UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE

AUTOMATIZACIÓN DE LA FISCALIZACIÓN DE LOS DERECHOS DE PROPIEDAD INTELECTUAL DE MÚSICA CHILENA EN RADIOEMISORAS ONLINE

MARCELA ALEJANDRA VIDAL RAMÍREZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA	:	MARCELO MENDOZA
PROFESOR CORREFERENTE	:	JAVIER ROBLEDO

Noviembre 2017

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

RESUMEN

La presente investigación tiene la finalidad de presentar una propuesta y metodología que permita automatizar la fiscalización de los derechos de propiedad intelectual de obras musicales, abarcando particularmente, las transmisiones de radioemisoras online. De esta manera es posible detectar la frecuencia de reproducción de canciones pertenecientes a artistas chilenos. Uno de los principales inconvenientes con el modelo que se utiliza en la actualidad, es la falta de acceso por parte del público a la información de la programación radial. Asimismo, músicos independientes deben autogestionar este proceso, pues sus obras no son parte de las bibliotecas musicales utilizadas como base para el reconocimiento acústico en el monitoreo de frecuencias radiales.

Para desarrollar el prototipo de la plataforma de monitoreo se analizaron diversos algoritmos, escogiéndose un proyecto open-source para implementar la identificación de canciones en base a huellas acústicas. Para la comprobación del cumplimiento de los objetivos planteados se llevó a cabo la configuración de radios online personalizadas, motivo por el cual es posible comprobar la autenticidad de la respuesta entregada por el programa. Adicionalmente se analizaron múltiples radios nacionales a fin de corroborar estabilidad. Finalmente, las pruebas realizadas demuestran la factibilidad de utilizar el sistema diseñado, siendo favorable para controlar aproximádamente cincuenta radioemisoras paralelamente en un solo equipo.

Palabras Clave. Algoritmos de reconocimiento de música. Monitoreo radioemisoras online. Huellas acústicas.

ABSTRACT

The present investigation has the purpose of presenting a proposal and methodology that allows to automate the control of the intellectual property rights of musical works, including, in particular, the transmissions of online radio stations. In this way it is possible to detect the frequency of reproduction of songs belonging to Chilean artists. One of the main drawbacks with the model that is currently used is the lack of access by the public to information on radio programming. Likewise, independent musicians should self-manage this process, since their works are not part of the musical libraries used as a basis for acoustic recognition in the monitoring of radio frequencies.

To develop the prototype of the monitoring platform, several algorithms were analyzed, choosing an open-source project to implement the identification of songs based on acoustic fingerprints. For the verification of the fulfillment of the proposed objectives, the configuration of personalized online radios was carried out, reason for which it is possible to verify the authenticity of the answer delivered by the program. Additionally, multiple national radios were analyzed in order to corroborate stability. Finally, the tests carried out demonstrate the feasibility of using the designed system, being favorable to control approximately fifty radio stations in a single unit.

Keywords. Music recognition algorithms. Online radio monitoring. Acoustic fingerprints.

ACRÓNIMOS

- **ARCHI** Asociación de Radiodifusores de Chile.
- **BMAT** Barcelona Music and Audio Technologies.
- **CNCA** Consejo Nacional de la Cultura y las Artes.
- **FFT** Fast Fourier Transform.
- **OMPI** Organización Mundial de la Propiedad Intelectual.
- **SCD** Sociedad Chilena del Derecho de Autor.
- **STFT** Short-Time Fourier Transform.

Índice de Contenidos

Resumen III

Abstract V

Acrónimos VII

Índice de Contenidos VIII

Índice de Tablas IX

Índice de Figuras XI

Introducción 1
Organización del documento . . . 3

1. Definición del problema 5
1.1. Ley 20810 . . . 5
1.2. Descripción del problema . . . 6
1.3. Objetivos . . . 8
1.3.1. Objetivo general . . . 8
1.3.2. Objetivos secundarios . . . 8

2. Estado del arte 9
2.1. Propiedad intelectual . . . 9
2.1.1. ContentID - Youtube . . . 11
2.1.2. Sistema de identificación de SoundCloud . . . 11
2.1.3. Identificador de Twitch . . . 12
2.1.4. Monitec . . . 12
2.1.5. Vericast . . . 13
2.2. Estructura y caracterización de la señal acústica . . . 13
2.3. Algoritmos de reconocimiento acústico . . . 23
2.3.1. Obtención de fingerprints en base a características . . . 23
2.3.1.1. Amplitud . . . 23
2.3.1.2. Algoritmo Phillips . . . 24
2.3.2. Obtención de fingerprints en base a machine learning . . . 24
2.4. Sistemas de reconocimiento acústico . . . 25
2.4.1. Echoprint . . . 25
2.4.2. Dejavu project . . . 26
2.4.3. Chromaprint . . . 26

3. Diseño de la solución 27
3.1. Metodología . . . 27
3.2. Selección de herramientas de desarrollo . . . 27

3.3. Descripción de Dejavu Project . . . 28
3.3.1. Extracción de fingerprints . . . 28
3.3.2. Coincidencia de fingerprints . . . 33
3.3.3. Estructura de archivos de Dejavu Project . . . 36
3.3.4. Análisis de funcionalidad . . . 38
3.4. Descripción del programa . . . 46
3.4.1. Fiscalización de una radioemisora online . . . 46
3.4.2. Fiscalización paralela . . . 48

4. Experimentos **51**
4.1. Casos de prueba . . . 51
4.1.1. Pruebas de estabilidad . . . 52
4.1.2. Pruebas de carga . . . 52
4.1.3. Pruebas de estrés . . . 53
4.2. Implementación . . . 54
4.3. Características del equipo . . . 55
4.4. Resumen de versiones . . . 56

5. Resultados **57**
5.1. Pruebas de estabilidad . . . 57
5.1.1. Clasificación de respuestas obtenidas por la plataforma . . . 57
5.1.2. Confianza . . . 59
5.1.3. Tiempo de respuesta . . . 60
5.2. Pruebas de carga . . . 61
5.2.1. Confianza . . . 61
5.2.2. Tiempo de respuesta . . . 63
5.2.3. Uso de memoria . . . 63
5.3. Pruebas de estrés . . . 64
5.3.1. Tiempo de respuesta . . . 65
5.3.2. Uso de memoria . . . 65

Conclusiones **67**

Bibliografía **71**

Apéndices **73**

Índice de Tablas

2.1. Función ventana ideal acorde al tipo de señal. 19
2.2. Frecuencia de resolución según el tamaño de la ventana. 20

3.1. Cuadro comparativo entre softwares open-source de reconocimiento acústico. 28
3.2. Nombre y tipos de datos que conforman las tablas de la base de datos de Dejavu. 33
3.3. Resumen de caso ejemplo para algoritmo de identificación de canciones. 35
3.4. Cálulo de diferencia entre offsets de caso ejemplo. 35
3.5. Correlación en nombres de variables del algoritmo de reconocimiento acústico. 37
3.6. Valores de los parámetros de la nueva configuración de Dejavu. 42
3.7. Tiempo de creación de bases de datos, con los valores de la nueva configuración. 43

4.1. Resumen de los equipos utilizados para testear la plataforma de fiscalización. 56

A1. Valores originales de los parámetros de Dejavu. 73
A2. Tiempo de creación de bases de datos, con los valores originales de Dejavu. 73
A3. Valores de configuración de las bases de datos de 3 y 5 canciones. 74
A4. Resumen de los resultados de Dejavu al probar 27 extractos de música. 75
A5. Resumen de los resultados de Dejavu al probar 18 extractos de música. 76
A6. Resumen de los resultados de Dejavu al probar 45 extractos de música. 77
A7. Resumen de los resultados de Dejavu al probar 216 extractos de covers. 78
A8. Resumen de los resultados de Dejavu al probar 27 extractos de covers. 79

Índice de Figuras

2.1. Muestreo acústico de señal analógica. 14
2.2. Dominio temporal y frecuencial. 16
2.3. Dominio frecuencia de señal Onda Cuadrada. 16
2.4. Ejemplo de muestreo de señales. 17
2.5. Aplicación de función ventana a una señal. 17
2.6. Ejemplo de funciones ventana. 19
2.7. Modos de representación visual de un archivo de música. 21
2.8. Espectrograma con sus peaks de audio. 22
2.9. Agrupación por tuplas de un peak de audio para generar fingerprints. 22

3.1. Representación matricial para espectrograma. 29
3.2. Representación matricial de máximos locales. 30
3.3. Representación matricial de peaks. 31
3.4. Diagrama de ejemplo de tres archivos de audio con sus rerpectivos fingerprints. 34
3.5. Histograma con el número de repeticiones para selección de solución en Dejavu. 36
3.6. Tiempo de creación de base de datos en función del número de canciones con la configuración original de Dejavu. 39
3.7. Fingerprints en función de diversos valores de configuración. 40
3.8. Tiempo de creación de base de datos en función del número de canciones con la nueva configuración de Dejavu. 43
3.9. Tiempo de respuesta en función de la duración de los extractos de música. 44
3.10. Comparación de tiempos de respuesta en función de la cantidad de canciones de la base de datos. 45

5.1. Clasificación de respuestas al fiscalizar 1 radio online. 58
5.2. Valores de confianza de los aciertos al fiscalizar 1 radio online. 59
5.3. Valores de confianza de identificaciones incorrectas al fiscalizar 1 radio online. 59
5.4. Tiempo de respuesta de los aciertos al fiscalizar 1 radio online. 60
5.5. Tiempo de respuesta de identificaciones incorrectas al fiscalizar 1 radio online. 61
5.6. Valores de confianza de los aciertos al fiscalizar múltiples radios personalizadas. 62
5.7. Tiempos de respuesta de los aciertos al fiscalizar múltiples radios personalizadas. 63
5.8. Uso de memoria al fiscalizar múltiples radios personalizadas. 64
5.9. Tiempos de respuesta al fiscalizar múltiples radios nacionales. 65
5.10. Uso de memoria al fiscalizar múltiples radios nacionales. 66

Introducción

Tras ocho años de discusión en el Congreso, el 18 de abril de 2015 se publicó la ley 20810 donde se declara que "[. . .] las radioemisoras que operen concesiones de radiodifusión sonora, en su programación diaria deberán emitir al menos una quinta parte (20 %) de música nacional, medida sobre el total de canciones emitidas, distribuida durante la jornada diaria de transmisión de cada emisora"[1]. Desde entonces, las diversas radios del país se vieron en la necesidad de adecuar sus parillas de programación para cumplir con la nueva disposición.

La propuesta fue promulgada con el fin de promover la diversidad y potenciar la industria musical chilena, buscando incrementar la presencia de este tipo de obras en la programación diaria de las radios, como una forma de fortalecer su demanda por parte de la población. Puesto que antes de esta moción, era común que quienes trataban de mantener el folclor en los medios radiales, como asociaciones de periodistas y de difusores de radio regionales, no lograban mantenerlas constantemente al aire por no tener auspiciadores.

Como se puede inferir, como resultado de esta ley, las dos contrapartes influyentes en este escenario musical tomaron diferentes posturas. Por un lado está la Sociedad Chilena del Derecho de Autor (SCD), una sociedad de gestión colectiva, que tiene por objetivo administrar derechos autorales generados por la utilización de obras musicales y fonogramas, vale decir, recaudación de licencias por producciones musicales en cualquiera de sus formatos [2]. En palabras simples, es el ente encargado de distribuir las ganancias a sus artistas afiliados, por lo que estuvieron a favor de la promulgación. Según el comunicado realizado un año antes de la aceptación de esta ley, afirmaban que "[...] fomentar la música chilena también es ampliar los espacios para su difusión, algo que requiere del compromiso de todos los actores involucrados en la industria nacional. Cabe recordar que el espectro radioeléctrico es un bien de uso público, un espacio normado por el Estado, que pertenece a todos los chilenos."[3]

No obstante, muchos de los artistas asociados a la SCD confiesa no saber cómo se realiza la gestión del proceso de derechos de autor sobre sus canciones, es más, según un estudio realizado por Open Business "muchas veces el público, los usuarios o los mismos artistas, no saben con precisión en qué consiste el trabajo de esta entidad, especialmente en un país donde ésta posee gran notoriedad pública por temas que no se relacionan directamente con la gestión que ella realiza"[4]. Esto conlleva a generar cierta incertidumbre

en los artistas sobre sus ganancias, pues no es factible fiscalizar las más de 30 radioemisoras locales para determinar cuál es la cantidad de reproducciones de sus canciones y llevar un control de la emisión radial de sus obras.

Por su parte, la Asociación de Radiodifusores de Chile (ARCHI) que agrupa a la gran mayoría de las radios nacionales, apelan a que géneros musicales cómo el soul, electrónica, merengue, reggaetón, lounge, sinfónico o rock progresivo, no tienen artistas o exponentes relevantes en el ámbito nacional, por consiguiente, oyentes que buscan estos estilos emigrarán hacia otros servicios de emisión musical, como es el caso del streaming. Lo que provocaría una disminución considerable en la cantidad de auditores que escuchan transmisiones radiales, puesto que en palabras de la asociación, con la ley "[...] se impone a las radios incluir canciones en su parrilla programática por obligación, vulnerando la libertad de expresión"[5]. Esclareciendo que "No se trata de una ley para la música nacional, sino de una ley para algunos músicos chilenos".

Un factor relevante a considerar es que esta ley no resuelve la situación de los músicos independientes, aquellos que no son patrocinados por un sello, no son afiliados a la SCD, y que en sísntesis, todo su trabajo debe basarse en la autogestión. El no contar con una asociación de respaldo con el respectivo marketing que los de a conocer, los pone en considerable desventaja en contraposición a lo que ocurre con los músicos consagrados y más conocidos. Éstos últimos son favorecidos dado que sus canciones se programarán una y otra vez, puesto que las radios, al velar por sus intereses, apostarán por éxitos indiscutibles antes que canciones nuevas y desconocidas.

Otro aspecto que ha generado controversia es la inexistencia de una entidad fiscalizadora que se encargue de monitorear el cumplimiento del porcentaje, puesto que no ha quedado estipulado en la legislación. Según fue explicado por la ministra de Cultura, "no hay un ente fiscalizador, cualquier persona puede hacer la denuncia frente a un juzgado civil y aportar los antecedentes". Pese a esto, la mayoría de la población asume que este labor es atribución de la SCD, quienes a su vez, afirman que no cuentan con un equipo que monitorice e impulse denuncias por vía judicial.

Es así que en relación a estas circunstancias, en el presente trabajo se realiza un análisis y diseño del prototipo de una plataforma que permita a los artistas chilenos, independiente del tipo de afiliación, llevar un control del uso de su música a nivel radial. Para lograrlo, se comienza la investigación abarcando las tecnologías que permiten implementar programas de reconocimiento acústico, con el fin de monitorear simultáneamente las transmisiones radiales.

Para tal efecto, se abarcará el estudio del concepto fundamental que permite poner en funcionamiento este tipo de solución, la huella digital acústica (Audio Fingerprint). Es un mecanismo generalmente utilizado para enfrentar la copia no autorizada de contenido y así promover la defensa de los derechos de autor, útil para las personas que consumen, utilizan y crean obras musicales. Pues se debe considerar que al ser capaz de identificar canciones desconocidas, un sistema que maneje este tipo de huellas puede ser utilizada para determinar la propiedad del contenido y rastrear cómo lo música es ofrecida a los consumidores a través

de radios, u otro tipo de distribución digital. De esta forma, el monitoreo de difusión masiva (Broadcast monitoring) se puede utilizar para crear una lista del audio transmitido por una estación de radio, con el propósito de garantizar las regalías pagadas a los artistas cuya música es reproducida.

Organización del documento

La estructura de esta memoria es la siguiente:

- **Capítulo 1** detalla la propuesta a desarrollar, las razones para hacerlo, y los objetivos que se desean cumplir con ello.
- **Capítulo 2** comienza con una descripción básica del procesamiento y digitalización de señales acústicas, para continuar con una indagación en la literatura, comprendiendo principalmente las metodologías que existen para el reconocimiento de canciones, y los servicios y/o algoritmos más conocidos.
- **Capítulo 3** plantea el diseño inicial de la solución, explicando la metodología para desarrollar la propuesta del prototipo y detalla la forma en que se llevó a cabo la implementación del algoritmo de monitoreo.
- **Capítulo 4** señala las tareas realizadas para comprobar el rendimiento de la plataforma, incluyendo el estudio en diferentes escenarios.
- **Capítulo 5** muestra los resultados obtenidos tras llevar a cabo la etapa anterior, analizando los valores recopilados.
- Para finalizar, la **Conclusión** resume el trabajo presentado, se comprueba el cumplimiento de los objetivos planteados, y se identifican los posibles trabajos a futuro.

1 | Definición del problema

En este capítulo se exponen las claves principales para comprender la repercusión de la ley 20810, con el fin de contextualizar al lector la razón para desarrollar una plataforma de monitoreo de radioemisiones online. Se aborda con posterioridad los objetivos del presente trabajo de titulación, que permiten englobar los alcances de la solución a implementar.

