura 90: Fórmula de puntuación por factores.

Tabla 17: Factores incluidos en la fórmula de puntuación de resultados.

Factor	Descripción
$f_t(D)$	Frecuencia del término *t* en el documento *D*
$inv\_f_t(D)$	Frecuencia inversa documental del término (en cuántos documentos aparece).
$boost(field)$	Factor de realce o ponderación del campo, puede ser configurado en la indexación. En este sistema, todos los campos melódicos tienen un factor de 2, frente al valor por defecto del resto de los campos que es 1.
$lengthNorm(field)$	Valor de normalización del campo, dado el número total de términos existentes en él. Provoca que una misma coincidencia obtenga más puntuación cuando tiene lugar en un campo más pequeño. Este valor realmente se calcula en tiempo de indexación y se almacena en el índice.

5.2.2. Interfaz de usuario

Se incluye en el diseño de la aplicación de búsqueda una interfaz, destinada a introducir peticiones de búsqueda y presentar los resultados de las mismas. Como ya se ha señalado, se ha empleado una interfaz preexistente adaptada a los requerimientos de esta aplicación.

Se trata de una interfaz web que presenta:

- Teclado reducido para la introducción de melodías.
- Cuadro de texto de sólo lectura que presenta los valores de las notas introducidas.
- Pentagrama para la visualización de las notas introducidas.
- Menús desplegables para la selección de diferentes opciones de búsqueda melódica.
- Cuadros de entrada de texto para la introducción de campos de información (autor, título, letra, etc.)

Los menús desplegables presentan las cinco opciones de búsqueda de melodías que son mutuamente exclusivas, según se definieron en el módulo *Buscador*. Las diferentes opciones implican la realización de una consulta en un campo concreto por parte del módulo *Buscador*, tal y como se detalla en la tabla 18.

Tabla 18: Diferentes opciones de selección de métodos de búsqueda para melodías.

Opción seleccionada	Efecto
"Buscar melodías relativas"	Búsqueda de la melodía en notación relativa (campo *melodía*)
"Buscar melodías exactas"	Búsqueda de la melodía en notación absoluta (campo *melodía_abs*)
"Ignorar silencios"	Búsqueda de la melodía en notación relativa sin silencios (campo *melodía_sin*)
"Independiente de la altura (sólo proporción rítmica)"	Búsqueda de la melodía en notación relativa sin los tonos, sólo duraciones (campo *melodía_st*)
"Independiente de las duraciones (sólo relación de tonos)"	Búsqueda de la melodía en notación relativa sin las duraciones, sólo tonos (campo *melodía_st*)

En lo que respecta a los cuadros de entrada de texto, debido al modo en que se ha decidido que se realicen las consultas en el módulo *Buscador*, se comportarán entre sí como una intersección lógica (*AND*). Sin embargo, los términos individuales contenidos seguirán un relación *OR*, que se podrá alterar si se introduce entre ellos el texto '*AND*'.

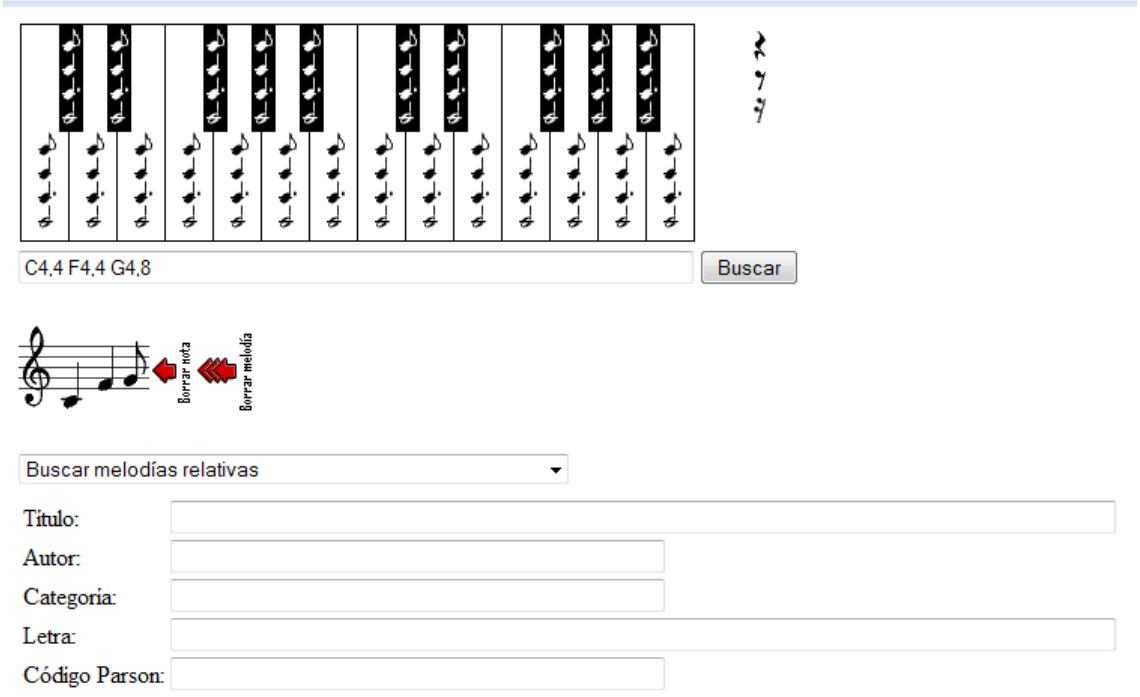


Figura 91: Captura de pantalla de la interfaz de usuario web.

6.Validación del sistema

La validación de este sistema ha tenido lugar de forma paralela a su desarrollo, como factor importante en la optimización del diseño, así como tras la finalización del mismo.

De acuerdo con la estructura del sistema, organizado en torno a dos bloques funcionales, indexación y búsqueda, las pruebas de evaluación se han realizado en ambos. Si bien, debido a la naturaleza de la aplicación, las pruebas efectuadas en la búsqueda han sido de mayor importancia.

6.1. Descripción del conjunto de prueba

Para la evaluación se ha empleado un conjunto de 115 melodías, con sus correspondientes archivos MIDI y descripción textual asociada. Tales materiales han sido extraídos en su totalidad de la enciclopedia abierta Musipedia [42] . Los archivos MIDI utilizados son, en su mayoría, de tipo 0 -una única pista-, y un número reducido de ellos de tipo 1 -varias pistas sincronizadas, en este caso dos-.

El proceso a seguir para la consecución de las pruebas, corresponde de manera lógica con la secuencia de pasos realizados en el sistema: se realiza la petición de extracción de todos esto archivos MIDI -junto con sus correspondientes metadatos asociados- al módulo *Lector;* para a continuación llevar a cabo las tareas de verificación detalladas en los siguientes apartados.

6.2. Pruebas de indexación

Para verificar el funcionamiento correcto de la indexación, era necesario comprobar la adecuada generación de archivos XML, así como la posterior creación y agregación de unidades documento al índice. En ambos, debían aparecer los datos y la representación correcta para cada melodía introducida.

En la tabla 19 se recogen los campos de contenido del índice, los cuales deberán corresponder con el contenido especificado.

Tabla 19: Campos a verificar en una unidad documento del sistema.

Nombre de campo	Contenido verificable	Separación términos
nombre	Texto que debe contener el nombre del archivo MIDI	No
autor	Texto formado por 0 o más términos que debe contener la información procedente, en consonancia con el archivo introducido. Sin errores de codificación de caracteres.	Sí
titulo		Sí
categoria		Sí
letra		Sí
parson		Sí
melodia	Texto formado por varios términos que debe contener la representación numérica correspondiente de la melodías. La serialización de melodía realizada en el *Lector* ha de ser correcta.	Sí
melodia_abs		Sí
melodia_st		Sí
melodia_sd		Sí
melodia_sin		Sí

6.2.1.Documentos XML

La comprobación de los documentos en XML no requirió de procesamientos ni herramientas específicas, su visualización tiene carácter inmediato. Todos los documentos generados fueron examinados tras la finalización del desarrollo con un resultado satisfactorio.