1.1. Ley 20810

La ley del 20 % como es conocida coloquialmente, estipula el porcentaje de música nacional que debe ser emitido obligatoriamente por radios locales, cantidad que esperan los ministros, sea superada por iniciación propia según el compromiso de cada estación radial. Asimismo, se trata de una medida inmediata, es decir, una vez promulgada no habrá plazo para adecuarse paulatinamente a los cambios, puesto que el plan de ajuste de dos años que había sido considerado en sus inicios, fue rechazado. De esta forma, la reprogramación de parillas musicales para adaptarse a la nueva normativa debía llevarse a cabo de un día para otro.

Referente a las multas por incumplimiento de esta disposición, los valores se encuentran entre 5 y 50 UTM, monto que puede ascender en caso de repetirse la falta. Aunque, si bien existen castigos por infringir la ley, la única manera de comprobar si las radios están cumpliendo es a través de un certificado emitido por lo mismos medios, ya que no existe software controlador, como es el caso del sistema de pago a los músicos chilenos contratado por la SCD. Es más, definen que la ley 20810 es una norma donde cualquier ciudadano puede presentar un requerimiento al advertir o sospechar que esta obligación no se está cumpliendo.

Concerniente al horario en que este porcentaje debe ser emitido, establece que la proporción de música nacional debe estar repartida a lo largo del día, con el fin de prevenir que las estaciones concentren este tipo de música en horarios de baja audiencia. De esta manera, la cuota de canciones acumuladas entre las 22:00 horas y las 6:00 de la mañana, no puede superar el 50 %. Dentro de este marco, el cumplimiento de la ley abarca la reproducción de música chilena en otros idiomas, como también la puesta en aire de programas de difusión de música u otras expresiones culturales, de compositores, artistas o creadores indígenas, para

cumplir con la asignación solicitada.

Para finalizar, se debe destacar que del 20 % en cuestión, un 25 % de la música chilena transmitida debe estar destinado a composiciones o interpretaciones musicales emergentes, obras que para entrar en esta categoría, deben haber sido grabadas en los últimos tres años contados desde la fecha en que se emite; corresponder a composiciones o interpretaciones de identificación local o regional, según el área de extensión geográfica donde se propaga el servicio.

1.2. Descripción del problema

Una vez conocidos los alcances de la ley, en necesario recalcar que en Chile, la Sociedad del Derecho de Autor es el ente que se encarga de administrar los derechos de propiedad intelectual generados por la utilización de obras y fonogramas musicales, por tanto, su gestión se responsabiliza de determinar los ingresos que los artistas reciben por el uso de sus creaciones musicales. El modo de operar se basa en la Ley de Propiedad Intelectual, la cual dispone que todas las radios y canales de televisión deben entregar a la sociedad sus planillas de ejecución, que reúnen la información de cada obra que comunican a través de sus programaciones.

Por su parte, la SCD señala en su página web que la distribución de los derechos se basa en "[...] una muestra aleatoria estadística de aproximadamente 6 días de emisión por cada mes completo enviado por las radioemisoras. Adicionalmente a la muestra aplicada a este rubro de reparto, se incluyen las obras (canciones) difundidas los días 17,18 y 19 de septiembre y 24, 25 y 31 de diciembre de cada año [...]", apoyándose además del Software de reconocimiento de música Vericast de BMAT "que tiene como objetivo aumentar la información que es emitida por las radioemisoras que cuentan con señal online [...]". [6]

Cabe destacar que la licencia para el uso del software Vericast fue adquirida a partir de la convocatoria de licitación pública realizada por el Concejo Nacional de las Artes (CNCA), bajo el programa de Medición Radial, difundido por el gobierno de Chile para el fomento de la música nacional. Según este documento se requería contar con un proyecto que abarcara la compra e implementación de un sistema de medición radial que:

I Recoja la programación de alrededor de 100 señales en todo Chile, en los horarios de transmisión.

II Permita el aumento de las señales en forma paulatina hasta llegar a un total de 400 señales.

III La información esté disponible en forma permanente para el CNCA.

IV La información requerida correspondrá al menos a: identificación del autor, de la canción ejecutada; del artista que interpreta la canción y canciones no identificadas; obtener directamente de la fuente, la información del uso de la música.

V Contemple garantía y soporte total.

El presupuesto del proyecto abarcaba setenta millones de pesos aportados por el consejo de fomento de la música nacional, y debía contemplarse por parte de los postulantes, el financiamiento del costo total con aportes propios o de terceros [7]. De esta forma, la SCD se adjudicó el proyecto, proponiendo la contratación del servicio de Vericast, y la puesta en marcha de su implementación hasta finales del año 2016. Las características principales de la propuesta son señaladas a continuación.

I El proyecto contempla la adquisición de un sistema que realiza un reconocimiento automático de las obras musicales emitidas por las radioemisoras nacionales, brindando reportes fidedignos de las emisiones de canciones de las radioemisoras nacionales indicando qué canciones son emitidas y cuántas veces, en cada una de las radioemisoras monitoreadas.

II El proyecto adjudicado contempla la contratación, por parte del Adjudicatario, por un plazo de 5 años, de un sistema de monitoreo radial provisto por la empresa española BMAT, entidad que presta servicios para la búsqueda y recomendación de canciones, así como para su monitoreo. El sistema que se adquirirá se denomina VERICAST y consiste en un sistema de identificación global de música. Cabe destacar que, para lo anterior, existirá un operador local encargado de administrar el sistema e incorporar las nuevas canciones nacionales.

III El costo total del proyecto es de $290.850.000 (doscientos noventa millones ochocientos cincuenta mil pesos), de los cuales $70.000.000 (setenta millones de pesos) serán aportados por el Consejo. Dicha suma será entregada por el Consejo en una sola cuota, dentro de los 10 días siguientes a la fecha de total tramitación de la resolución administrativa. El resto de los recursos deberán ser aportados por el Adjudicatario, ya sea con recursos propios o de terceros.

No obstante, el primer balance tras un año de la promulgación de la ley reveló ciertos problemas a los que se vieron enfrentados los actores de la industria. Algunos directores de radios establecieron que en algunas ocasiones se programaron artistas y bandas locales que no figuran en los registros de la SCD provocando que no fueran reconocidos por el sistema: "Si bien hace algunos meses la SCD puso a disposición de las radios una base de datos digitalizada de música nacional, con el fin de complementar catálogos -en especial para las radios regionales-, el problema persiste y con esto muchas veces las cifras del organismo no cuadran con los balances internos de cada señal [...]". [8]

Como ha sido detallado con anterioridad, el sistema en cuestión que se encarga del monitoreo radial consiste en un software que compara la música que emiten las radios con una base de datos que no tiene la totalidad de la música que se difunde, como los mismos socios de la SCD lo reconocen. Es más, hay decenas de sellos independientes que no pertenecen ni han registrado sus obras en la SCD, y cientos de artistas en la misma situación; ya que es justamente en el mundo de las compañías discográficas autónomas y de los artistas autogestionados donde se ubica una parte esencial de la música emergente. Inclusive, cabe

mencionar que el Consejo Nacional de la Cultura y las Artes financia solo el 7 % de la producción nacional. Llevando estas cantidades a números reales, se puede realizar la comparación de que apenas tres discos de los aproximadamente mil álbumes editados en 2014 fueron costeados por los grandes sellos multinacionales [9].

En consonancia con estos inconvenientes, se suma un tópico que, a dos años de la promulgación de la ley, sigue siendo tarea pendiente. El punto en cuestión consiste en que los artistas musicales, productores independientes y radios nacionales, están marginados de conocer el número real y detalle de las radios fiscalizadas; distribución efectiva de derechos entre todos los titulares, y acceso a esta base de datos que aglomera el control de las transmisiones. Especialmente porque no ha existido una campaña de información pública para la inclusión de productores y artistas autónomos a la plataforma, cuya utilización efectiva, que considera tanto descargas digitales como ventas físicas, es mayor que esta rotación radial incompleta.

Teniendo en cuenta estos escenarios, surge la idea de diseñar y desarrollar una prototipo de plataforma nacional, que permita fiscalizar en una primera etapa un rango menor de radioemisoras locales, de tal forma de comprobar de manera automática las parrillas musicales de cada una de éstas. De esta modo, la información recopilada de la programación radial permitiría a los artistas chilenos no asociados a la SCD, llevar un control del uso de su música a nivel radial.

1.3. Objetivos

1.3.1. Objetivo general

- Detectar la frecuencia de reproducción de canciones de artistas chilenos en radioemisoras online.

1.3.2. Objetivos secundarios

- Almacenar un catálogo de prueba de canciones nacionales en una base de datos.
- Diseñar e implementar un sistema de reconocimiento acústico para analizar en paralelo un conjunto de radioemisoras online.
- Implementación de radioemisoras para validar solución desarrollada.

2 | Estado del arte

Gracias a la influencia y avances de las tecnologías, la creación de emisoras que transmiten a través de internet han ido en aumento en los últimos años, facilitando a los usuarios el acceso a este tipo de contenido, puesto que las emisoras online no requieren invertir en equipos radiales análogos, antenas, repetidores, o número de frecuencia para ser sintonizada. Esta nueva forma de hacer radiofusión es más barato y sencillo, logrando abarcar cobertura mundial.

Actualmente, cualquier usuario que tenga un equipo con los mínimos requerimientos es apto para grabar y transmitir una radio online, aunque, si bien no hay necesidad de contar con permiso o concesión adicional, la única normativa legal que se debe tener en consideración es pagar por los derechos de autor de las canciones que se pongan al aire.

Esta flexibilidad a la hora de transmitir música, junto a la inexistencia de una entidad reguladora o una organización gremial como lo es la ARCHI para las emisoras análogas, ocasiona que fiscalizar estas emisiones requieran de un mayor trabajo.

Con el objetivo de desarrollar una plataforma que facilite este labor, se hará una revisión bibliográfica para comprender los métodos que actualmente se utilizan en el mercado para este tipo de control. En primera instancia, se agruparon diversos sistemas desarrollados ya sea para el cobro, o simple gestión de los derechos de propiedad intelectual. Posteriormente, se toma un enfoque más técnico, para analizar cuales son los algoritmos claves que dieron pie al desarrollo de aplicaciones de reconocimiento acústico. Y para finalizar, se detallan sistemas que basándose en los algoritmos anteriores, han surgido como servicios de reconocimiento musical.

2.1. Propiedad intelectual

Se entiende por propiedad intelecual, según lo establecido por la entidad especializada dentro del sistema de las Naciones Unidas, la OMPI, los derechos relativos a [10]:

- Obras literarias, artísticas y científicas es posible aceder de manera.

- Interpretaciones de los artistas intérpretes y a las ejecuciones de los artistas ejecutantes, a los fonogramas y a las emisiones de radiodifusión.
- Invenciones en todos los campos de la actividad humana.
- Descubrimientos científicos.
- Dibujos y modelos industriales.
- Marcas de fábrica, de comercio y de servicio, así como a los nombres y denominaciones comerciales.
- Protección contra la competencia desleal, y todos los demás derechos relativos a la actividad intelectual en los terrenos industrial, científico, literario y artístico.

En palabras simples, y reduciendo la definición al contexto de esta investigación, la propiedad intelectual es la parte legal que da sentido a las carreras musicales, puesto que de no existir, todos podrían interpretar, distribuir, producir, o realizar cualquier otro tipo de actividad, sin solicitar permiso para utilizar estas creaciones.

La Propiedad Intelecutal, a su vez, se divide en dos categorías, la propiedad industrial, correspondiente a un derecho de exclusividad que otorga y reconoce el Estado, protegiendo a los artistas de usurpación, competencia desleal y falsificación, pues comprende las patentes, marcas, diseños industriales e indicaciones geográficas.

En segundo lugar está el derecho de autor. Este último abarca los derechos de los artistas intérpretes y ejecutantes sobre sus interpretaciones o ejecuciones, los de los productores de fonogramas sobre sus grabaciones y los de los organismos de radiodifusión respecto de sus programas de radio y televisión, evitando así, que toda persona ajena a la creación pueda hacer uso de la idea, sin haber recibido consentimiento legal.

Específicamente en Chile, fue durante la década de los noventa que se favoreció el reconocimiento legal de las sociedades de autores o sociedades de gestión colectiva que promovieron el ejercicio del derecho de autor [11], recaudando y distribuyendo eficientemente los ingresos generados por otorgamiento de licencias de uso de las obras. Sin embargo, con el correr de los años, la piratería, la descarga, y reproducción de obras a través de internet han desencadenado nuevos desafíos, pues con los nuevos niveles de desarrollo, es posible acceder de manera ilegal a una gran variedad de obras y distribuirlas instantáneamente a nivel mundial. Dicho fenómeno genera problemáticas a la tarea de fiscalización de los derechos de autor debido a que, en la práctica, la reproducción y uso de las creaciones artísticas evaden el pago de los derechos de autor correspondientes.

Para comprender las estrategias que hoy en día se utilizan como mecanismos de defensa de los derechos de autor, se ha recopilado información de tres sistemas que regulan estos conceptos, los que se detallan a continuación.

2.1.1. ContentID - Youtube

Esta herramienta utilizada por YouTube y autorizada por la compañía Audible Magic, permite detectar a través de diversos algoritmos si un vídeo con derechos de autor ha sido publicado y divulgando por usuarios no autorizados, gracias a la comparación con una base de datos de vídeos previamente suministrados por el poseedor de los derechos. Así, su principal funcionalidad es escanear los videos que se suben a la plataforma, de tal forma que si se encuentra coincidencia en contenido con derechos de música, o imagen, notifica al usuarios poseedor del derecho legal sobre la situación, siendo éste, el encargado de decidir que acciones se llevarán a cabo con el video detectado [12]. Actualmente las opciones disponibles son:

- Bloquear un video completo para que no se pueda ver.
- Monetizar el video a través de anuncios y, en algunos casos, compartir ingresos con la persona que sube el video.
- Seguir las estadísticas de reproducción del video.

Adicionalmente, es posible que el estado de la cuenta del usuario sea considerada bajo vigilancia, y en ese caso, dejará de tener acceso a muchas funcionalidades de YouTube.

En contraparte, existen acciones que el usuario reclamado puede realizar, ya sea silenciando el audio, o de lleno, eliminar el vídeo infractor, teniendo en consideración que esto último no suprime la reclamación. Ahora bien, aunque todas estas gestiones y acciones se realizan dentro del marco de YouTube, los demandantes podrían iniciar un proceso legal externo a la plataforma de Google, formalizando una demanda.

2.1.2. Sistema de identificación de SoundCloud

Esta red social utilizada principalmente por músicos, es una plataforma de distribución de audio, donde los usuarios pueden colaborar y promocionar sus creaciones, difundiendo su música a través de canales, de tal forma, que se envía el enlace de una canción y los contactos la reciben en forma de reproductor de audio. El objetivo fundamental de SoundCloud es ofrecer música que pueda comercializase sin las complicaciones de necesitar servicios de almacenamiento en línea.

Durante el transcurso del 2011 esta plataforma puso marcha blanca a un nuevo sistema automático de identificación de contenido, para reconocer el audio que los titulares de derechos han solicitado que se retire de SoundCloud [13]. Ya que en sus inicios la única validación para el resguardo de derechos de propiedad intelectual era realizado por los mismos usuarios, que al reconocer extractos de otros artistas en las creaciones que escuchaban, podían dar aviso de esta situación vía notificación.

De esta forma, si los usuarios cargan archivos de audio con extractos de música que son propiedad de los titulares de los derechos, se oculta el archivo infractor, enviando un correo electrónico al usuario

esclareciendo los pasos a seguir. Este mensaje contiene información sobre el titular que, en la mayoría de los casos, es una empresa discográfica o editorial.

Eventualmente, el archivo puede ser marcado por error si el propietario del canal cuenta con el permiso de distribución, y en tal caso, es posible abrir una disputa para volver a mostrar el audio en cuestión, con todos los comentarios y estadísticas que el perfil tenía antes del acontecimiento.

2.1.3. Identificador de Twitch

La plataforma que ofrece un servicio de srteaming de videos, propiedad de Amazon, también se ha asociado con la compañía Audible Magic para implementar su servicio de reconocimiento de información para el cumplimiento de derechos de autor. Como se puede inferir, Twitch escanea la transmisión de los videos para determinar si éstas presentan audio de terceros o música ambientada no autorizada. Una vez que esta revisión identifica un video grabado que infringe un reclamo de derechos de autor, el audio del video es silenciado por media hora.

Destaca de este servicio de reproducción en línea, la creación de su propia biblioteca musical [14], que contempla una lista de pistas dejadas a disposición de los locutores de Twitch para que sean utilizadas exclusivamente en las transmisiones de la plataforma. De esta forma, es posible salvaguardar los derechos de autor sobre creaciones de terceros, sin afectar la transmisiones en vivo, de tal forma de proteger tanto a los locutores, manteniendo su fidelidad y preferencia, y también a los propietarios de las canciones.