En este proceso de examen se prestó especial atención al modo en que eran representados los campos melódicos, ya que estos son los documentos generados precisamente a la salida del módulo de extracción (encargado de proporcionar las representaciones). Los valores generados debían corresponder con los valores MIDI, y presentarse en un orden correcto tras la serialización. Este último aspecto era importante ya que, aunque los archivos de tipo 0 contenían una única línea melódica, sin notas simultáneas, en el caso de los de tipo 1, algunos se encontraban como monofónicos en la práctica (tenían dos pistas, pero no de notas simultáneas sino sucesivas), y otros como polifónicos (con pares de notas simultáneas). El módulo generó para todos los archivos los valores adecuados y realizó la serialización del modo esperado: el contenido completo de una pista a continuación de la otra.

Un ejemplo del formato exacto que siguen estos documentos ya ha sido expuesto en el capítulo de implementación, figura 69.

6.2.2.Contenido indexado

En segundo lugar, se requería verificar si la composición del índice era la correcta. Tal operación no es inmediata, sino que precisa de la ayuda de la aplicación Luke [34] para visualizar su contenido.

El contenido del índice debía coincidir con el existente en los documentos XML, partiendo de la situación en que la generación de éstos era correcta. Además, dicho contenido tendría que estar convenientemente separado en términos individuales, tras la intervención del analizador elegido, y según los requisitos marcados.

Con la aplicación Luke, se exploraron uno a uno los campos de documentos indexados y los términos individuales presentes en el índice, para constatar la validez de los procesos llevados a cabo en la indexación, según los parámetros detallados con anterioridad. Las observaciones realizadas a la conclusión del desarrollo permiten afirmar que tales procesos se realizan de forma correcta.

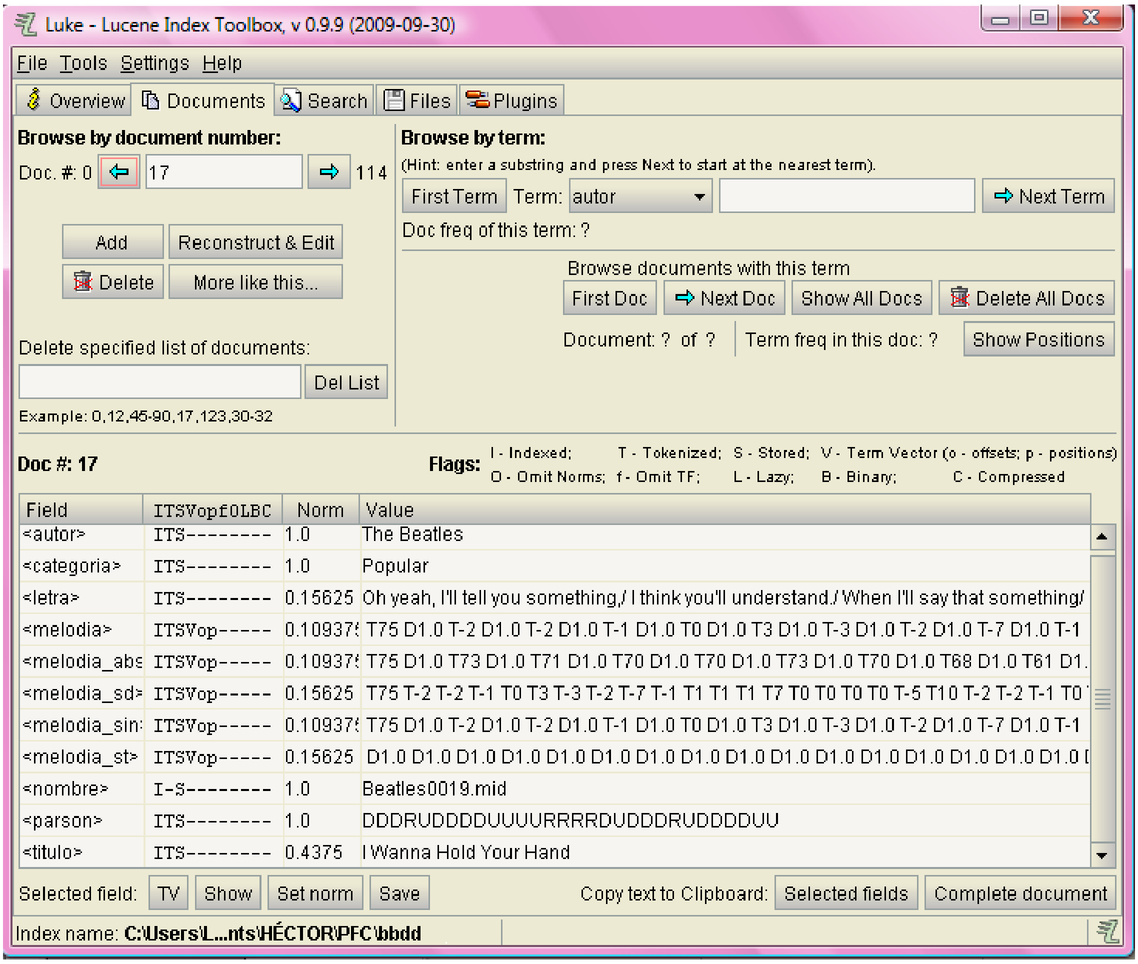


Figura 92: Campos presentes en el índice para una melodía determinada, visualización mediante la aplicación Luke.

6.3. Pruebas de búsqueda

Las pruebas centradas en procesos de búsqueda se consideraron como las de mayor relevancia en la evaluación global del sistema. No en vano, se trata del diseño de un buscador, para el cual la eficiencia en esta fase es el aspecto más crucial.

Se realizó una batería de pruebas haciendo uso del bloque de búsqueda integrado al completo, es decir, ejecutando consultas desde la interfaz de usuario, cuyos resultados eran visionados también por este medio tras la interacción con el módulo *Buscador*. En las pruebas se elaboraron todo tipo de consultas en base al contenido conocido del índice, tanto búsquedas de texto como melódicas y también ambas a la vez. Se pretendió demostrar su eficacia para las diferentes opciones de búsqueda, con la inclusión de términos erróneos, combinaciones inválidas, melodías transportadas y con duraciones alteradas, etc.

El resultado de la evaluación se puede calificar de positivo, ya que no se observaron fallos en la recuperación de melodías dentro de los límites lógicos. En otras palabras, el sistema es capaz de recuperar información correctamente siempre que cuenta con elementos mínimamente identificativos, incluso a pesar de que existan errores, omisiones o alteraciones ligeramente notables. Para una mayor claridad, se incluyen a continuación varios ejemplos tomados a partir de las pruebas realizadas.

6.3.1.Ejemplo de prueba 1

6.3.1.1.*Descripción*

Se realiza una búsqueda exclusivamente textual que pretende encontrar melodías pertenecientes a la ópera *Die Zauberflöte*.

6.3.1.2.*Interpretación de resultados*

Se retornan todas las melodías identificadas como *"Zauberflote"* en su título, con una puntuación del 99%, no presentan un 100% ya que los títulos contienen más términos.

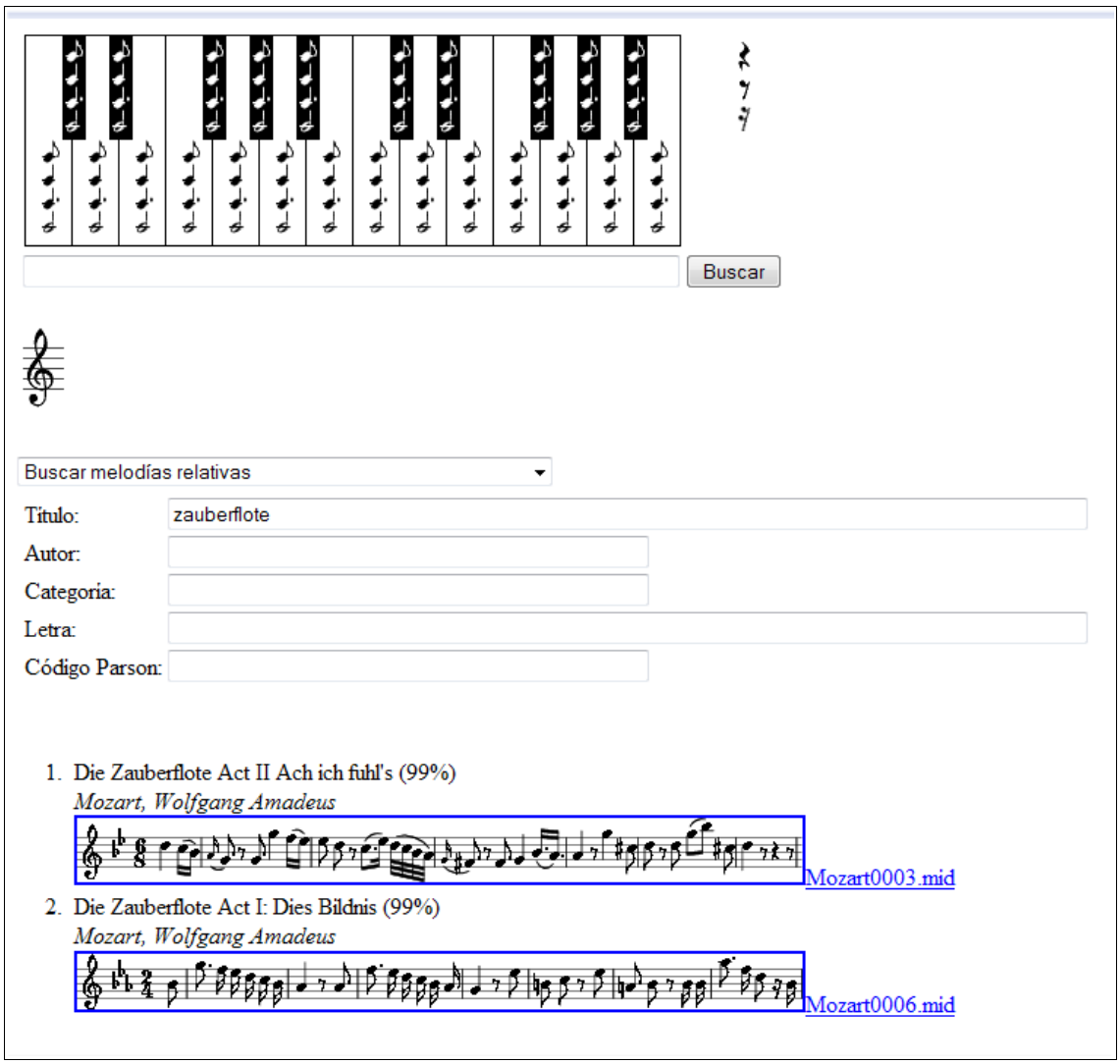


Figura 93: Detalle de prueba Nº 1.

6.3.2.Ejemplo de prueba 2

6.3.2.1.*Descripción*

Se elabora una petición de consulta errónea en la que se pide una melodía titulada "turkish march" con autor "mozart", y en realidad es una melodía de Beethoven.

6.3.2.2.*Interpretación de resultados*

El sistema, que relaciona los campos de entrada de texto mediante un *AND*, no devuelve ningún resultado. Tal efecto es el deseable ya que se trata de una combinación errónea.

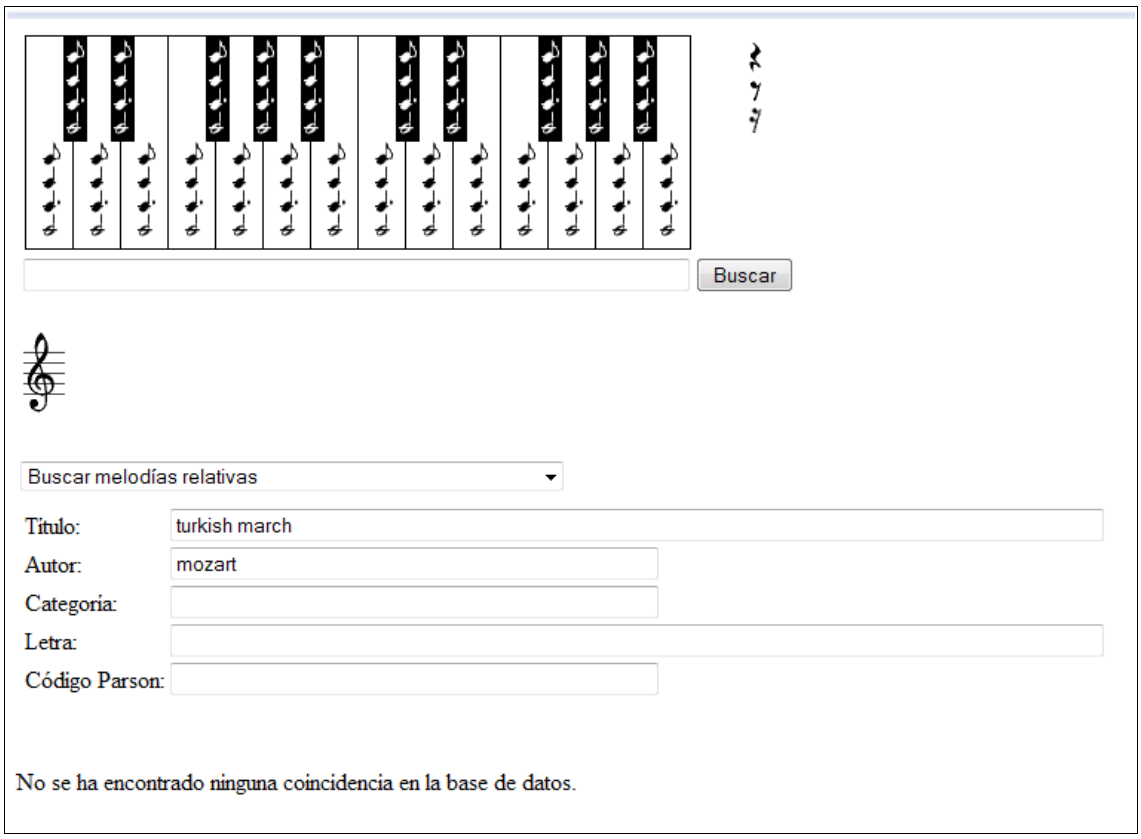


Figura 94: Detalle de prueba Nº 2.

6.3.3.Ejemplo de prueba 3

6.3.3.1.*Descripción*

De nuevo, se ejecuta una consulta textual en relación a la *Marcha Turca*, en esta ocasión se consigna el término "turkish" para el título, pero en el caso del autor, el usuario introduce "mozart beethoven", ya que no tiene claro a quien pertenece.

6.3.3.2.*Interpretación de resultados*

La aplicación de búsqueda devuelve la melodía que existente en el índice de la *Marcha Turca* de Beethoven. Los términos introducidos en el campo autor (al igual que en el resto) se relacionan por defecto mediante un *OR*, y ello permite resolver satisfactoriamente situaciones como ésta.