2.1.4. Monitec

Monitec es una compañía costarricense que se especializa en el desarrollo y exportación de tecnologías aptas para la inspección de frecuencias de radio, televisión y cable, para el monitoreo de medios [15].

Su sistema contempla el hardware encargado de la captura y digitalización de los diferentes medios electrónicos y acústicos, la que posteriormente es procesada al ser captada y comparada por su software. Con el objeto de monitorear las canciones y anuncios publicitarios que se quieran detectar, utilizan los patrones digitales como base para recorrer, en tiempo real, la señal de radio o televisión que captan sus antenas en diferentes frecuencias.

Esta tecnología, a su vez, posibilita su utilización desde la web, por lo que permite distribuir los elementos que captura por diferentes geografías dentro de un mismo país, al desplegar reportes en diferentes formatos de visualización. De acuerdo a este tipo de información, las disqueras, grupos radiales, empresas publicitarias y artistas del país, pueden saber cuáles obras se están posicionando mejor en ciertas emisoras, cuántas veces o con cuánta calidad se escuchó un anuncio, o en el caso de las sociedades de gestión de cobro de derechos de autor, para confirmar y certificar el uso que se hace de las creaciones y realizar cobros

pertinentes. Tal es el avance de esta empresa en el rubro, que Monitec cuenta con un servicio de monitoreo de música en locales comerciales, al ofrecer el arriendo de dispositivos que se conecta el sistema de audio interno del recinto, junto a una conexión a internet para el envío de datos.

2.1.5. Vericast

Este servicio de identificación de música de rango mundial, perteneciente a la empresa española BMAT, monitoriza millones de canciones en más de tres mil radios y televisiones de más de sesenta países. La solución implementada por esta compañía ofrece una identificación a tiempo real [16], con la posibilidad de generar reportes auditables, basando su funcionamiento en el cotejamiento de los patrones digitales del audio con su extensa base de datos. El algoritmo utilizado es capaz de resistir las alteraciones de la señal, como en un canal degradado o ruidoso, o cuando una canción es utilizada de fondo y no transmitida directamente.

Es necesario destacar, como fue explicado con anterioridad, que este software de origen español, cuyas bases de datos se encuentran centralizadas en servidores del proveedor BMAT en Barcelona, es ampliamente utilizado tanto en América Latina como en Europa, por muy diversos usuarios. Específicamente en Chile, es la SCD y el CNCA los organismos que utilizan Vericast para monitorear actualmente más de 300 radios en todo Chile.

2.2. Estructura y caracterización de la señal acústica

Con la recopilación anterior de aplicaciones que se especializan en el monitoreo de derechos de autor, fue posible tener un primer acercamiento a las tecnologías que se utilizan para identificación musical. Sin embargo, antes de abordar el funcionamiento y especificaciones de los algoritmos de reconocimiento acústico, en necesario detallar la forma en que la música o señales de audio en nuestro entorno son transformadas para ser almacenadas en un sistema computacional.

El sonido es una vibración que se propaga como una onda mecánica, y cuando ésta llega al oído humano, genera compresión en el medio que lo rodea. De esta forma, las alteraciones de presión transmiten la energía modulada en forma de impulso nervioso hasta el cerebro, el cual se encarga de procesar esa información captada.

Por su parte, los dispositivos de grabación imitan este proceso con el fin de transformar la onda de sonido en una señal eléctrica. Sin embargo, esta señal continua debe ser procesada y convertida en una señal discreta para poder se almacenada digitalmente. Con este fin, el procedimiento para hacer esa transformación comienza con el muestreo, de forma que un convertidor analógico digital realiza múltiples conversiones de la señal en pedazos más pequeños, obteniéndose así, una matriz de bytes que representa la variación de la presión con respecto al tiempo.

En la imagen 2.1 se observa una representación sencilla de lo que significa hacer la digitalización de una señal analógica. Básicamente consiste en obtener varias mediciones sucesivas para almacenarlas en código binario. De esto se desprende el concepto de Frecuencia de muestreo, definido como la cantidad de muestras de audio que se toman en un segundo.

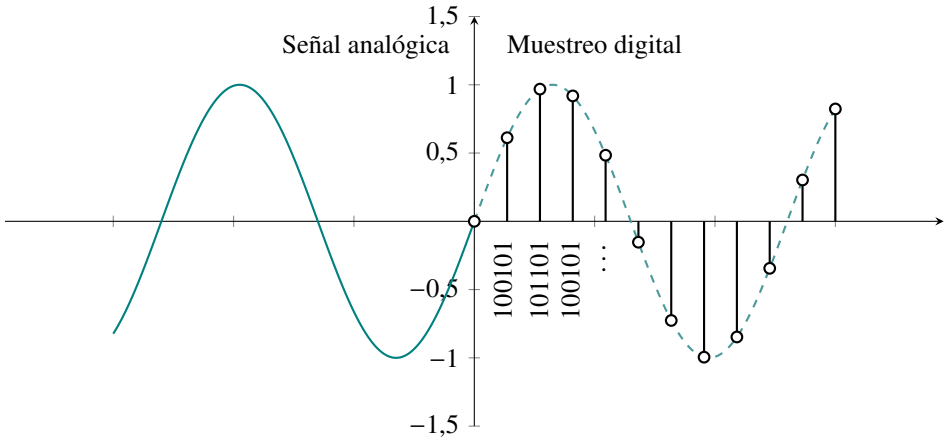

Figura 2.1: Muestreo acústico de señal analógica.
La imagen simula la toma de muestras de una onda de sonido. Las cantidades binarias no representan valores reales.

Si bien cabe esperar que a mayor cantidad de muestras mayor fidelidad tendrá el audio grabado, esto trae la desventaja de requerir mayor espacio de almacenamiento. Adicionalmente, se debe tener presente que el oído humano solo es capaz de procesar sonidos en un determinado rango de frecuencias, por lo que tomar un elevado número de pruebas conllevaría a un malgasto de recursos, al reunir información auditiva que las personas no serían capaces de oír.

El valor estándar para la mayoría de los proyectos musicales es de 44100 muestras por segundo. Dicho valor proviene del teorema de muestreo de Nyquist-Shannon, pero debido al alcance de esta memoria, abordar minuciosamente su descripción no es requerido para comprender el funcionamiento de los algoritmos. Basta con señalar que este teorema matemático establece que para no notar saltos en la continuidad de la señal, y no perder frecuencias, es necesario tomar el doble de muestras de la frecuencia máxima que el promedio de las personas puede distinguir. De esta forma, es posible reconstruir la señal original captando los máximos sin saltos y sin perdida en la percepción del oído.

Considerando que el ser humano es incapaz de oír frecuencias por sobre los 20,000 $[Hz]$, la norma establece que un límite de frecuencia máximo es 22,500 $[Hz]$, por lo que utilizando el teorema de Nyquist-Shannon, es posible obtener el valor de las 44100 muestras por segundo:

$$Muestras\ por\ segundo = Frecuencia\ alta * 2 = 22500 * 2 = 44100.$$

Una vez realizado el muestro de la onda acústica, es posible interpretarla gráficamente mediante

un oscilograma (véase figura 2.7.a). Esta representación de la onda en función del tiempo, conocida como dominio temporal, brinda las amplitudes de la señal en los instantes en los que se hizo el muestreo. No obstante, para la mayoría de los casos es más conveniente conocer el comportamiento de la frecuencia de la señal que se está estudiando, vale decir, la cantidad de oscilaciones completas que hay en un determinado intervalo de tiempo, y que en el ámbito audiovisual, es conocido como dominio frecuencial.

La Transformada Rápida de Fourier (FFT de sus siglas en inglés: Fast Fourier transform) es un método que permite hacer la transición entre los dos dominios, pues el teorema afirma que cualquier forma de onda puede ser representada por la suma acumulada de senos y cosenos, proporcionando una interpretación estática de una señal dinámica. En la figura 2.2 se observan los dos dominios que permiten representar una señal. La función Senoidal de amplitud 1 completa cinco longitudes de onda en un segundo, por ello, en el dominio frecuencial, se alcanza esa amplitud a una frecuencia de 5 $[Hz]$. Asimismo, para una onda de sonido que no será periódica, pero si puede ser transformada y representada por la suma de senos y cosenos, obtendrá su espectro a partir de las frecuencias de las funciones que la componen, tal cual se aprecia en la última imagen de la figura 2.2.

En síntesis, la Transformada de Fourirer es un procedimiento mediante el cual se mapean las señales a una serie infinita de sinusoides, cuyas estructuras estarán determinadas por sus amplitudes y fases respectivas, especificando todas las componentes de frecuencia de una señal con su aporte energético. En la figura 2.3 se observa la representación del dominio frecuencial de la señal denominada Onda Cuadrada. Los valores de cada frecuencia son calculados en base a la composición de funciones senoidales que conforman la señal original.

Adicionalmente, la FFT supone que se está trabajando sobre una onda estacionaria, en otras palabras, al considerarse como una onda periódica, se espera que los dos extremos de la señal presenten continuidad al conectarse, como si se tratara de una tipología circular.

Como en la mayoría de los casos la señal analizada no es un número entero de periodos, asumir esta periodicidad provoca que las muestras del audio tomen una representación truncada con diferentes características a la onda continua original, introduciendo transiciones discontinuas inesperadas. Las imágenes de la figura 2.4 son un claro ejemplo de esta anormalidad, donde las velocidades de muestreo bajas limitan el intervalo de frecuencias que pueden registrarse, lo que puede dar como resultado una grabación que no representa correctamente el sonido original.

El principal inconveniente con esta información inexacta es que las discontinuidades artificiales aparecen como componentes de frecuencias más altas que la de Nyquist, generando versiones distorsionadas de energías que se liberan a otras frecuencias. Este fenómeno de frecuencias adulteradas, conocido como fuga, provoca que las líneas espectrales se difundan con señales más amplias (véase figura 2.4.b).

Para minimizar estos efectos se utilizan las funciones ventana, encargadas de reducir la amplitud

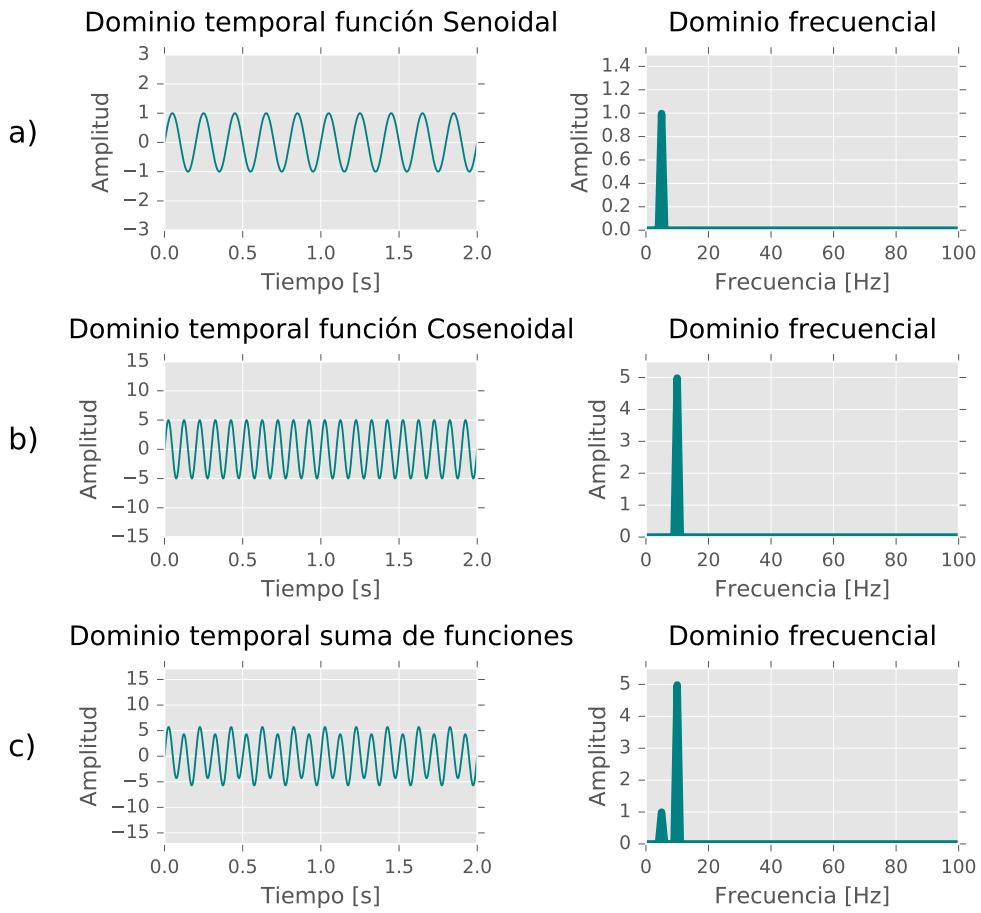

Figura 2.2: Dominio temporal y frecuencial.
Las imágenes de la izquierda representan ondas senoidal y cosenoidales en función del tiempo. A su derecha se encuentra su representación en función de la frecuencia.

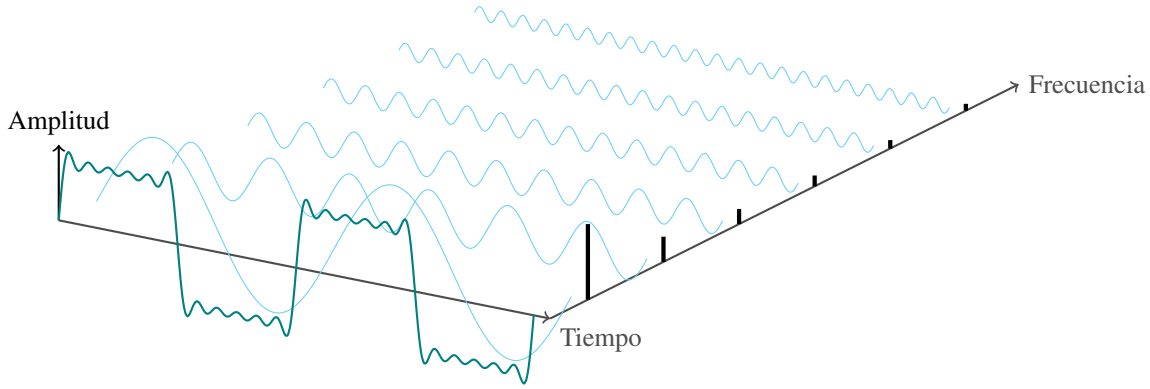

Figura 2.3: Dominio frecuencia de señal Onda Cuadrada.
En la imagen se visualiza la onda Cuadrada original, con sus respectivas descomposiciones en funciones sinoidales. Con estas últimas se generan los gráficos de frecuencia.

de las discontinuidades en los límites de cada secuencia finita que ha sido muestreada. Esto se logra al multiplicar cierto intervalo de tiempo por la función ventana de longitud finita, con una amplitud que varía

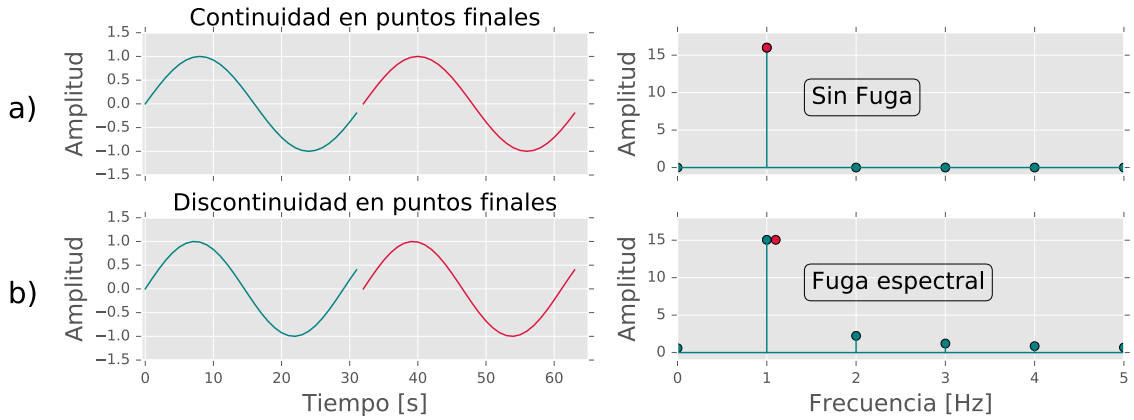

Figura 2.4: Ejemplo de muestreo de señales.
La imagen izquierda de (a) representa dos periodos de una onda, cuyos puntos finales presentan continuidad. A su derecha, se muestra el dominio frecuencial de la señal. En (b), la baja velocidad de muestreo provoca discontinuidad en la señal, distorsionando la onda de sonido original.

paulatinamente a cero en los bordes. En la imagen 2.5 se observa un ejemplo que esclarece el efecto que esta función auxiliar produce.

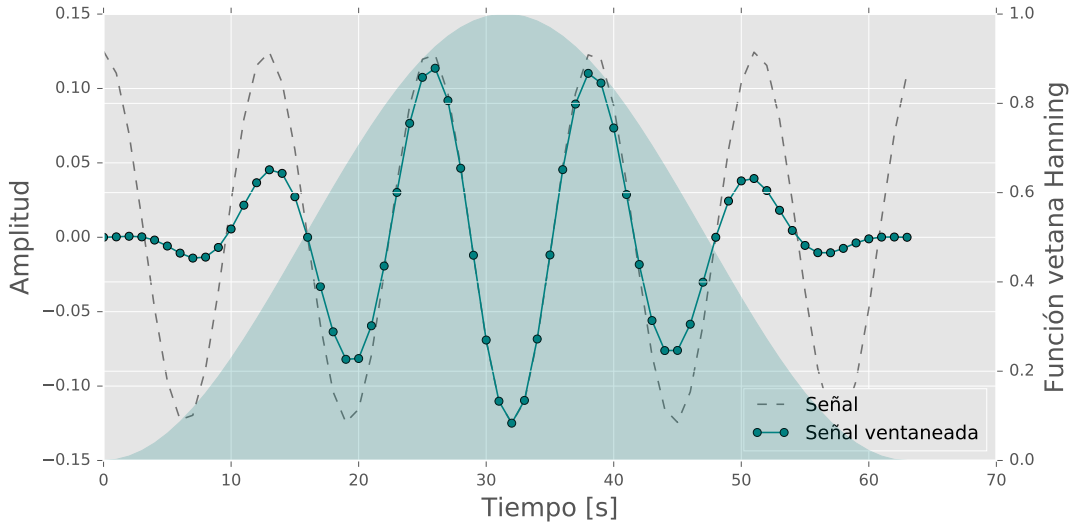

Figura 2.5: Aplicación de función ventana a una señal.
Las ondas superpuestas de la imagen representan el antes y después de aplicar una función ventana, mientras que la zona opaca corresponde a la forma de dicha función. Como se puede observar, los bordes de la muestra tienden a cero luego de su aplicación.

Con esta alteración, se obtiene un resultado de una forma de onda sin transiciones bruscas, puesto que de juntar los extremos de la muestra obtenida ambos se estarían conectando en cero. Básicamente, al implementar una función de ventana temporal $v(t)$, se está dividiendo una señal $s(t)$ en pequeños segmentos a través del tiempo. Con la función temporal es posible encuadrar la onda alrededor de un intervalo τ calculándose su TTF. Posteriormente, la ventana es trasladada para cubrir un nuevo segmento, procedimiento que se repite hasta completar toda la muestra. De esta forma, se analiza la señal desde una ventana estrecha,

de tal forma que la misma parece una onda estacionaria. Esta versión revisada de la Transformada de Fourier es llamada Transformada de Fourier de término reducido (Short Time Fourier Transform STFT).

En la literatura enfocada en el procesamiento digital de audio, existen diversas funciones ventanas cuya aplicación dependerá del tipo de señal y propósito de su estudio, ya que la forma que ésta posea influirá sobre las propiedades que se desean destacar en la transformación. Ya sea para conseguir amplitudes más precisas o minimizar el ruido, la ventana escogida determinará el grado de distorsión que se verá en el espectro. Las funciones más conocidas se listan a continuación, cuyas representaciones gráficas pueden observarse en la figura 2.6.

- **Kaiser-Bessel:** función normalmente utilizada para revelar señales inferiores, cercanas al eje cero que otras ventanas podrían ocultar.
- **Hamming y Hanning:** generalmente conocidas como Hann, se caracterizan por poseer una forma sinusoidal. La diferencia entre ambas recae en que Hanning toca totalmente el eje de las abscisas, no así Hamming, que solo se acerca a cero por ambos extremos. En general, la ventana Hanning es satisfactoria en el 95 % de los casos. Tiene buena resolución de frecuencia, menor fuga espectral, y debido a su velocidad y rendimiento para opacar ruido, es habitualmente usada en comunicaciones.
- **Blackman-Harris:** corresponde a un enfoque más elaborado que las funciones Hann que debido a su forma, sirve de propósito general.
- **Flat Top:** tiene la característica de cruzar el eje cero, y volver a alzarse, ocasionando una aproximación a la amplitud verdadera de la señal, por tanto, es útil cuando se prioriza la medición de la amplitud en el espectro.

Si bien no existe un enfoque general para escoger una función ventana, conocer el tipo de onda estudiada puede guiar en el proceso de selección. La tabla 2.1 presenta las características básicas o contenido fundamental de la señal, con la respectiva función que en la mayoría de los casos genera buenos resultados.

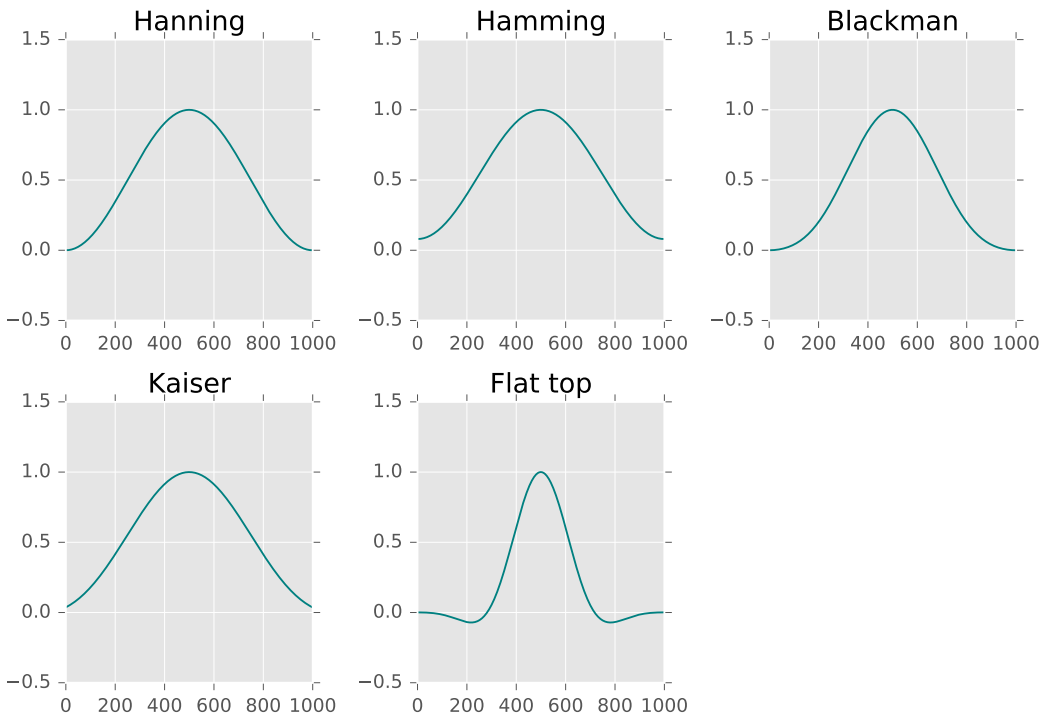

Figura 2.6: Ejemplo de funciones ventana.
Las cinco imágenes representan la forma de diversas funciones ventana que pueden utilizarse para el procesamiento de señales.

Tabla 2.1: Función ventana ideal acorde al tipo de señal.

Tipo de señal	Función ventana
Señales de excitación (ejemplo: golpe de martillo)	Blackman
Señales de respuesta	Exponencial
Onda senoidal con énfasis en la ampliud	Flat Top
Onda senoidal con énfasis en la frecuencia	Flat Top
Medidas exactas de amplitud para un tono	Flat Top
Ondas seno cercanamente espaciadas	Hamming
Onda senoidal o combinación de senoidales	Hanning
Contenido desconocido	Hanning
Señales al azar de ancho de banda (vibración de información)	Hanning
Dos tonos con frecuencias cercanas pero amplitudes muy distintas	Kaiser-Bessel
Ancho de banda al azar (ruido blanco)	Uniforme
Dos tonos con frecuencias cercanas y amplitudes casi equivalentes	Uniforme

El tamaño de la ventana delimita la resolución de la frecuencia y habitualmente es definida como potencia de dos, por ejemplo: 512, 1024, 2048, 4096, etc. Lo ideal es considerar ventanas pequeñas para que dentro de cada una de ésta, la señal pueda considerarse cuasiestacionaria. Empero, a menor tamaño de la ventana, menor es la resolución de la frecuencia, aumentando la dificultad para reconocer entre dos frecuencias similares o identificar en qué momento la señal cambia.

Teniendo en cuenta que la resolución de la frecuencia puede calcularse como la razón entre el número de muestras y el tamaño de la ventana, se ha elaborado la tabla 2.2 para apreciar cómo afecta el tamaño de la función a esta variable. Recordando además que el ser humano percibe ondas entre los $20 - 20000$ $[Hz]$, queda en evidencia que 4096 es el tamaño ideal, si se digitaliza la onda de sonido a 44100 muestras por segundo. A su vez, la duración temporal de esta función es de 0,93 [s], lo que indica que se pueden detectar cambios en la señal cada 93 [ms].

Tabla 2.2: Frecuencia de resolución según el tamaño de la ventana.

Numero de muestras	Tamaño de la ventana	Frecuencia de resolución [Hz]	Duración de la vetana [s]
44100	512	86,13	0,012
44100	1024	43,07	0,023
44100	2048	21,53	0,046
44100	4096	10,76	0,093
44100	8192	5,38	0,186

Una vez se aplica la transformada rápida de Fourier a este conjunto de muestras, es posible generar uno de los elementos más importantes del tratamiento del audio, el espectrograma, una representación visual en dos dimensiones de la amplitud de la señal en función del tiempo y la frecuencia. Como es posible observar de la figura 2.7.b, al aplicar la STFT al conjunto de muestras, y unirlas en una sola matriz, se desprende la amplitud de la señal a una frecuencia particular, donde el color rojo corresponde a amplitudes mayores, en oposición al cian, de amplitudes bajas. Vale decir, si se grabara la señal en un solo tono, el espectrograma creado se apreciaría como una línea recta horizontal en la frecuencia de dicho tono.

Ahora, si bien cabe esperar que este espectrograma sea único para cada canción o audio, se debe tener en consideración que cualquier tipo de ruido que afecte las muestras de grabaciones en el ambiente donde se reproduce la música, provocará que la representación matricial por espectrogramas varíe una infinidad de veces, dependiendo del tipo de ruidos externos que se filtren en la grabación. Por tanto, para los algoritmos de reconocimiento acústico es necesario encontrar la forma de identificar huellas únicas para cada canción, independiente del ruido que exista en los archivos de audio.

Una forma de hacerlo es centrándose en las amplitudes más grandes de una canción, pues debido a sus altos valores, no son afectados por posibles ruidos que pueda contener el archivo de entrada. De esta forma se definen peaks de audio correspondientes al par de variables tiempo y frecuencia, que aluden a alguna amplitud cuyo valor es el más alto dentro de un vecindario de muestras, de tal forma que es posible discretizar la señal de audio en valores enteros a partir de las amplitudes más grandes.

Para lograr lo anterior, es necesario tratar el audio de entrada e identificar los máximos locales, tal como lo señala la imagen 2.8.b. De este modo, se extrae toda aquella información del espectrograma que represente amplitudes altas, por lo que para cierto margen, el ruido de un archivo ya no influye en el algoritmo.

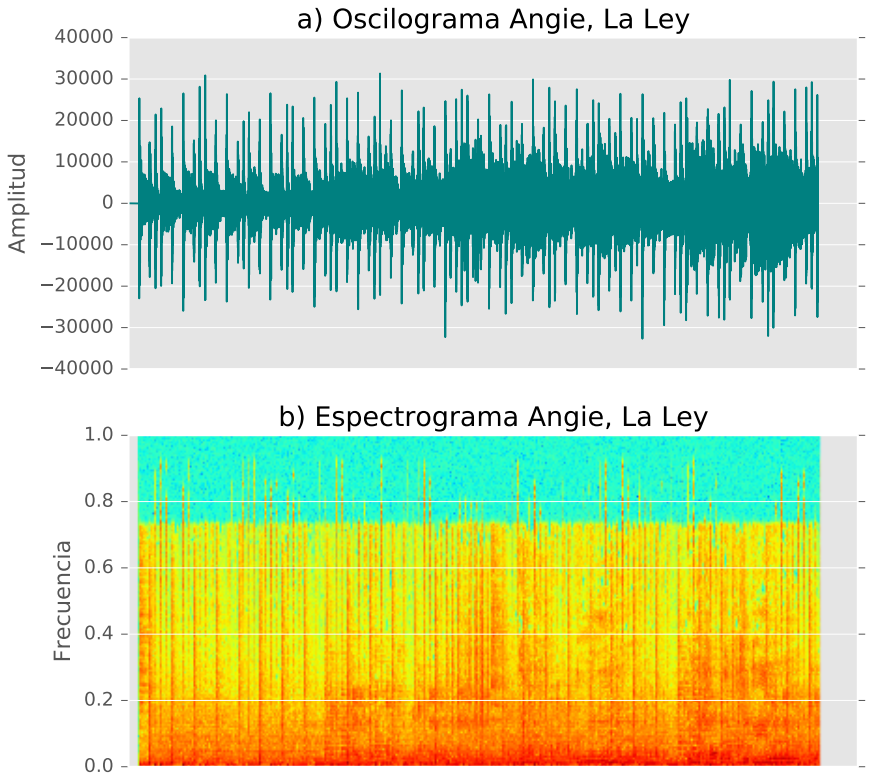

Figura 2.7: Modos de representación visual de un archivo de música.
La figura (a) corresponde a un oscilograma de los primeros 30 segundos de la canción Angie de La Ley que señala la variación de la amplitud respecto al tiempo. La imagen (b) es la representación gráfica de la misma canción, llamada espectrograma, donde la variable dependiente es la frecuencia, en función del tiempo.

No obstante, es posible que más de algún par discreto de [$tiempo, frecuencia$] de alta amplitud en una canción, coincida con pares provenientes de otras canciones, por lo que es necesario volver a tratar la información obtenida. En base a esto surge otro concepto relevante del procesamiento de música, los fingerprints, o huella digital acústica. Ésta se define como el resumen generado a partir de una señal de audio que contempla la utilización de una función hash, para crear a partir de una entrada, una salida alfanumérica que representa la información que le fue dada inicialmente. Por lo tanto, los datos de entrada generan una cadena que solo pueden crearse con esos mismos datos.

Específicamente, la información resumida que contempla un fingerprint corresponde una combinación entre el valor de la frecuencia en un peak del audio, y la diferencia en tiempo, entre algún punto aledaño dentro del vecindario, como se observa en la figura 2.9. De tal forma que al combinar estos peaks en función del tiempo que los separa, es posible identificar elementos en el archivo que son únicos para cada canción.

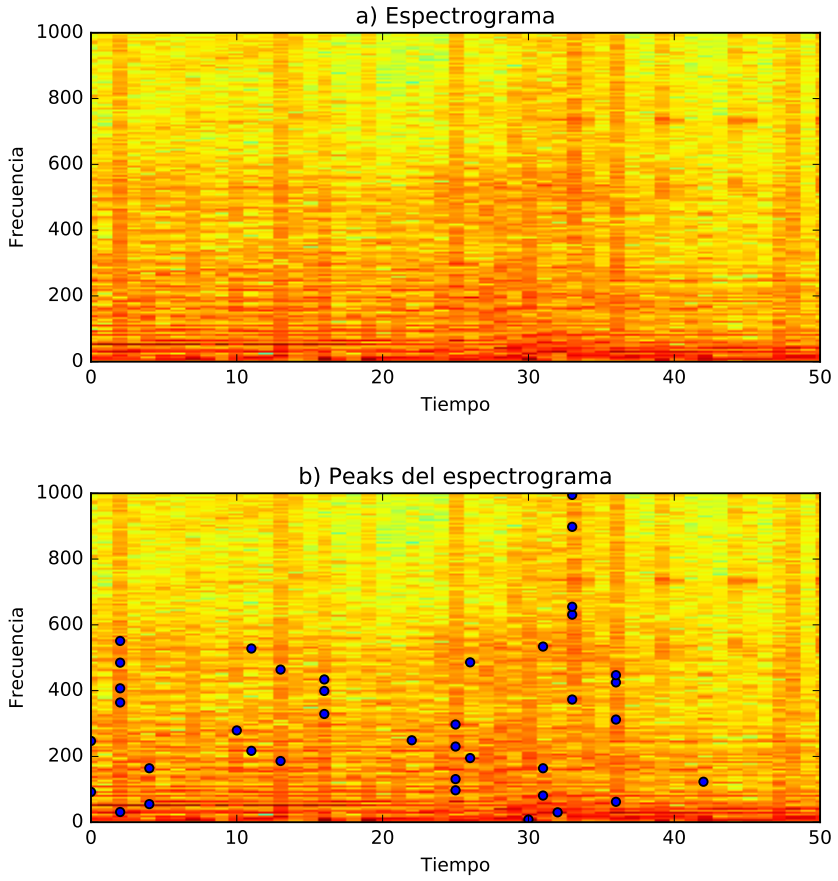

Figura 2.8: Espectrograma con sus peaks de audio.
La imagen (a) corresponde a una porción aumentada del espectrograma de los primeros 30 segundos de la canción Angie de La Ley. En (b) se observa el mismo espectrograma con los peaks de audio, cuyos valores de tiempo-frecuencia son utilizados para la creación de fingerprints.