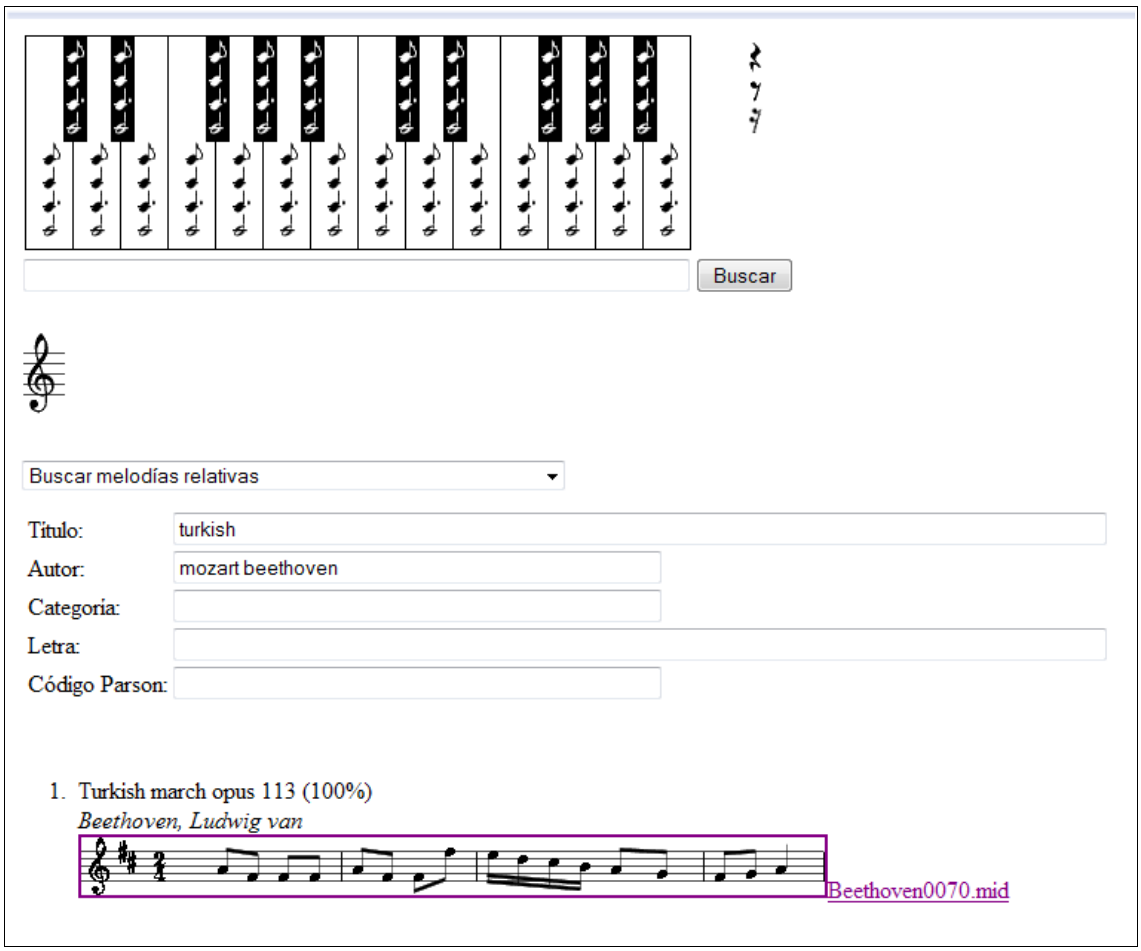


Figura 95: Detalle de prueba Nº 3.

6.3.4.Ejemplo de prueba 4

6.3.4.1.*Descripción*

Se realiza una búsqueda sólo textual basada en el contorno melódico de Parson, para localizar una melodía concreta con autor "beethoven" y de la categoría "classical".

6.3.4.2.*Interpretación de resultados*

El proceso de búsqueda retorna la melodía que corresponde exactamente a la información introducida. No existen más resultados que puedan relacionarse con los términos introducidos, y por tanto sólo se devuelve éste.

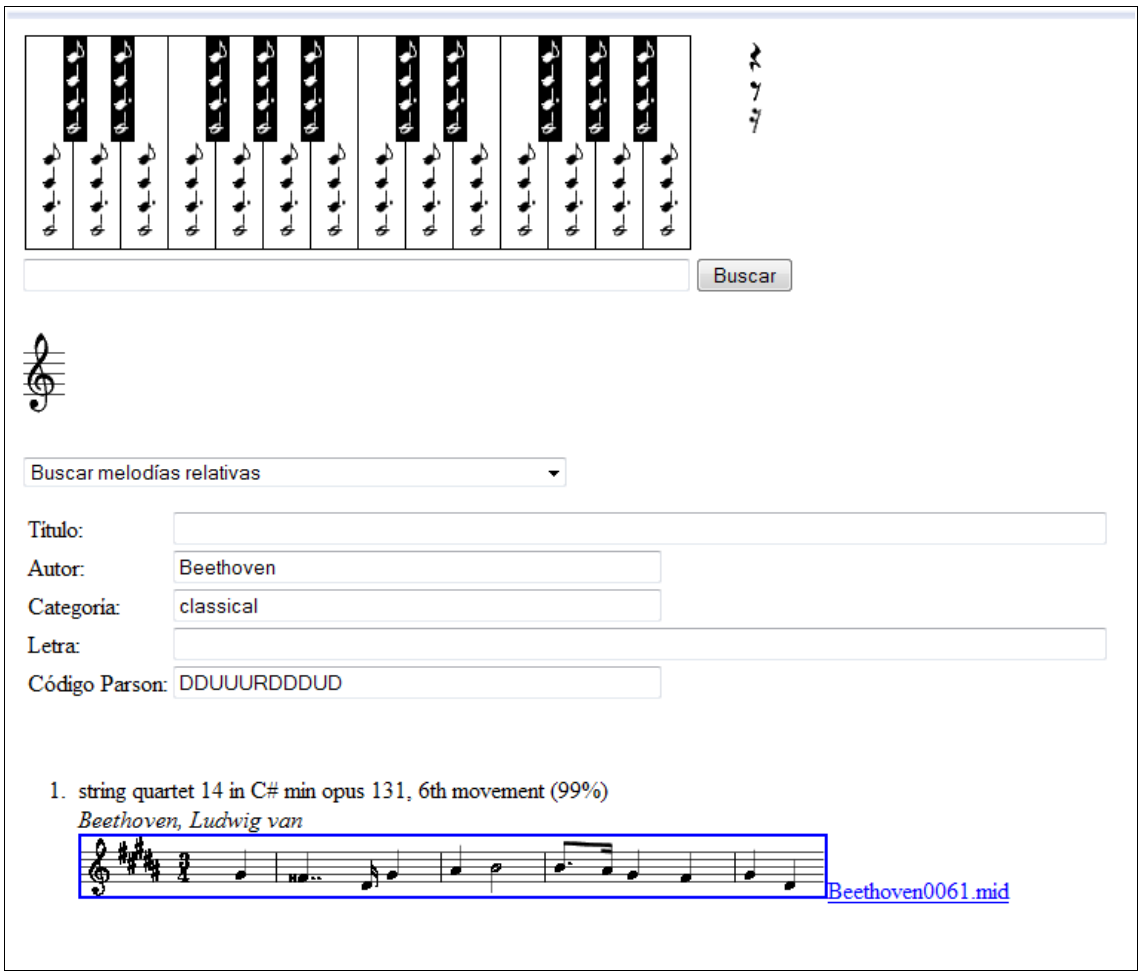


Figura 96: Detalle de prueba Nº 4.

6.3.5.Ejemplo de prueba 5

6.3.5.1.*Descripción*

En esta ocasión, se lleva a cabo una consulta melódica y se elige la opción de búsqueda exacta. Se efectúa con un fragmento comprendido entre el tercer y el cuarto compás de la obra "Piano Sonata 29 Bb opus 106" (el texto no se incluye).

6.3.5.2.*Interpretación de resultados*

A pesar de introducirse sólo un pequeño fragmento de la obra, el sistema devuelve sin problemas la melodía buscada con una puntuación del 100% y no retorna ninguna más, ya que es una combinación perteneciente en exclusiva a ésta.