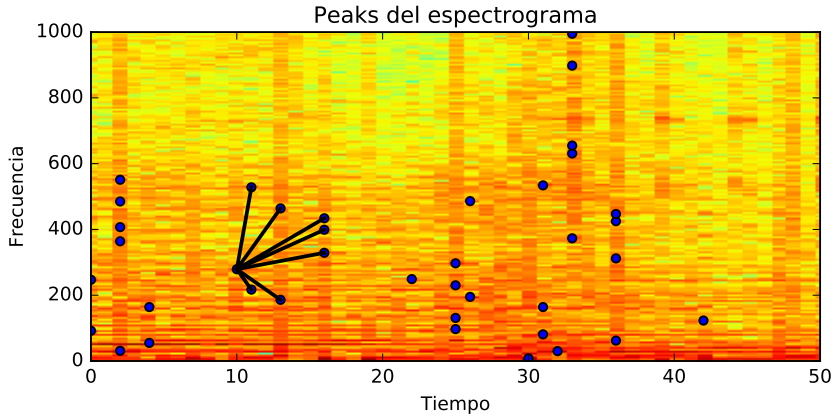

Figura 2.9: Agrupación por tuplas de un peak de audio para generar fingerprints.
Porción aumentada del espectrograma de los primeros treinta segundos de la canción Angie de La Ley, donde se realza uno de los peaks de la canción, junto a otros máximos cercanos, para la generación de fingerprints.

2.3. Algoritmos de reconocimiento acústico

Con la información presentada hasta el momento es posible afirmar que las aplicaciones de reconocimiento acústico basan su funcionamiento en la espectrografía, disciplina que se dedica a medir la frecuencia de los sonidos en un determinado espacio de tiempo. De esta forma, un fragmento de música puede ser representado como una gráfica de frecuencias llamadas espectrogramas, donde en un eje se observa el tiempo, en otro la frecuencia, y la amplitud de las señales o intensidad del sonido se visualiza mediante una escala de grises o de colores.

Otro rasgo en común que comparten los diferentes esquemas de reconocimiento acústico es su estructura general, la cual es dividida en dos secciones fundamentales: extracción de huellas acústicas y coincidencia de huellas acústicas. La primera calcula un conjunto de características intrínsecas del audio de entrada. Estos atributos distintivos son las denominadas huellas acústicas (acoustic fingerprints), las que pueden consistir en una muestra uniforme, o un resumen de los puntos de interés del espectrograma, puesto que los algoritmos de generación de finferprints convierten estas características en códigos numéricos o hashes para facilitar su manejo. Luego de la extracción, los fingerprints de la consulta son utilizados para buscar coincidencias en una base de datos que contiene los múltiples archivios de audio preliminarmente analizados [17]. El número de comparaciones puede ser muy alto, por lo que una buena configuración es primordial.

Dado que la principal funcionalidad a desarrollar para esta memoria decae en el reconocimiento y tratamiento de archivos de audio para realizar el análisis a partir de comparaciones con el catálogo de canciones en una base de datos, se han investigado algunos algoritmos que detallan la forma de obtener estos patrones de reconocimiento.

2.3.1. Obtención de fingerprints en base a características

2.3.1.1. Amplitud

Papaodysseus [18] propone un sistema diseñado para trabajar en difusión radial cuya característica principal es que la señal no posee ruido adicional. El algoritmo funciona dividiendo y enmarcando el audio para aplicar la STFT, y luego, el espectro de cada encuadre se divide en 48 componentes de dominio de frecuencia, o *bins*. De estos últimos, un vector representante de cada ancho de banda es creado, conteniendo el valor 1 si se trata de un peak, o 0 en caso contrario. Posteriormente, los vectores de banda se comaparan al realizar una operación a nivel de bits, determinando que hay coincidencias siempre que los vectores difieran a lo más en 2-3 bits.

Wang [19], por su parte, presenta en su investigación la forma en que el programa de reconocimiento

de música Shazam funciona. El algoritmo describe la forma en que luego de aplicar la STFT al audio, selecciona una grupo de peaks de audio al que denomina *constelación* que se singularizan por contener amplitudes que son más grandes que el resto del área que los rodea. De esta manera, la distancia de tiempos y frecuencias entre estos puntos son utilizados para codificarse por una función hash.

En el caso de Baluja y Covell [20] ellos realizan la transformación al dominio frecuencial mediante el uso de *wavelets*, método que tiene la particularidad de utilizar una función ventana de longitud variable para procesar la señal. Posteriormente, utilizan un método similar a Wang para extraer los *waveprints*, nombre representativo utilizados por los autores para referirse a los fingerprints.

2.3.1.2. Algoritmo Phillips

Haitsma, Kalker, y Oostveen [21] implementaron un algoritmo de reconocimiento acústico significativamente preciso. Inicialmente, cada encuadre de audio es convertido con las STFT. Luego, el espectro de frecuencias entre los $300 - 2000$ $[Hz]$ es dividido nuevamente en 33 *bins*, acorde a la escala psicoacústica de Bark [22], obteniéndose una banda asociada a diferentes tonos musicales. Posteriormente, cada banda es codificada por 0 o 1 dependiendo si la energía entre los rangos encontrados aumenta o disminuye, resultando un dato de 32 bits al que denominan *subfingerprint*. Al juntar 256 de éstos, se genera recién un fingerprint que abarca un tiempo de tres segundos.

En el proceso de reconocimiento todos las subpistas que contienen el hash son agregadas a una lista, así, la respuesta candidata será aquella que posee el menor rango de errores al determinar la diferencia a nivel de bit.

2.3.2. Obtención de fingerprints en base a machine learning

Los métodos que utilizan este enfoque agregan un subpreceso de entrenamiento a la estructura general de los algoritmos de reconocimiento, donde un modelo de aprendizaje automático es usado para mapear y asignar los sonidos musicales a los identificadores.

Kastner [23] convierte la señal de audio al dominio de frecuencia y luego calcula que tan plana es la superficie del espectro de cada encuadre, de modo de caracterizar el grado de tonos que posee el sonido al compararlo con ruido. Con posterioridad, se utiliza el método de reconocimiento de patrones llamado *Vector Quantization* para agrupar valores similares.

Por otra parte, Batlle, Masip y Gauss [24] usan los *Coeficientes Cepstrales* en las frecuencias de Mel como método para extraer las singularidades del audio, y así crear resúmenes de éste utilizando el modelo estadístico *Oculto de Markov* [25]. Además, con el fin de minimizar el efecto del ruido en el audio de búsqueda, el método realiza un modelamiento adicional del ruido basado en un filtro lineal, para luego

aplicar su inverso en la señal antes de generar los fingerprints. Esta metodología es masívamente empleada para determinar si las grabaciones han sido editadas o se han mezclado grabaciones diferentes.

Ke, Hoiem y Sukhankar[26] desarrollaron un método al que denominaron *Adaboost* con el cual son capaces de encontrar iterativamente la mejor forma para crear imágenes en blanco y negro que luego pueden ser aplicadas a las representaciones espectrales del audio. Una vez que las imágenes son aplicadas al archivo de audio se continúa con el proceso de búsqueda de coincidencias, al rastrear *subfingerprints* con bajos rangos de errores al diferenciar entre bits.

2.4. Sistemas de reconocimiento acústico

En la última década se ha visto el surgimiento de dispositivos móviles con micrófonos integrados que ha propiciado la masificación de herramientas para la identificación de temas musicales, a partir de grabaciones de audio. Si bien una de las aplicaciones más conocidas es Shazam, también existen otras alternativas y proyectos open-source, destinados al reconocimiento de versiones originales. Vale decir, audios de un mismo tema musical, pero que divergen en el formato de compresión o que presentan ruido de fondo. Sin embargo, no son algoritmos recomendados para reconocer patrones en versiones musicales en vivo o reconocimiento de tarareo, tal como lo ha incorporado SoundHound en sus últimas versiones.

A continuación se describen algunas opciones open-source disponibles para el reconocimiento acústico.

2.4.1. Echoprint

Este sistema de generación de fingerprints e identificación de canciones fue liberado en el 2011 por parte de la empresa The Echo Nest, cuyo rubro está orientado al análisis de contenidos musicales. Aunque Echoprint fue uno de los proyectos más reconocidos y con gran cantidad de usuarios, especialmente por la variedad de formatos de archivos de audio permitidos para el procesamiento, en los últimos años esta herramienta ha quedado discontinuada debido a ciertas limitaciones.

Las restricciones o malas prestaciones más destacables incluyen, en primer lugar, el tiempo de grabación requerido para obtener buenos resultados en la identificación, pues éste debe ser superior a los veinte segundos. Más aún, el porcentaje de acierto disminuye considerablemente si el audio grabado presenta ruidos de fondo.

2.4.2. Dejavu project

Este algoritmo de reconocimiento acústico está implementado en Python, e incluye la particularidad de poder grabar sonidos al conectar un micrófono. Según las pruebas realizadas por el autor, este método alcanza una precisión por sobre el 90 % con extractos de canciones que bordean los cinco segundos.

Sin embargo, en términos de almacenamiento, este algoritmo está por bajo el rendimiento de Echoprint ya que según la configuración que se escoja, puede generar una considerable cantidad de fingerprints. Si bien esto parece ser la opción idea si lo que se busca es rigurosidad en el reconocimiento de música, el tener una mayor cantidad de características o patrones de la canción aumenta el tiempo requerido para la etapa de coincidencia de fingerprints.

El producto fue desarrollado como pasatiempo por un aficionado, por lo que al no contar con una biblioteca musical en un servidor externo, presenta mayor flexibilidad para gestionar el audio que se desea analizar.

2.4.3. Chromaprint

Este producto es el componente central del proyecto AcousticID, servicio que en los últimos años se ha mantenido activo, gracias al catálogo de canciones que incluye más de 8.4 millones de grabaciones indexadas. Éstas, a su vez, son parte de la enciclopedia musical MusicBrainz cuya base de datos recopila los metadatos de estos archivos para compartir al público, información como títulos, intérpretes, año de publicación del álbum, etc.

Esta tecnología comparte su servicio mediante una API en C++, y esta biblioteca presenta la particularidad que solo calcula los fingerprints a partir de archivos de audio que no han sido comprimidos o procesados. Esto quiere decir que los formatos más conocidos para almacenar audio, como MP3, MP4 o FLAC, no son soportados como entrada, y por tanto, se debe contar con un método diferente para decodificarlos antes de agregarlos al servidor. Adicionalmente, los casos de uso de la aplicación son la identificación de archivos completos y para monitorear flujos prolongados de audio. En otras palabras, no es factible utilizarla si se quiere identificar breves fragmentos de música desde una grabación.

Otra característica a destacar es que, para que un desarrollador pueda utilizarlo es necesario contar con una cuenta en la plataforma web del sistema, para incluir archivos de audio que no sean parte del amplio repertorio de este servicio, pues de esta forma, se aseguran que el tema musical que se adiciona contiene los metadatos que MusicBrainz utiliza.

3 | Diseño de la solución

3.1. Metodología

Para desarrollar la plataforma y cumplir con los objetivos planteados en 1.3.1, se separó el proyecto utilizando un plan de trabajo que puede resumirse en las siguientes etapas:

- Analizar algoritmos de reconocimiento acústico para implementar uno de ellos en la plataforma de fiscalización radial, y establecer su configuración según los requerimientos del sistema.
- Diseñar y desarrollar la plataforma, testeando su funcionamiento con una única radio online, y comprobar que ésta detecta las canciones del repertorio de prueba.
- Ampliar el alcance de la plataforma para que sea capaz de reconocer canciones de manera simultánea, fiscalizando paralelamente en diversas radioemisoras.

3.2. Selección de herramientas de desarrollo

Bsándose en la investigación realizada en el apartado 2.4 se establecieron 4 aspectos para comparar los sistemas analizados, y determinar así, cual se ajusta mejor a los requerimientos del proyecto. Por ejemplo, una de las exigencias más relevantes de la plataforma según los objetivos secundarios planteados en 1.3.2, es contar con un base de datos propia que contenga todo el repertorio nacional, incluyendo además, las obras realizadas por artistas independientes con tal de mantener toda esta información de manera local, ya sea para evitar desfases de tiempo, o cualquier otro tipo de inconveniente a la hora de ingresar nuevas canciones a la plataforma.

En la tabla 3.1 se señalan las características fundamentales de cada sistema evaluado. De esta colección de herramientas, se ha optado por implementar Dejavu Project, puesto que es la única que establece una conexión local a la hora de analizar y trabajar con los fingerprints, al permitir crear una base de datos propia. De esta forma, el tiempo de búsqueda de coincidencias entre canciones se reduce en comparación a los

otros sistemas, si se considera que estos últimos requieren realizar solicitudes a fuentes externas, que no necesariamente estarán actualizadas con los últimos estrenos musicales del país.

Tabla 3.1: Cuadro comparativo entre softwares open-source de reconocimiento acústico.

Características	Echoprint	Chromaprint	Dejavu Project
Año de lanzamiento	2011	2012	2013
Lenguaje	C/C++	C	Python
Caso de uso	Extractos	Audio completo	Extractos
Precisión con 5 [s]	75-89 %	85-100 %	92-100 %
Conexión a base de datos	servidor Echonest	servidor AcousticID	local

3.3. Descripción de Dejavu Project

Como se ha señalado en el transcurso de esta investigación, el proyecto Dejavu es un sistema open source desarrollado en Python que basa su implementación en fingerprints. En esta sección se presentará una descripción técnica de la metodología con la finalidad de cubrir la descripción del funcionamiento de las dos etapas que conforman el esquema básico del algoritmo. La primera parte del proceso implica la generación de fingerprints, tanto del audio que será almacenado en la base de datos como la del extracto que se busca reconocer. En segundo lugar se especifica la manera en que las coincidencias entre fingerprints y candidatas a solución son procesadas.

3.3.1. Extracción de fingerprints

En vista que se ha aclarado el funcionamiento general de Dejavu, es posible ahondar en las operaciones y funciones de las diferentes extensiones de Python que permiten la generación de fingerprints para el funcionamiento de la plataforma. De esta forma, en párrafos posteriores no solo se indicará parte del código fuente del proyecto open source escogido, sino también, se incluirán ejemplos básicos de prueba para mejorar la comprensión del algoritmo.

El primer paso a seguir es leer el archivo de sonido con *AudioSegment* de *Pydub* para obtener los canales y muestras del audio.

```
audio = AudioSegment.from_file(nombreArchivo)
canales = audio.channels
muestras = audio.frame_rate
```

En segundo lugar se debe transformar las señales de audio a un dominio de tiempo y frecuencia, vale decir, se utiliza la Transformada corta de Fourier para la generación del espectrograma, por lo que en primera

instancia, se definen las constantes que serán utilizadas para la función ventana.

```
NUM_MUESTRAS = 44100
TAM_VENTANA = 4096
SUPERPOSICION = 0.5
TAM_VECINDARIO = 20
```

Posteriormente se hace uso de la función *specgram* de *Pyplot* para la representación de la onda en diversas frecuencias en función del tiempo.

```
arr2D = plt.specgram(mono,
             NFFT=TAM_VENTANA,
             Fs=NUM_MUESTRAS,
             window=mlab.window_hanning,
             noverlap = int(TAM_VENTANA * SUPERPOSICION))[0]
```

Las filas de la matriz *arr2D* representan la frecuencia, las columnas los segundos, y el valor de la posición M_{ij} es la amplitud. En el caso de la figura 2.7.b, los valores del espectrograma están comprendidos en una matriz de $[2049 \times 358]$, siendo su amplitud máxima igual a 3889624, punto que se encuentra en la zona más roja del diagrama.

Para ejemplificar el código señalado, se ha optado por diseñar una matriz sencilla con el fin de representar con ella las transformaciones que sufren los datos al utilizar el algoritmo. Sea una mtriz de dimensiones $arr2D_{[20\times10]}$ cuyos valores se muestran en la figura 3.1.

$$arr2D = \begin{pmatrix} 33 & 26 & 13 & 43 & 45 \\ 34 & 29 & 14 & 46 & 48 \\ 33 & 26 & 27 & 43 & 45 \\ 21 & 17 & 22 & 19 & 15 \\ 20 & 16 & 21 & 18 & 14 \\ 0 & 7 & 12 & 9 & 0 \\ -53 & -52 & -51 & -59 & -50 \\ -51 & -44 & -50 & -40 & -49 \\ -53 & -52 & -54 & -58 & -50 \\ -54 & -53 & -55 & -59 & -51 \end{pmatrix}$$

Figura 3.1: Representación matricial para espectrograma.
Matriz simplificada de ejemplo de un espectrograma obtenido tras aplicar la Transformada corta de Fourier.

Cabe destacar que la función *specgram* retorna un arreglo linear, por lo que se hace necesario aplicar

una escala logarítmica con tal de representar de mejor forma las diferentes magnitudes que esta transformación genera.

El tercer lugar se debe detectar los peaks del especgtrograma, y para lograrlo, es necesario definir aquel punto que tendrá la particularidad de ser máximo dentro de su vecindario. Para generar este listado de puntos adyacentes se crea una estructura binaria sobre la que se realizarán las operaciones morfológicas aplicacbles a la matriz que representa el espectrograma.

Numpy proporciona la función *generate_binary_structure*, pero dado que ésta solo permite matrices máximas de tres dimensiones, se debe utilizar *iterate_structure* para alcanzar el tamaño deseado.

```
estructura = generate_binary_structure(2, 1)
vecindario = iterate_structure(estructura, TAM_VECINDARIO).astype(int)
```

Antes de aplicar esta estrucutra representante del vecindario al espectrograma, se utiliza un filtro *maximum_filter* de *Scipy* para redefinir cada punto de la matriz como el máximo en su vecindario. Vale decir, si se considera un vecindario V_i dentro de la matriz, todos los puntos que conforman V_i tendrán el mismo valor, y éste será igual a la magnitud del punto máximo. Para el caso de nuestra matriz de ejemplo, los valores de máximos locales se señalan en la figura 3.2.

```
max_local = maximum_filter(arr2D, footprint=neighborhood)
```

$$max_local = \begin{pmatrix} 34 & 29 & 14 & 46 & 48 \\ 34 & 29 & 14 & 46 & 48 \\ 33 & 26 & 27 & 43 & 45 \\ 21 & 17 & 22 & 19 & 15 \\ 21 & 17 & 22 & 19 & 15 \\ 0 & 7 & 12 & 9 & 0 \\ -53 & -53 & -51 & -59 & -50 \\ -51 & -64 & -50 & -70 & -49 \\ -54 & -53 & -55 & -59 & -51 \\ -54 & -53 & -55 & -59 & -51 \end{pmatrix}$$

Figura 3.2: Representación matricial de máximos locales.
Matriz simplificada de ejemplo de los máximos locales de cada vecindario que conforman el espectrograma.

Si se comparan las matrices para determinar todas las posiciones donde los valores son iguales entre ambas, se obtienen los peaks de audio. En la figura 3.3 se han marcado con un recuadro todos las posiciones de la matriz cuyos valores coinciden entre *arr2D* y *max_local*.

```
peaks_detectados = arr2D == max_local
```

$$\begin{pmatrix} 33 & 26 & 13 & 43 & 45 \\ \boxed{34} & 29 & 14 & \boxed{46} & \boxed{48} \\ 33 & 26 & 27 & 43 & \boxed{45} \\ \boxed{21} & \boxed{17} & 22 & \boxed{19} & 15 \\ 20 & 16 & 21 & 18 & 14 \\ 0 & 7 & 12 & 9 & 0 \\ \boxed{-53} & -52 & -51 & \boxed{-59} & -50 \\ -51 & -44 & -50 & -40 & -49 \\ -53 & -52 & -54 & -58 & -50 \\ \boxed{-54} & -53 & \boxed{-55} & \boxed{-59} & \boxed{-51} \end{pmatrix} \quad == \quad \begin{pmatrix} 34 & 34 & 46 & 46 & 48 \\ \boxed{34} & 34 & 46 & \boxed{46} & \boxed{48} \\ 34 & 34 & 34 & 46 & \boxed{45} \\ \boxed{21} & \boxed{17} & 21 & \boxed{19} & 45 \\ 21 & 21 & 21 & 19 & 19 \\ 21 & 21 & 21 & 19 & 19 \\ \boxed{-53} & -53 & -53 & \boxed{-59} & -59 \\ -53 & -53 & -53 & -59 & -51 \\ -54 & -53 & -55 & -59 & -51 \\ \boxed{-54} & -54 & \boxed{-55} & \boxed{-59} & \boxed{-51} \end{pmatrix}$$

$\boxed{}$: Valores iguales entre la matriz del espectrograma *arr2D* y el conjunto de vecindarios *max_local*.

Figura 3.3: Representación matricial de peaks.
Igualación entre matrices para determinar los puntos que son coincidentes, y así obtener los peaks del audio.

Una vez determinados estos puntos de la matriz se extrae su información referente a la frecuencia y tiempo pues la amplitud ya no será necesaria. Además, es importante destacar que debido al elevado número de peaks que se pueden encontrar en una canción, es factible filtrarlos al establecer una amplitud mínima para que sean considerados en el arreglo final.

```
AMPLITUD_UMBRAL = 10

amplitudes = arr2D[peaks].flatten()
j, i = numpy.where(peaks_detectados)
peaks = zip(x, y, amplitudes)
peaks_filtrados = [x for x in peaks if x[2] > AMPLITUD_UMBRAL]

indice_frecuencia = [x[1] for x in peaks_filtrados]
indice_tiempo = [x[0] for x in peaks_filtrados]

maximos_espectrograma = (indice_frecuencia, indice_tiempo)
```

La siguiente tarea en el algoritmo de reconocimiento acústico es utilizar los máximos del espectrograma para encontrar los patrones propios de cada canción o fingerprints, y por consiguiente, la creación de tuplas que contemplen dos frecuencias entre peaks, con su respectivo Δt que las separe.

Es importante señalar que dichas agrupaciones serán almacenadas luego de utilizar una función hash de la librería *Hashlib*, en otras palabras, se emplea un algoritmo matemático que transforma una entrada de

datos en una salida alfanumérica de longitud fija, la cual representa una versión compacta de la información. Con esta implementación resulta más fácil buscar coincidencias entre los fingerprints amacenados en la base de datos y los que provienen del audio desconocido, puesto que de este modo solo se compara un único campo.

Para tal efecto, se definen las variables *PARES_FINGERPRINT*, *MIN_TIEMPO* y *MAX_TIEMPO*. La primera indica la cantidad de pares o asociaciones que se definirán para cada máximo a la hora de conformar las tuplas, y las dos últimas establecen la distancia máxima medida en tiempo para asociar las frecuencias, ya que de lo contrario se podrían agrupar peaks muy lejanos. Se adiciona igualmente una tercera variable, *MAX_CHAR_HASH*, para acortar el string del hash con la finalidad de reducir espacio de almacenamiento.

Llevando esta definición al espectrograma de la figura 2.7.b, PARES_FINGERPRINT corresponde a las siete líneas de la imagen; mientras que el largo de cada una de ellas no supera el máximo definido por el delta *MAX_TIEMPO* y *MIN_TIEMPO*.

```
PARES_FINGERPRINT = 7
MIN_TIEMPO = 0
MAX_TIEMPO = 200
MAX_CHAR_HASH = 20

FREC_I = 0
TIEMPO_J = 1

for i in range(len(maximos_espectrograma)):
    for j in range(1, fan_value):

        if (i + j) < len(maximos_espectrograma):
            frec1 = maximos_espectrograma[i][FREC_I]
            frec2 = maximos_espectrograma[i + j][FREC_I]
            t1 = maximos_espectrograma[i][TIEMPO_J]
            t2 = maximos_espectrograma[i + j][TIEMPO_J]
            t_delta = t2 - t1

            if t_delta >= MIN_TIEMPO and t_delta <= MAX_TIEMPO:
                hash_fingerprint = hashlib.sha1(
                    "%s|%s|%s" % (str(frec1), str(frec2), str(t_delta)))
                % hash como string
                h = (hash_fingerprint.hash_fingerprint()[0:MAX_CHAR_HASH], t1)
```

Una vez generados los fingerprints que permiten caracterizar a cada una de las canciones, creando perfiles únicos para su identificación, se debe contar con un sistema que permita almacenarlos en una base de datos, puesto que ésta será utilizada a futuro para comparar sus registros con el extracto de canción que se desea reconocer.

La base de datos utilizada con Dejavu es MySQL, cuyo esquema contiene solo dos tablas, *fingerprints* y *songs*, con una clave foránea *song_id* autoincremental que las relaciona. De la tabla 3.2, donde se señalan los campos de la estructura de la base de datos, se desprenden dos facotores importantes. En primer lugar, se debe tener en consideración que se crean muchos fingerprints por canción, por tanto el campo *hash* es un binary(10) para reducir el espacio ocupado para su almacenamiento. Con esta disminución de aproximadamente un 75 %, los cuarenta caracteres [$char(40) = 320\ b$] entregados por la función *sha1*, pasan a ser veinte [$char(20) = 160\ b$] debido a *MAX_CHAR_HASH*, para finalizar con diez bytes [$binary(10) = 80\ b$], al transformar el código hexadecimal a binario.

En segundo lugar destaca el campo *offset*, valor que almacena la ventana de tiempo donde se originó el hash. Dicho de otro modo, el valor en cuestión corresponde al valor inicial del delta de tiempo que separa el par de frecuencias del fingerprint. Este dato toma relevancia pues se utiliza como complemento en la etapa de búsqueda de coincidencias al tratar de reconocer una canción.

Tabla 3.2: Nombre y tipos de datos que conforman las tablas de la base de datos de Dejavu.

Nombre de la tabla	Dato	Tipo
songs	song_id	mediumint(8) unsigned
	song_name	varchar(250)
fingerprints	fingerprinted	tinyint(4)
	hash	binary(10)
	song_id	mediumint(8) unsigned
	offset	int(10) unsigned

3.3.2. Coincidencia de fingerprints

Tan pronto como se obtienen los fingerprints del extracto de audio que se desea reconocer, no basta con comparar los hash, sino que es necesario analizar el offset previamente insertado. Sin embargo, hay que recordar que esta cifra indica un intervalo respecto al inicio de la canción original, valor que no cobra sentido si se utiliza para comparar una muestra aleatoria, pues estas cantidades solo coincidirían si el fragmento de música contuviese exactamente el inicio de la canción. Entonces, para poder utilizarlo como método complementario al hash, la información debe ser previamente tratada.

Para apoyar la explicación del algoritmo, se ha elaborado un caso ejemplo donde se han simplificado considerablemente las diversas variables que se han trabajado hasta el momento. El modelo consta de dos

canciones de once segundos de duración que presentan seis peaks de audio, utilizados para generar cuatro fingerprints almacenados en una base de datos. La figura 3.4 es una representación gráfica de la situación descrita. Las imágenes (a) y (b) indican la posición de los peaks con sus respectivos tiempos y frecuencia, en tanto el gráfico (c) es la simulación de un fragmento de ocho segundos de la primera canción, el cual ha comenzado a grabarse dos segundos después del tiempo en que el audio inicia originalmente.

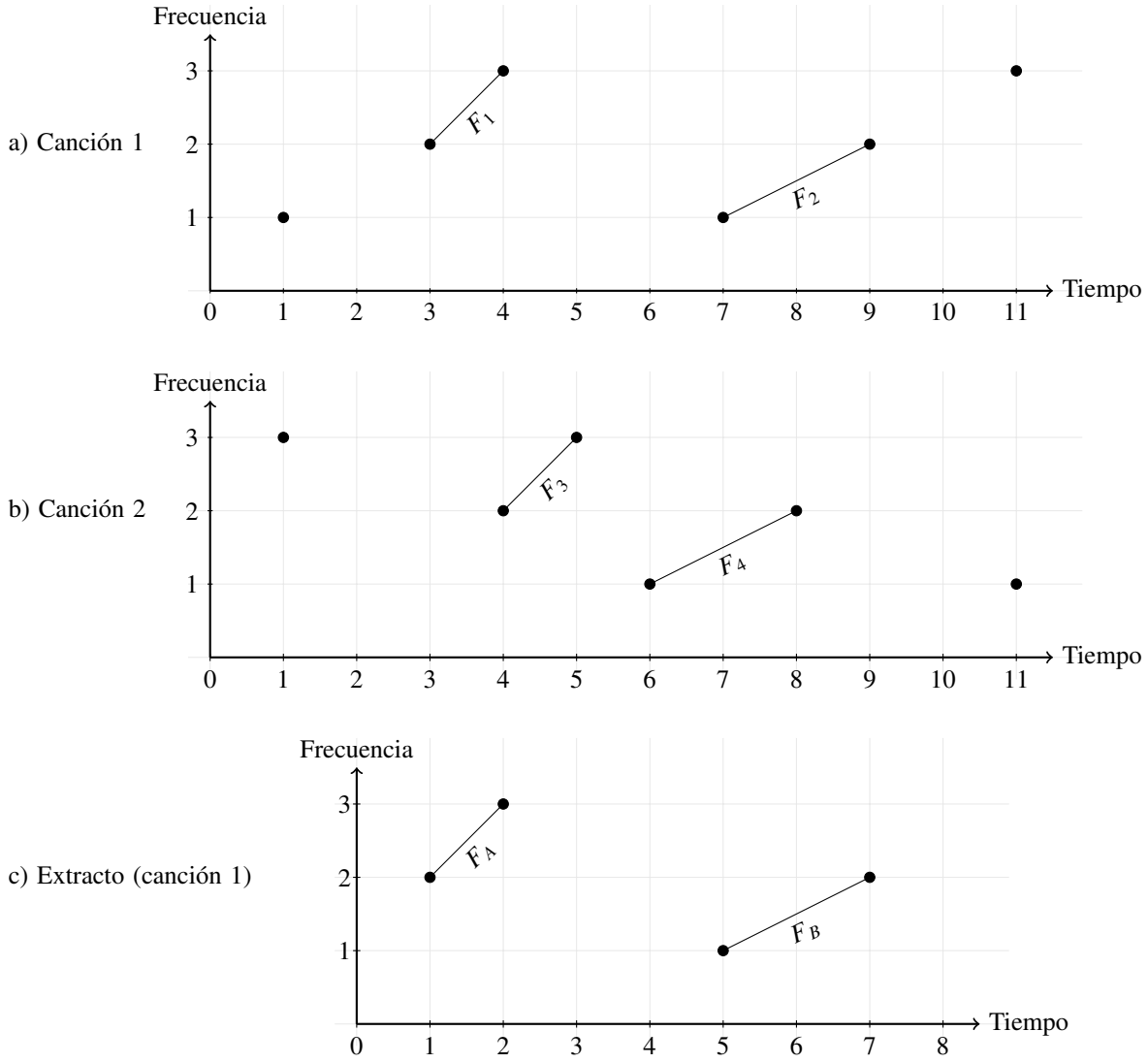

Figura 3.4: Diagrama de ejemplo de tres archivos de audio con sus rerpectivos fingerprints.
La imagen (a) y (b) emulan dos canciones de ocho segundos que contienen seis peaks de audio cada una, utilizados para la generación de sus fingerprints. En tanto la figura (c), representa una fragmento de música grabado dos segundos después que comienza la Canción 1, al que también se le han obtenido los máximos del audio.

Se debe recalcar que los hash generados del extracto de música, almacenados temporalmente en un arreglo para buscar coincidencias en la base de datos, pueden provocar varias concurrencias, principalmente debido a que estos valores son truncados para reducir espacio, según fue señalado en la sección anterior. Por este motivo se utiliza el cálculo de la diferencia entre offsets para mejorar la distinción entre canciones.

```
diferencia = offset_bdd_original - offset_extracto
```

Volviendo al ejemplo guía, se genera la tabla 3.3 donde se detallan los cuatro fingerprints almacenados, con sus respectivos id, además de tuplas de frecuencia y delta tiempo que los componen.

Se recalca que la mayoría de los valores de este ejemplo son ficticios, con el fin de facilitar la comprensión y visualización de las posibles coincidencias para reconocer las canciones. Tal es el caso del campo hash, que ha sido representado solo por cuatro caracteres, y no de forma binaria como ocurre en el algoritmo. Adicionalmente, se agregan en las dos últimas columnas de la tabla los fingerprints encontrados en el fragmento de música. Sus hash correspondientes son *abcd* y $efgh$, información que es cotejada para encontrar similitudes.

Tabla 3.3: Resumen de caso ejemplo para algoritmo de identificación de canciones.

Id canción	Nombre	Fingerprint			Hash	Offset
		Frecuencia 1	Frecuencia 2	Δt		
1	F_1	2	3	1	abcd	3
1	F_2	1	2	2	efgh	7
2	F_3	2	3	1	abcd	4
2	F_4	1	2	2	efgh	6
Extracto	F_A	2	3	1	abcd	1
Extracto	F_B	1	2	2	efgh	5

Para el hash *abcd* producido por F_A, existen dos concurrencias en la base de datos, específicamente F_1 y F_3, ambos, de canciones diferentes. De este modo, para obtener finalmente la identificación de la canción, se calculan las diferencias entre los ofsset de cada registro. Este análisis se condensa en la tabla 3.4, donde se incluye también, el estudio para la segunda huella del extracto, F_B. De la misma lista, se desprende que los fingerprints de la canción con $id = 1$ tienen el mismo valor de la variable *Diferencia*, y por consiguiente, Dejavu devuelve el id de la *Canción 1* como respuesta.