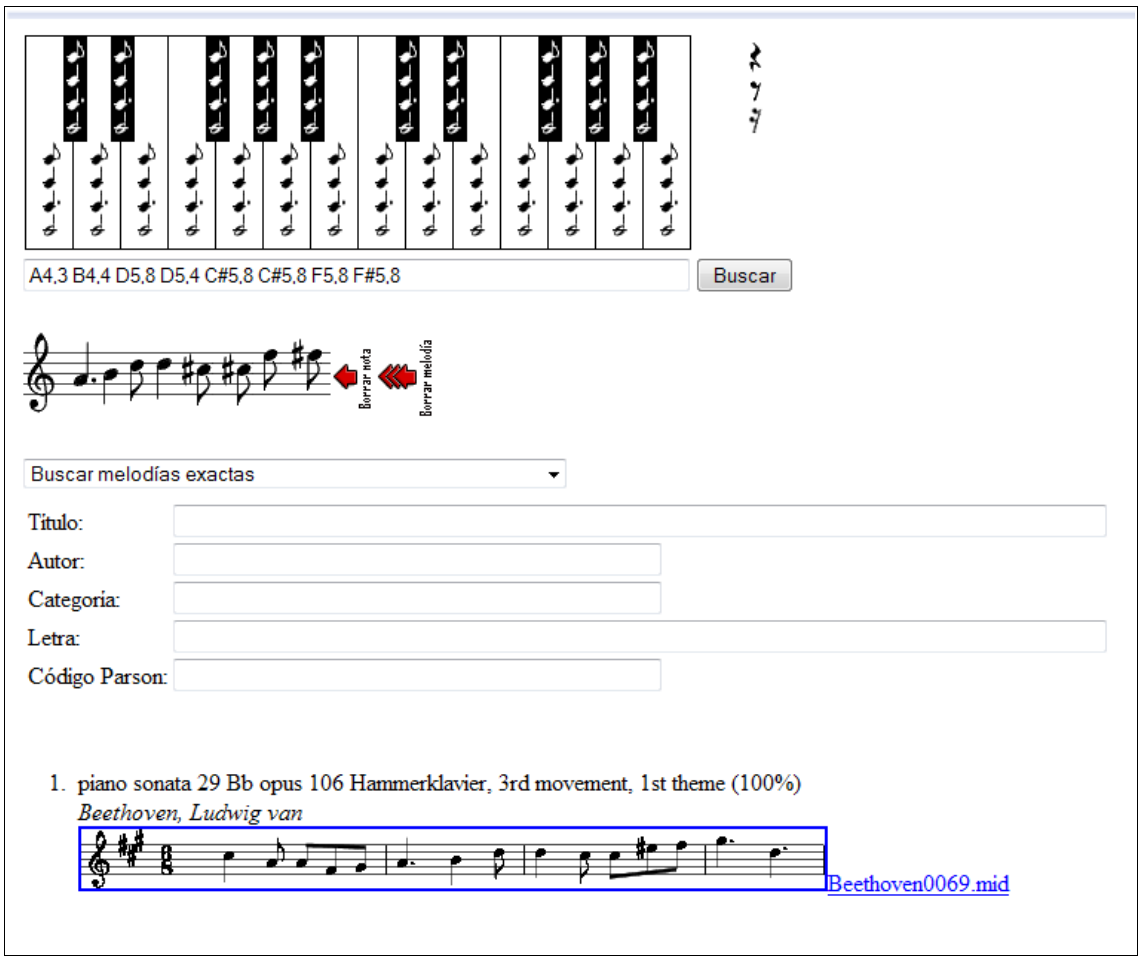


Figura 97: Detalle de prueba Nº 5.

6.3.6.Ejemplo de prueba 6

6.3.6.1.*Descripción*

Se efectúa la búsqueda de una melodía correspondiente a la canción *Let it bleed* a partir de un fragmento transportado de doce notas, un tono más agudo que el original y con errores en las dos últimas notas. Para ello se selecciona la opción de búsqueda relativa.

6.3.6.2.*Interpretación de resultados*

Se obtiene la melodía buscada en primer lugar, y con una puntuación relativamente alta otras dos melodías, que contienen dentro de sí muchas relaciones de alturas y de duraciones equivalentes a las buscadas. El resto de las melodías retornadas (se presentan de forma ordenada el resto de las presentes en la base de datos) tienen una puntuación muy inferior, ya que sólo existirá coincidencia para algunos términos aislados.

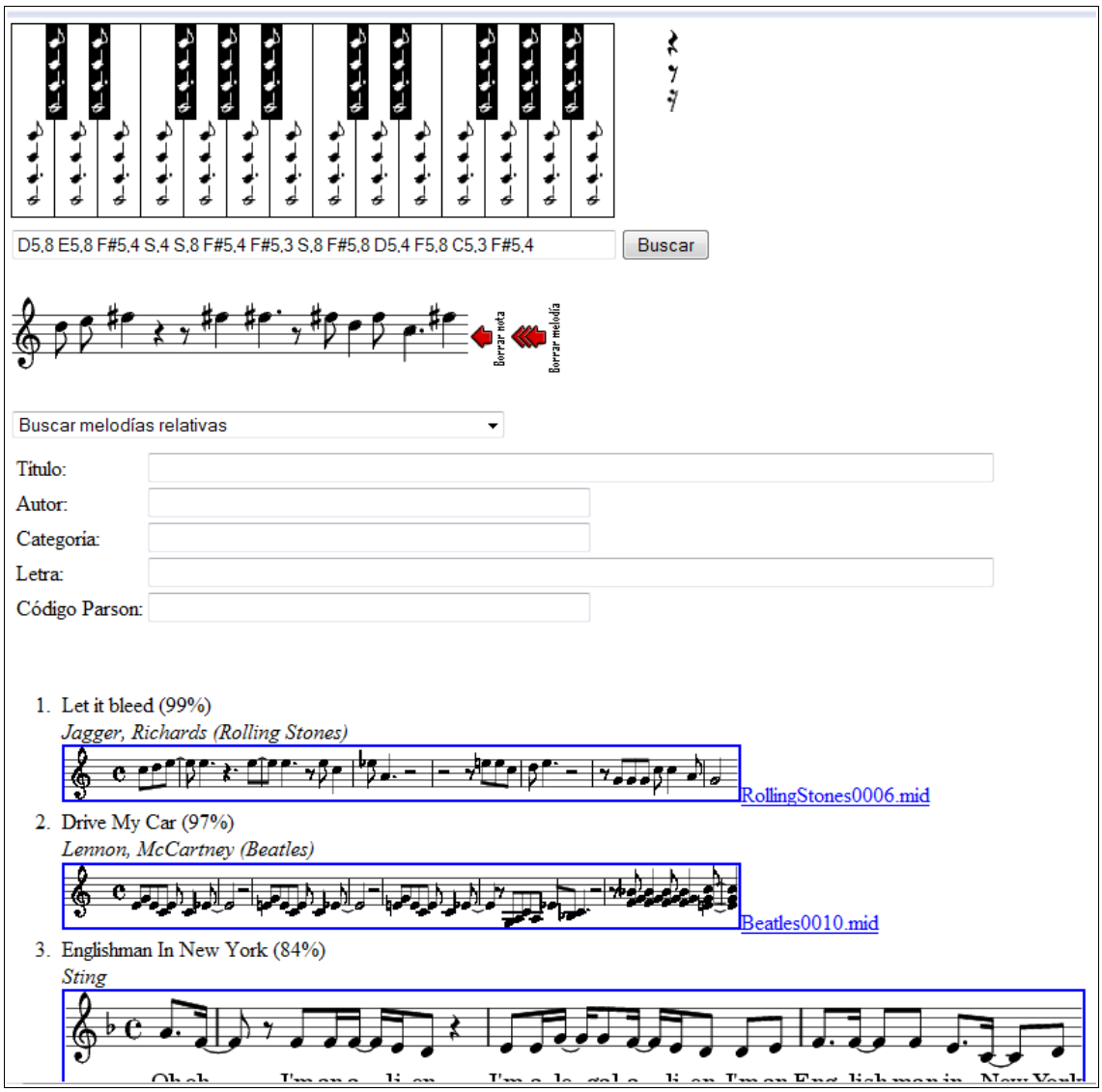


Figura 98: Detalle de prueba Nº 6.

6.3.7.Ejemplo de prueba 7

6.3.7.1.*Descripción*

En este ejemplo, se realiza una búsqueda exclusivamente rítmica de una melodía de *Piano sonata 31- opus 110*. Para comprobar que sólo se tienen en cuenta las duraciones y que se consideran de forma relativa, se introduce un fragmento con tonos aleatorios muy dispares y con figuras de duración que doblan la longitud de las originales.

6.3.7.2.*Interpretación de resultados*

Se retornó en primer lugar la melodía buscada con puntuación de 100%, pero también otras catorce con un puntuación mayor o igual que 90%. Tal hecho se produce porque estas melodías también contienen fragmentos con una proporción

de duraciones muy similar. Situaciones similares se solían producir para el resto de pruebas realizadas con esta especificación, incluso en algún caso la melodía buscada quedaba relegada y relativamente alejada del primer puesto (aunque con puntuación alta). Tal suceso no se puede considerar un error, simplemente es muy habitual que una secuencia rítmica de corta extensión, y hablando en términos de proporción, sea un elemento común para bastantes melodías. Sólo en algunos casos, por la extensión de la secuencia o bien porque existe una gran variabilidad en las duraciones de las notas, se generan resultados claramente aventajados.