Tabla 3.4: Cálulo de diferencia entre offsets de caso ejemplo.

Nombre	Fingerprint coincidentes	Diferencia	Id canción
F_A	F_1	3 - 1 = 2	1
	F_3	4 - 1 = 3	2
F_B	F_2	7 - 5 = 2	1
	F_4	6 - 5 = 1	2

Para el caso de una base de datos que contiene mayor número de fingerprints, la cantidad de registros que coincidirán con la búsqueda aumenta considerablemente, y en virtud de ello, es esperable que el arreglo que almacena todas las posibles respuestas, involucre huellas acústicas de canciones diferentes. Por tanto,

Dejavu cuantifica, para cada uno de estos id, si la diferencia entre offset coincide para todos los fingerprints almacenados y extraídos del fragmento que se pretende reconocer, de modo de generar pares que contemplan la información respectiva al id de una canción y la diferencia entre offsets.

```
aceirtos = []
aciertos.append([id_cancion, diferencia])
```

Un modo de imaginar este última definición es utilizando el histograma de la figura 3.5. Éste se ha diseñado con la información de la tabla 3.4 que recapitula los valores que se han generado en el ejemplo de la sección. Como se puede apreciar, la *Canción 1* tuvo el mismo valor de *Diferencia* para sus dos fingerprint, de modo que el par [Id canción: 1 - Diferencia: 2] se repitió dos veces. No así ocurre con los fingerprints de la *Canción 2* de donde se obtuvieron dos cifras distintas para la variable *Diferencia*, razón por la cual, cada uno de estos pares son adicionados al histograma por separado, a pesar de pertenecer a la misma canción.

Finalmente, se deduce el nombre de la canción que se busca identificar, al distinguir el par [id_cancion - diferencia] con mayor número de repeticiones dentro del histograma.

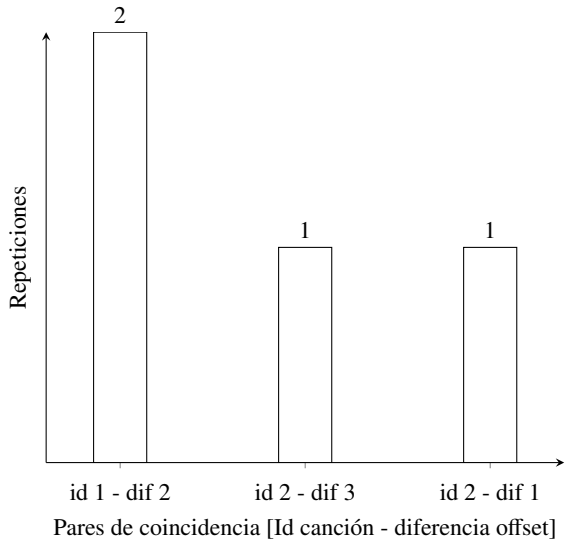

Figura 3.5: Histograma con el número de repeticiones para selección de solución en Dejavu.
El gráfico agrupa pares [id_canción - diferencia offset] según los valores obtenidos en la tabla 3.4. Mientras mayor sea el número de repeticiones, aumenta la certeza que hay de haber encontrado la canción correcta.

3.3.3. Estructura de archivos de Dejavu Project

Se deben mencionar, además, los archivos .py que componen a Dejavu project. *_init_.py* contiene las definiciones de la clase Dejavu cuyos métodos más relevantes son *get_fingerprinted_songs* puesto que informa si existe o no una determinada canción en la base de datos y así evitar información redundante. Y en segundo lugar se encuentra *recognize*, encargada de llamar a los módulos que se dedican a generar los

fingerprints del extracto de una canción, y así comenzar la búsqueda en la base de datos para el posible reconocimiento.

A su vez, *database.py* y *database_sql.py* contienen las funciones que se encargan de implementar y manejar la base de datos. El primer archivos solo contiene el nombre de los métodos, como mera referencia para la ejecución del mismo. Mientras que la segunda detallas las definiciones del archivo anterior, donde se incluye la configuración de la conexión con la base de datos, y toda consulta utilizada por las diversas secciones de Dejavu, como creación de tablas y función hashing.

El par de archivos *decoder.py* y *wavio.py* contienen las funciones que permiten el procesamiento de la música. El primero hace uso de Pydub, una librería de python desarrollada para la manipulación de archivos de audio, de tal forma que, a la hora de ingresar canciones al sistema, basta con proporcionarle al objeto Dejavu el directorio que contiene los archivos .mp3 o .wav para que comience el proceso de identificación y generación de fingerprints. No obstante, Pydub no soporta arhivos .wav de 24-bit, por lo que en dicha instancia, llama a *wavio.py*

Un sexto archivo es *fingerprint.py*, cuya finalidad es el análisis de los archivos de audio para reconocer los peaks de las canciones, y por consiguiente, de la generación de fingerprints, mediante su método *get_2D_peaks*. También, incluye el método *generate_hashes*, que como lo indica su nombre, provee de la estructura para la generación del índice hash. Se debe agregar que dicho archivo también contiene 5 variables que pueden ser modificadas a la hora de ejecutar el programa, pues sus valores permiten variar la cantidad de fingerprints, a costa de disminuir la calidad de la solución entregada por el sistema, y con ello, la seguridad de la respuesta. Estas variables son: *DEFAULT_OVERLAP_RATIO*, *DEFAULT_FAN_VALUE*, *DEFAULT_AMP_MIN*, *PEAK_NEIGHBORHOOD_SIZE*, y *PEAK_SORT*.

Cuatro de estos conceptos tienen su correlación con las variables que han sido definidas en los códigos de 3.3.1. La equivalencia de nombre se define en la tabla 3.5.

Tabla 3.5: Correlación en nombres de variables del algoritmo de reconocimiento acústico.

Nombre en Dejavu	Nombre en descripción
DEFAULT_OVERLAP_RATIO	SUPERPOSICION
DEFAULT_FAN_VALUE	PARES_FINGERPRINT
DEFAULT_AMP_MIN	AMPLITUD_UMBRAL
PEAK_NEIGHBORHOOD_SIZE	TAM_VECINDARIO
PEAK_SORT	N/A

Para finalizar, existe *recognize.py*, que como se mencionó en párrafos anteriores, es un módulo llamado desde *_init_.py*. Éste contiene dos clases cuya funcionalidad es muy similar. La clase *FileRecognizer* utiliza un archivo de audio para comenzar el reconocimiento acústico, mientras que la clase *MicrophoneRecognizer*, permite utilizar el micrófono del dispositivo para realizar el mismo procedimiento, por tanto, su principal diferencia es el tipo de entrada.

3.3.4. Análisis de funcionalidad

Una vez escogido el algoritmo de reconocimiento acústico, se procede a realizar pruebas a su funcionamiento, para determinar el comportamiento frente a diversos escenarios que requieren un rendimiento adecuado por parte del programa, de modo de identificar correctamente las canciones.

De esta forma, para evaluar el rendimiento de Dejavu se han escogido dos circunstancias que podrían provocar un rendimiento ineficaz por parte del algoritmo. Asimismo, tomando en consideración los atributos del programa mencionados en la sección anterior, se determinará su comportamiento, analizando variables como tiempos de creación de fingerprints y tamaño de la base de datos, además de tiempo de respuesta, aciertos y confianza de las salidas del algoritmo, para determinar la mejor configuración del mismo.

Cabe mencionar, que las características del equipo utilizado para realizar las pruebas mencionadas, se detallan en 4.3.

A: Tiempo de creación de fingerprints

Utilizando la configuración de Dejavu por defecto, se han creado seis bases de datos, diferenciando cada uno de ellas en la cantidad de canciones, comenzando desde 10 obras musicales hasta 250, con tal de estudiar el tiempo transcurrido para realizar los fingerprints correspondientes.

En la tabla A2 se comparan las bases de datos creadas, observándose su tamaño, además de tiempos de creación. Se ha elaborado el gráfico de la figura 3.6, tomando los valores de ésta, que permite estimar el tiempo que tomaría crear fingerpirnts para 80.000 canciones, al extrapolar con una tendencia lineal. Es importante destacar que el valor 80.000 no es arbitrario, sino que corresponde a una aproximación realizada por la SCD sobre la actividad musical del país [27].

Analizando la ecuación del gráfico se obtiene un tiempo de 274 días, por lo que se hace necesario modificar los atributos originales del programa para reducir el tiempo de creación de la base de datos oficial de la plataforma.

B: Atributos de configuración

Como ha sido señalado, Dejavu cuenta con cinco atributos que pueden ser modificado a la hora de crear la base de datos, por lo que cada uno de ellos fue analizado por separado. Las bases de datos contienen las mismas canciones, pero que cuya cantidad varía entre 3 y 5, con tal de identificar la forma en que cada elemento estudiado influye en el número de fingerprints creados.

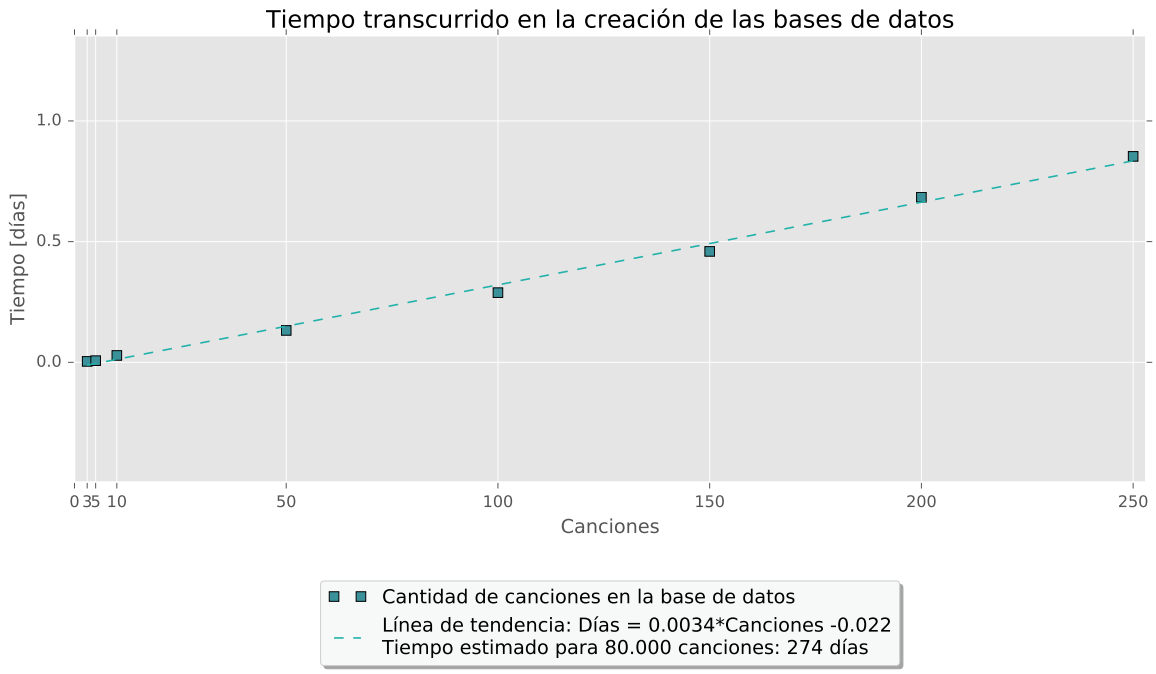

Figura 3.6: Tiempo de creación de base de datos en función del número de canciones con la configuración original de Dejavu.
La ecuación de la línea de tendencia, permite extrapolar el valor de los días necesarios para crear la base de datos de 80.000 canciones de la plataforma.

La tabla A3 del apéndice contiene los valores de configuración de cada base de datos diseñada, con sus respectivos tiempos de creación, tamaños y cantidad de fingerprints. Estos datos se utilizan para la generación de los gráficos de la figura 3.7 que permitan identificar la relación entre cada uno de estos parámetros, con su respectiva cantidad de fingerprints. Las variables DEFAULT_OVERLAP_RATIO y DEFAULT_FAN_VALUE son curvas crecientes, por lo que el aumento de huellas acústicas es proporcional al aumento en el valor del parámetro. En tanto, DEFAULT_AMP_MIN y PEAK_NEIGHBORHOOD_SIZE tienen comportamiento inverso. Hay que mencionar además, que la quinta variable estudiada llamada PEAK_SORT, es una variable booleana, que al modificarle el valor a *False*, la cantidad de fingerprints desciende considerablemente en comparación a las otras bases de datos. No obstante, no se realizaron mayores estudios de dicha variable debido al mal rendimiento que ocasiona esta disminución en el número de identificadores únicos de cada canción. Una vez creadas las bases de datos, se consideraron los siguientes factores para analizar el algoritmo de Dejavu, realizando pruebas de reconocimiento acústico:

- **Tiempo de respuesta de Dejavu** indica el lapso que transcurre desde que el algoritmo recibe una consulta, crea los fingerpirnts del extracto de música analizado, y se buscan coincidencias en la base de datos hasta que se devuelve la respuesta.

- **Confianza:** Certeza de la respuesta entregada por el algoritmo, pues indica la cantidad de fingerprints coincidentes en la búsqueda.

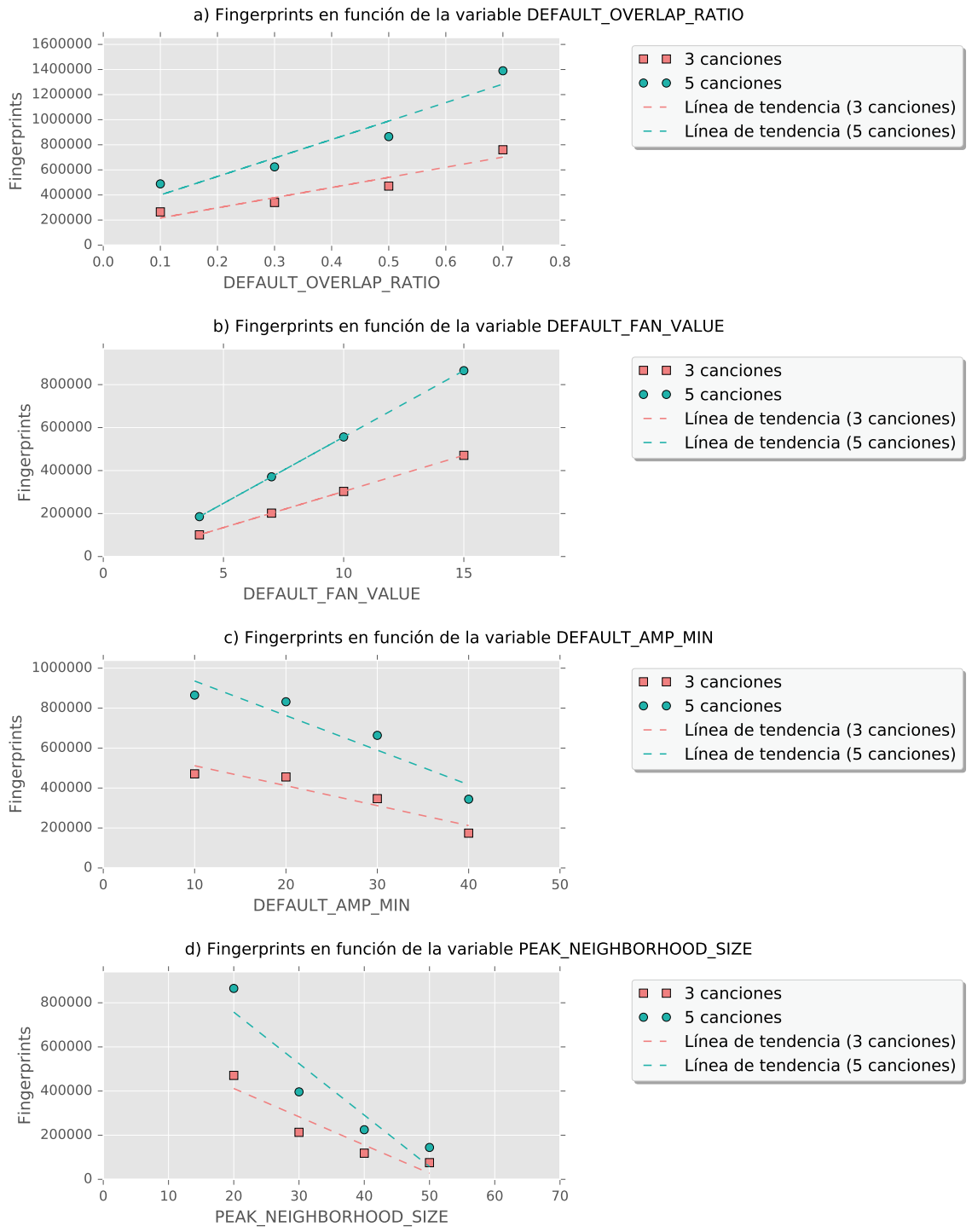

Figura 3.7: Fingerprints en función de diversos valores de configuración.
Los gráficos muestran la variación de la cantidad de fingerprints en función de los valores asignados a las variables de configuración: DEFAULT_OVERLAP_RATIO y DEFAULT_FAN_VALUE como rectas crecientes, PEAK_NEIGHBORHOOD_SIZE y DEFAULT_AMP_MIN como rectas decrecientes.