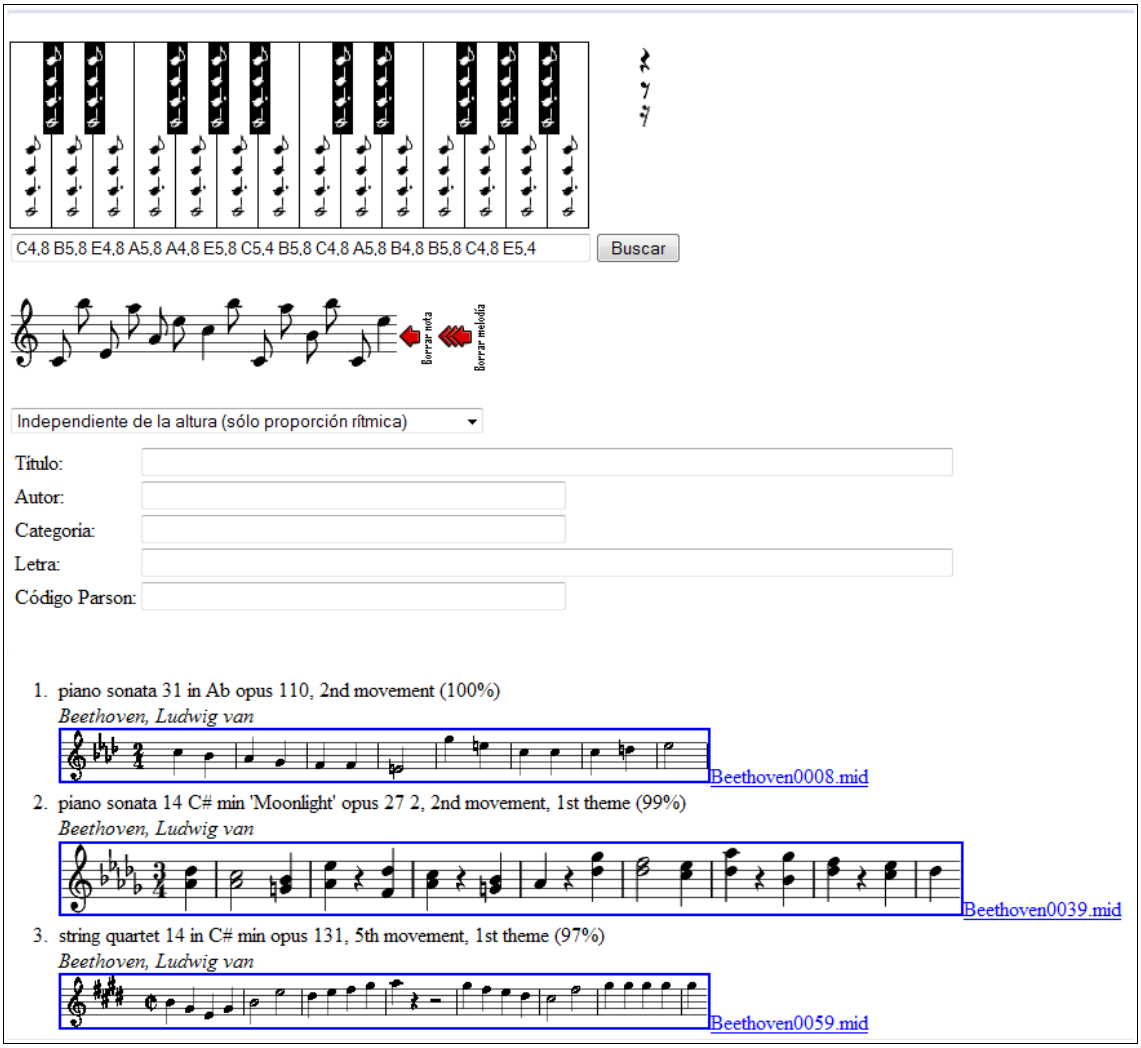


Figura 99: Detalle de prueba Nº 7.

6.3.8.Ejemplo de prueba 8

6.3.8.1.*Descripción*

Se lleva a cabo una consulta combinada que reúne texto e información melódica. Se simula el comportamiento de un usuario que conoce una información

limitada acerca de la melodía buscada: sabe que es una sinfonía y conoce un pequeño fragmento de la melodía, el cual introduce con una altura dos tonos más grave e incluye varios errores (la quinta nota introducida E4,4 debiera ser F4,4, y las dos últimas notas deberían ser C4,4 y A#4, en vez de E4,4 y G4,4).

6.3.8.2.*Interpretación de resultados*

El sistema presenta la melodía buscada en primer lugar puntuada al 100%, con clara ventaja respecto al resto de resultados. Los fragmentos que ocupan los primeros puestos son de manera lógica también pertenecientes a alguna de las sinfonías presentes en el índice.

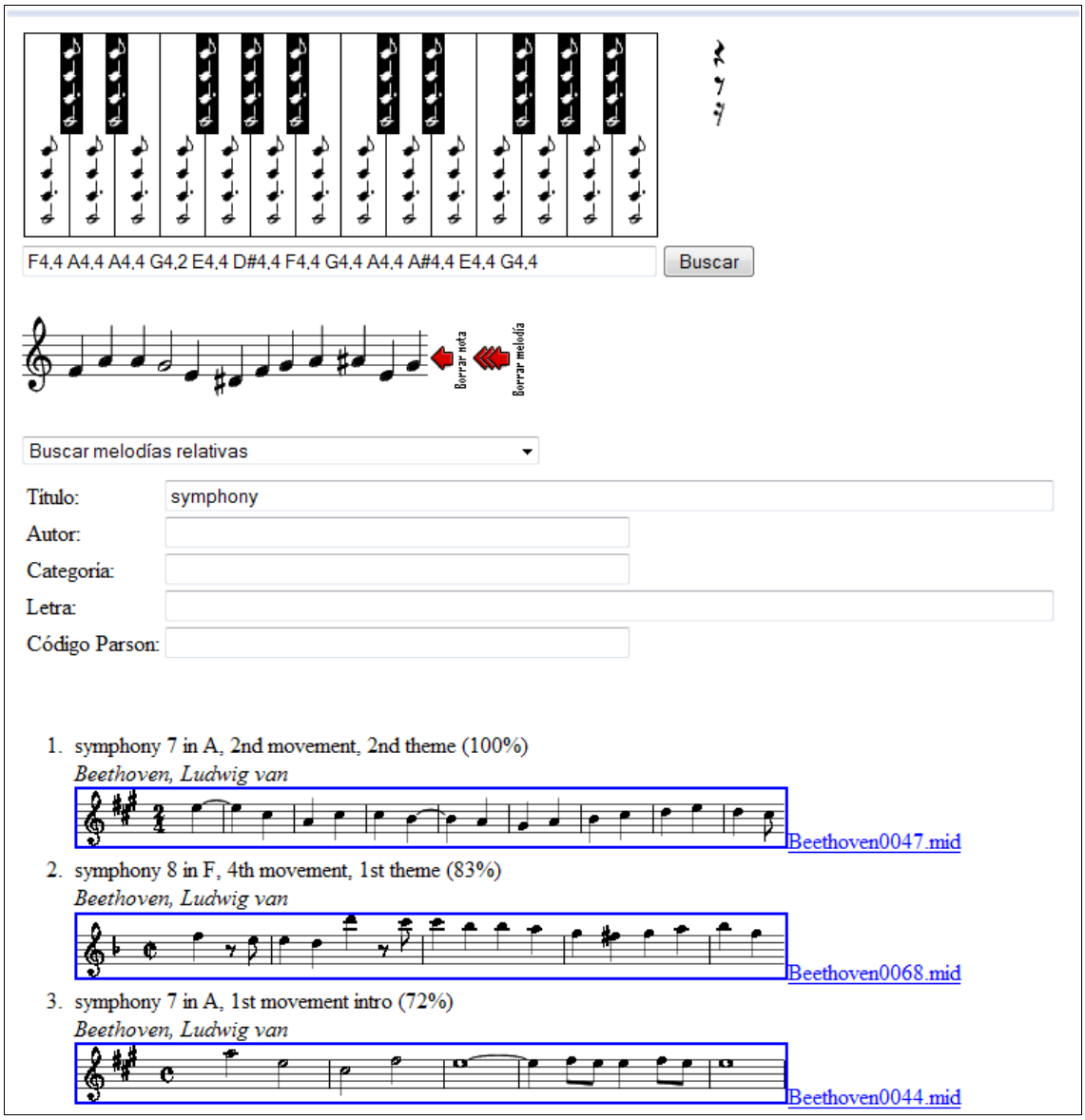


Figura 100: Detalle de prueba Nº 8.

7.Conclusiones y trabajos futuros

7.1. Conclusiones

El proyecto aquí descrito ha pretendido desarrollar un sistema capaz de realizar la búsqueda de una melodía musical determinada, entre un conjunto de melodías almacenadas.

El sistema se divide en dos bloques funcionales principales, tal y como se ha evidenciado a lo largo de los apartados expuestos con anterioridad. El primer bloque abarca los procesos que introducen elementos en el índice para constituir el conjunto de búsqueda. El segundo engloba la aplicación que permite realizar el proceso de consulta del buscador.

Las funciones realizadas por el bloque de indexación se agrupan en torno a dos módulos. El primero de ellos, el módulo de extracción, ha sido el encargado de realizar las tareas dependientes del formato de entrada para entregárselas de un modo transparente a los módulos encargados de la indexación. Este módulo lleva a cabo la función de representación de las melodías, por ello en esta primera etapa ya ha sido necesario plantearse cómo representarlas de forma adecuada en el índice.

El mayor esfuerzo en esta primera etapa de extracción se ha centrado en definir modos de representación significativos, tanto para identificar correspondencias exactas (representación absoluta), como correspondencias proporcionales (representación relativa). El módulo encargado de este procesamiento ha sido el que ha requerido una labor más intensa en lo que a desarrollo Java se refiere. Dada la naturaleza del sistema, las acciones desempeñadas en esta parte influyen sobre la capacidad de búsqueda en el extremo final de la aplicación; en consecuencia, a medida que se realizaban modificaciones se ha requerido comprobar su validez mediante pruebas de búsqueda.

El segundo módulo, compuesto por Constructor e Indexador, construye una unidad documento Lucene para la indexación con el objetivo de ser analizada y agregada al índice.

En este módulo la dificultad ha estado centrada en realizar los planteamientos adecuados para manejar la información del índice. Como

dificultades derivadas de la anterior, hubo que decidir qué campos de información iban a ser analizados y qué analizador era más correcto utilizar, además de decidir si se aplicaba mayor relevancia a algún campo, y cómo hacerlo.

El segundo bloque está constituido por la aplicación de búsqueda, cuya función la realiza un único módulo Buscador que puede ser gestionado desde la interfaz.

El módulo Buscador ha de emplear las mismas representaciones y análisis que se utilizan en el bloque anterior, el de indexación; por ello, además de definir otros aspectos propios de las consultas, como la agrupación de términos, se ha tenido que desarrollar este módulo de forma conjunta y paralela a los dos existentes en el primer bloque. Este hecho ha aportado mayor dificultad en el desarrollo del bloque.

A partir del desarrollo de los dos bloques, se pueden exponer una serie de conclusiones comunes. Una de las más evidentes es que para el desarrollo de este sistema ha sido fundamental entender y llevar a a la práctica las teorías de Recuperación de Información, y especialmente la aplicación concreta de éstas en el ámbito de Lucene.

La interrelación de los dos bloques se ha revelado como un factor clave en la elaboración del proyecto, de tal manera que las decisiones de diseño relativas a uno de los bloques tienen generalmente implicación en el otro. Así, cuando se ha decidido que las búsquedas se hagan de un determinado modo, la indexación se ha debido adaptar a la nueva situación propuesta, y en el sentido contrario, el modo en que se ha confeccionado el índice ha afectado a las posibilidades de búsqueda. Por este motivo, ha sido necesario emplear con frecuencia pruebas de validación desde el comienzo hasta el final de la fase de implementación, y las decisiones de desarrollo elegidas han estado muy ligadas a las pruebas realizadas.

La decisión de que los archivos de origen sean audio sintetizado en MIDI, proporciona una estructura próxima al lenguaje musical que se puede hacer fácilmente independiente de los parámetros físicos. Por otro lado, facilita su uso en la aplicación de búsqueda por ser de tamaño reducido y poder reproducirse en el entorno web por parte de la mayoría de exploradores.

En general, es posible asumir que todos los objetivos planteados en el apartado 1.2. se han cumplido de forma satisfactoria. Así lo atestigua la evaluación realizada de forma continua durante el desarrollo y al término de éste. Sin embargo, es adecuado considerar que este sistema puede someterse a diversas

mejoras. Tales mejoras quedan alejadas del alcance de este proyecto, pero se ha tomado en cuenta un número significativo de ellas en el siguiente apartado, el cual recoge los trabajos futuros a desarrollar en una posible ampliación del sistema.

7.2. Trabajos futuros

El ámbito musical y tecnológico sobre el cual se ha desarrollado este proyecto permite múltiples posibilidades de ampliación. Por ello, si bien el proyecto planteado ha sido concluido de acuerdo a su alcance propuesto, es necesario tener presentes las posibles tareas a realizar de cara a una futura mejora del desarrollo. A continuación, se plantean algunas de las que se han detectado con mayor relevancia.

El escenario actual de la aplicación contempla la introducción de nuevos elementos sólo mediante peticiones explícitas: se extrae y se indexa la información de una o varias melodías exclusivamente tras efectuar la orden pertinente. En este sentido, sería interesante plantear distintas estrategias dirigidas a realizar una adquisición dinámica de contenidos, necesaria sobre todo en entornos donde los volúmenes de información fueran mucho mayores. Por ejemplo, si se decidiese construir el índice a partir de elementos presentes en diversos puntos de la *WWW,* se tendría que diseñar algún tipo de rastreo que fuera capaz de analizar y agregar los nuevos contenidos.

Como ya se ha descrito, el módulo de extracción en este sistema adquiere la información musical a partir de ficheros de audio sintetizado MIDI. De acuerdo con su concepción modular ampliable, una segunda línea de actuación que se plantea consistiría en la posibilidad de añadir elementos al índice a partir de diversos formatos digitales de grabación de audio: Audio PCM (WAV), MP3, AAC (MPEG 2,4), etc. Tales opciones constituyen un problema avanzado de tratamiento de señal, que demandaría un extenso planteamiento.

La misma ampliación descrita anteriormente podría emplearse en el extremo final de la aplicación, la interfaz de búsqueda, para proporcionar una capacidad de consulta basada en el reconocimiento de audio musical. Por ejemplo, para que el usuario pudiese identificar un fragmento musical grabado o interpretado por él mismo. Ésta sería una opción alternativa a la introducción de notas mediante teclado en la interfaz web del buscador.

Por otra parte, el sistema planteado responde a un planteamiento de melodías monofónicas (sin sonidos simultáneos). El proceso de serialización

realizado en la fase de extracción proporciona en todos los casos, tanto en la lectura de melódicas únicas como en la de estructuras polifónicas, una única sucesión de notas; lo cual constituye una solución eficaz de acuerdo a los objetivos planteados, ya que el usuario pretenderá buscar exclusivamente líneas melódicas. No obstante, desde el punto de vista musical sería interesante estudiar la posibilidad de construir un índice que permitiera estructuras polifónicas y de esta forma el usuario podría, si así lo deseara, buscar obras a partir de los acordes o contrapuntos que la forman.

El buscador emplea el sistema de indexación textual de la API Lucene, mediante el cual se han logrado resultados considerablemente satisfactorios, como ha quedado patente en la exposición de este documento. Sin embargo, resulta interesante plantear como trabajo futuro la opción de crear nuevas estructuras propias para el lenguaje musical, realizando para ello una reescritura de algunos métodos de la API. La característica *open source* de Lucene abre muchas posibilidades futuras al respecto. También se puede plantear el uso futuro de otros sistemas de cometido similar, como *Lemur*, *Xapian*, etc.

La interfaz de búsqueda empleada en este sistema es una herramienta muy simplificada y, aunque ha estado presente, no ha sido un elemento clave en el diseño. Sin embargo, su importancia no es despreciable tal y como se recoge en el capítulo de Estado del Arte, al tratar acerca de componentes básicos de una aplicación de IR (apartado 3.4.3.2.), puede llegar a ser un elemento fundamental en el éxito de una herramienta. Por tanto, han de remarcarse en estos trabajos futuros aquellas mejoras relacionadas con la interfaz. Entre ellas, se pueden mencionar algunas a priori, tales como: la captación de notas mediante dispositivos externos (como el teclado del ordenador o un teclado MIDI), la posibilidad de obtener realimentación del usuario para conocer si los resultados corresponden con lo que éste estaba buscando o la visualización de búsquedas similares (como posible método de ayuda para corrección de errores del usuario).

De forma adicional, se puede plantear la introducción de una segunda interfaz que sirviera de acceso al bloque de indexación. De acuerdo al planteamiento de este proyecto, la implementación de dicha interfaz permitiría que los usuarios pudiesen introducir en el sistema nuevos elementos (melodías y descripciones de obras) de forma colaborativa.

Referencias

Las referencias bibliográficas se encuentran ordenadas alfabéticamente por apellidos de autor; las referencias localizadas en URL aparecen en segundo término, ordenadas por nombres de organización.

Recursos bibliográficos:

[1] Gerald Abraham. The concise Oxford History of Music. Ed. Oxford University Press, 1979.

[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto. Modern Information Retrieval. Ed. Addison-Wesley-Longman Publishing co., 1999.

[3] Andrew R. Brown. Computers in Music Education: Amplifying Musicality. Ed. Routledge, 2007.

[4] Andrew R. Brown. Making Music with Java. Ed. Lulu Australia, 2005.

[5] Melvil Dewey. Abridged Dewey Decimal Classification and relative index. Ed. Forest Press, 1997.

[6] Otis Gospodnetic, Erik Hatcher. Lucene in action. Ed. Manning, first edition 2005.

[7] Otis Gospodnetic, Erik Hatcher, Michael McCandless. Lucene in action. Ed. Manning, second edition 2009.

[8] T. Hammmel Garland y Ch. Vaughn Kahn. Math and Music: Harmonious Connections. Ed. Dale Seymour, 1995.

[9] Anselm Hughes. New Oxford History of Music, II, pp. 37-37. Ed. Oxford University Press, 1955.

[10] ISO (International Organization for Standardization). *ISO 8879:1986(E). Information processing — Text and Office Systems — Standard Generalized Markup Language (SGML).* First edition — 1986-10-15.

[11] All Media Guide's LASSO: US Patent 7,277,766

[12] Ulrich Michels. Atlas de música. Alianza Editorial, 1985.

[13] Federico Miyara. Introducción a la psicoacústica. Reporte técnico (reproducido en [37]).

[14] M. Ángel Moreno Flórez. Educación musical y expresión dinámica. Ed. SM, 1995.

[15] Denys Parsons. The Directory of Tunes and Musical. S.Brown (1975)

[16] Ken C. Pohlmann. Principles of digital audio. Ed. McGraw Hill, fourth edition.

[17] J. Reinthaler. Mathematics and Music: Some intersections, Mu Alpha Theta, 1990.

[18] E. Rothstein. Emblems of Mind: The inner life of music and mathematics. Ed. Avon Books New York, 1996.

[19] Francis Rumsey, Tim McCornick. Sound and recording: an introduction. Ed. Focal Press/Elsevier, fourth edition 2002.

[20] Isidoro de Sevilla. *Etymologiae*, III, 15 (reproducido en [1])

[21] David A. Vise, Mark Malseed. The Google Story: For Google's 10th Birthday. Ed. Delacorte Press, 2008.

[22] Egon Wellesz. New Oxford History of Music, I, pp 359-360. Ed. Oxford University Press, 1960.

Recursos en línea:

[23] Apache Software Foundation. Especificación de la sintaxis del QueryParser de Lucene: http://lucene.apache.org/java/docs/querypasersintax.html [visitado el 12/09/2009]

[24] Apache Software Foundation. Página oficial del proyecto Droids http://incubator.apache.org/droids/ [visitado el 11/08/2009]

[25] Apache Software Foundation. Página oficial del proyecto Hadoop: http://hadoop.apache.org/core
[visitado el 11/08/2009]

[26] Apache Software Foundation. Página oficial del proyecto Lucene: http://lucene.apache.org/ [visitado el 12/09/2009]

[27] Apache Software Foundation. Página oficial del proyecto Nutch: http://lucene.apache.org/nutch/ [visitado el 10/08/2009]

[28] Apache Software Foundation. Página oficial del proyecto Tika: http://lucene.apache.org/tika/
[visitado el 16/08/2009]

[29] Apache Software Foundation. Página oficial del proyecto Xerces:
http://xerces.apache.org/
[visitado el 16/08/2009]

[30] Apache Software Foundation. Wiki del proyecto Lucene:
http://wiki.apache.org/lucene-java/PoweredBy [visitado el 12/08/2009]

[31] Aperture. Página oficial del *framework* Aperture:
http://aperture.sourceforge.net [visitado el 10/08/2009]

[32] FDMF, desarrollos *Open Source*: http://www.w140.com/audio/
[visitado el 01/08/2009]

[33] Fraunhofer Institute. AudioID, desarrollo registrado:
http://www.idmt.fraunhofer.de/eng/research_topics/audioid.htm
[visitado el 01/08/2009]

[34] GeptOpt. Luke - Lucene Index Toolbox: http://www.getopt.org/luke/
[visitado el 01/10/2009]

[35] Google, "About": http:// www.google.com/intl/es/about.html
[visitado el 28/08/2009]

[36] Internet Archive. Página oficial del proyecto Heritrix:
http://crawler.archive.org/
[visitado el 10/08/2009]

[37] Rafael Losada. Música y Matemáticas. DivulgaMAT -Universidad País Vasco:
http://divulgamat.ehu.es/weborriak/Cultura/Musika/index.asp
[visitado el 22/06/2009]

[38] Microsoft Bing, "About":
http://www.discoverbing.com/behindbing/about.aspx
[visitado el 28/08/2009]

[39] Microsoft y Yahoo!, alianza estratégica:
http://www.choicevalueinnovation.com/thedeal/Default.aspx
[visitado el 28/08/2009]

[40] MIDI Manufacturers Association. Complete MIDI 1.0 Detailed Specification:
http://www.midi.org/techspecs/midispec.php [visitado el 20/07/2009]

[41] Musipedia, artículo oficial en Wikipedia:
http://en.wikipedia.org/wiki/Musipedia [visitado el 31/07/2009]

[42] Musipedia. Página del buscador musical: http://www.musipedia.org [visitado el 02/08/2009]

[43] Queensland University of Tecnology. Página oficial del proyecto jMusic: http://jmusic.ci.qut.edu.au/ [visitado el 25/07/2009]

[44] Refsnes Data. Material formativo sobre XML: http://www.w3schools.com/xml/ [visitado el 22/08/2009]

[45] Sax Project. Página oficial de SAX: http://www.saxproject.org/ [visitado el 16/08/2009]

[46] Shazam Entertainment. Shazam, tecnología registrada: http://www.shazam.com/ [visitado el 01/08/2009]

[47] Sony Corp. Gracenote, tecnología en propiedad : http://www.gracenote.com/ [visitado el 01/08/2009]

[48] Sun Microsystems. Documentación acerca de J2SE 1.3: http://java.sun.com/j2se/1.3/docs/#api [visitado el 24/07/2009]

[49] Sun Microsystems. Página oficial JMF: http://java.sun.com/javase/technologies/desktop/media/jmf/index.jsp [visitado el 24/07/2009]

[50] Wikia Search. Página oficial de la aplicación Grub: http://www.grub.org [visitado el 11/08/2009]

[51] W3C. Página oficial de especificaciones XML: http://www.w3.org/XML/ [visitado el 03/08/2009]

[52] W3C. Página oficial de DOM: http://www.w3.org/DOM/ [visitado el 15/08/2009]

[53] Yahoo! Aplicaciones de marketing de búsquedas: http://sem.smallbusiness.yahoo.com/searchenginemarketing/ [visitado el 28/08/2009]