- **Acierto:** esta variable determina efectivamente si la salida del algoritmo corresponde al extracto de canción analizado.

- **Duración del extracto:** parámetro medido en segundos que determina la duración de cada archivo de música de prueba, abarcando un rango de 5, 10 y 15 segundos.
- **Sección del extracto:** corresponde al momento en que se ha creado un extracto de música de prueba. De esta forma, se comprueba si el algoritmo acierta con la respuesta, independiente si la solicitud de reconocimiento acústico se hizo al principio, al final, o en la mitad de la canción.

Para realizar las pruebas, se escogieron tres canciones presentes en todas las bases de datos diseñadas, y se crearon extractos de 5, 10 y 15 segundos de cada una de ellas. La extensa salida de los resultados fue resumida en la tabla A4 del apéndice.

La primera afirmación que se puede realizar es la relación proporcional entre la cantidad de segundos del extracto de música que Dejavu analiza, con la cantidad de aciertos, pues a medida que aumentan los segundos, también lo hace las respuestas correctas entregadas por Dejavu. Es más, para extractos de 15 segundos, la mayoría de las soluciones son consideradas correctas. A partir de esta observación, se establece que para el algoritmo de la plataforma, se utilizarán extractos de esta cantidad de segundos para analizar el streaming de las radios.

Además, a simple vista se observa que la base de datos E1, aquella que modifica PEAK_SORT como valor *False*, redujo en mayor proporción el numero de fingerprits creados, por lo que su tiempo estimado es el menor de todos. Empero, tuvo un improductivo rendimiento, donde solo logró 3 aciertos cuando se utilizaba un extracto de 10 segundos, y en consecuencia, el valor de este booleano no es considerado como posible factor para la configuración de Dejavu, puesto que la limitada cantidad de fingerprints reduce el desempeño del algoritmo.

Teniendo en cuenta que, la razón para modificar los valores de configuración es reducir el tiempo de la creación de la base de datos, se preseleccionan las bases B1, B2 y B3 debido a la menor cantidad de fingerprints que resultaron en su creación. Además presentan un buen rendimiento. Al revisar el número de aciertos por el total de pruebas realizadas, solo 1 de 9 tuvo una respuesta incorrecta, producto de un extracto de 5 segundos.

C: Prueba de extractos de música inexistente en la base de datos

Una segunda característica esencial esperable de un buen rendimiento además del número de aciertos, es el comportamiento del algoritmo en las ocasiones en que la canción testeada no es parte del repertorio.

Dado que la idea fundamental de esta plataforma es albergar solo la música chilena, se hizo este análisis con el propósito de determinar como sería mayoritariamente el comportamiento del algoritmo, debido

a que, en general, la programación emitida por las radioemisoras nacionales es extranjera, y por consiguiente, la mayoría de la parilla radial emitida en el país no será almacenada en la plataforma.

Por estas razones, se realizaron las mismas pruebas de reconocimiento acústico que en el caso anterior, con la salvedad que, a Dejavu se le asignó de entrada archivos de música que no existen en la base de datos. Los resultados de la tabla A5 señalan que las tres configuraciones preseleccionadas, B1, B2 y B3, no obtuvieron el resultado esperado. Es más, para todas las pruebas realizadas, Dejavu efectivamente devolvió un listado con el nombre de diversas canciones, sin embargo, estas respuestas eran erróneas, pues lo correcto hubiese sido que el algoritmo devolviera un valor nulo para cada una de ellas, dado que las canciones no existían en el repertorio.

Para ahondar en el estudio de este comportamiento, se analiza la fila de confianza máxima de las respuestas erróneas y confianza mínima de los aciertos de las tablas de resultados, pues así, es posible establecer un delta que permita definir un límite para aquellas respuestas erradas entregas por Dejavu. Vale decir que, independientemente que el programa devuelva el nombre de alguna canción, éste puede ser incorrecto, por tanto, si la confianza de esta respuesta no supera cierto umbral, no se considera segura.

De la tabla A4 donde sí hay respuestas correctas, el valor mínimo de confianza de los aciertos es de 68, 126 y 179 para B1, B2 y B3 respectivamente. Mientras que de la tabla A5 se desprenden que la confianza máxima de los errores es de 4 para cada base de datos preseleccionada. Debido a este bajo valor, es esperable que el nombre de la canción devuelto por Dejavu sea incorrecto. Como resultado, las respuestas con valores de confianza que se encuentren dentro de este delta, deben ser fácilmente identificadas a la hora de fiscalizar y realizar el listado de canciones emitidas por una radio, puesto que lo más seguro es que el nombre no sea correcto.

Dentro de este marco, al entrecruzar toda la información de tablas y gráficos presentados en este capítulo, revisando además el número de aciertos, en conjunto con los valores de tiempo estimado para creación de la plataforma, se ha optado por B2 como la configuración final de Dejavu, cuyos parámetros se definen en la tabla 3.6

Tabla 3.6: Valores de los parámetros de la nueva configuración de Dejavu.

Parámetro	Valor
DEFAULT_OVERLAP_RATIO	0,5
DEFAULT_FAN_VALUE	7
DEFAULT_AMP_MIN	10
PEAK_NEIGHBORHOOD_SIZE	20
PEAK_SORT	True

Una vez editados los valores de configuración, se han generado nuevas bases de datos para determinar una aproximación del tiempo estimado para la creación de los fingerprints oficiales de la plataforma. Los tiempos empleados para la elaboración de cada una de ellas se detalla en la tabla 3.7. En base a la

Tabla 3.7: Tiempo de creación de bases de datos, con los valores de la nueva configuración.

Número de canciones	Tiempo [seg]	Tiempo [min]	Tiempo [hr]	Espacio [MB]
10	396	6,6	0,1	62,20
20	802	13,4	0,2	146,4
30	1124	18,7	0,3	204,5
50	2206	36,8	0,6	370,1
100	5555	92,6	1,5	765,0
150	8666	144,4	2,4	966,0
200	12821	213,7	3,6	1321,0
250	17428	290,5	4,8	1653,0

representación gráfica de estos valores (véase figura 3.8) se conjetura un lapsus de 64 días para la extracción de fingerprints de las 80,000 canciones, con un tamaño aproximado de 510 [GB].

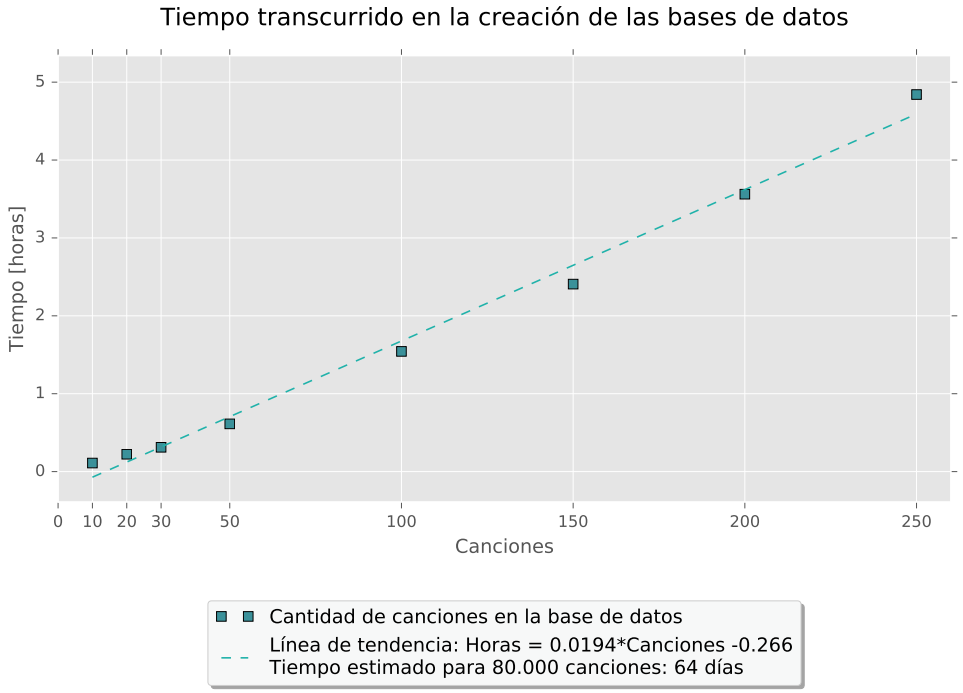

Figura 3.8: Tiempo de creación de base de datos en función del número de canciones con la nueva configuración de Dejavu.
Tiempo transcurrido en la creación de las bases de datos con la nueva configuración de Dejavu. La ecuación de la línea de tendencia, permite extrapolar el valor de los días necesarios para crear la base de datos de 80.000 canciones de la plataforma.

D: Tiempo de respuesta

Con los resultados obtenidos en las pruebas anteriores, se hace indispensable constatar nuevamente el comportamiento del sistema una vez que se han escogido los parámetros de configuración. Por ello, en esta

sección se compararán los tiempos de respuesta cuando la música está presente en la base de datos, como de búsquedas con canciones ausentes en el catálogo de prueba.

En primer lugar se verifica la relación entre la duración del extracto de música, que se le envía a Dejavu para que comience el reconocimiento, y los segundos que tarda el mismo en generar los respectivos fingerprints para buscar concurrencias. Tal como lo ilustra la figura 3.9 se obtiene una recta creciente cuya relación es casi lineal. Llama la atención además, que la variación en los tiempos de respuesta es menor para las canciones que no pertenecen al catálogo.

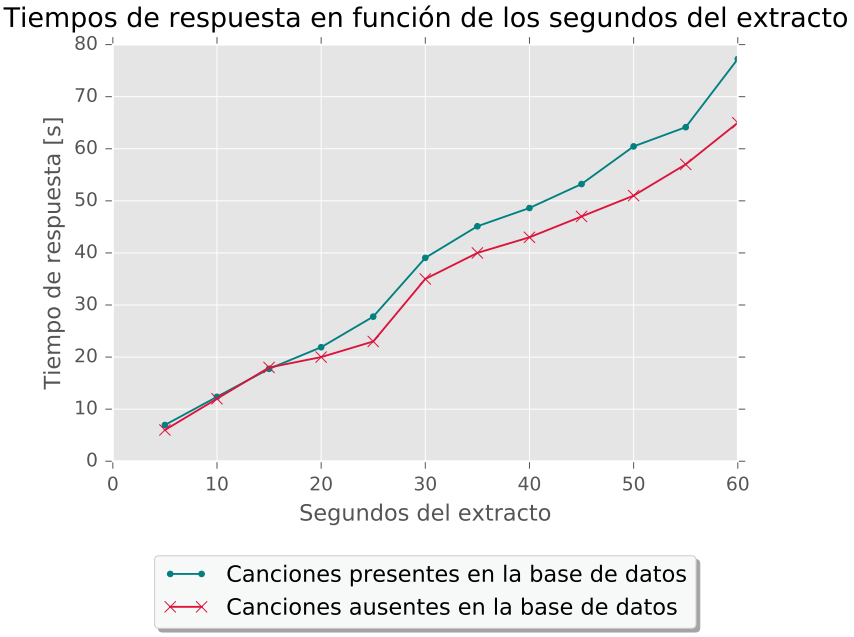

Figura 3.9: Tiempo de respuesta en función de la duración de los extractos de música.
El gráfico representa la variación del tiempo empleado para encontrar una solución al extracto que se desea identificar. La línea con cruces (x) corresponde a fragmentos de canciones que no pertenecen a la base de datos. Mientras que la línea con puntos(.) hace referencia a búsquedas de canciones existentes en el catálogo de prueba.

Para concluir, se ha estudiado si existe alguna conexión entre la cantidad de fingerprints y el tiempo de respuesta para encontrar la solución. Para lograrlo, se han evaluado mutliples extractos de música de 10 segundos de duración, cambiando la base de datos de tal forma de variar la cantidad de canciones que componían a cada una de ellas. Contrastando las figuras 3.10.a y 3.10.b se determina que el número de registros en la base de datos no influye en los tiempos de respuesta.

E: Prueba de covers en Dejavu

A continuación, es necesario examinar un último escenario, que consiste en evaluar el comportamiento de Dejavu, sobre aquellas canciones de artistas que han sido interpretadas por otros músicos, creando nuevas versiones de una obra musical.

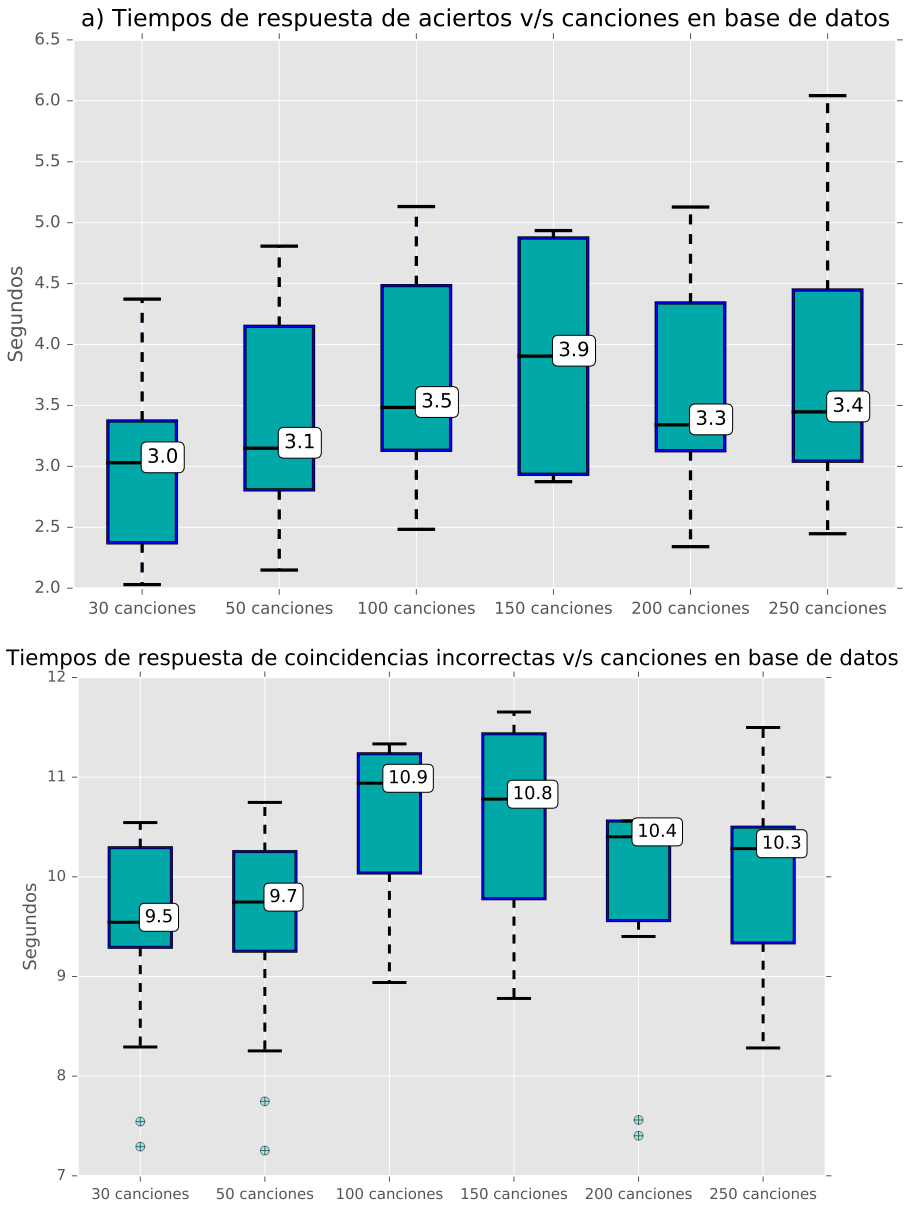

Figura 3.10: Comparación de tiempos de respuesta en función de la cantidad de canciones de la base de datos. La ilustración (a) corresponde al tiempo transcurrido hasta obtener la identificación de los extractos de 10 segundos cuyos fingerprints pertenecen a la base de datos. En contraposición, el gráfico (b) indica los tiempos de respuesta para música ausente en la recopilación de muestra.

Es esperable que los fingerprints entre covers tengan cierta similitud debido a la semejanza de sus ritmos musicales, por lo que se ha escogido una serie de canciones para observar las respuestas del algoritmo. Inicialmente se escogieron canciones disponibles en la base de datos, y luego, se eliminaron algunas para comprobar si la respuesta de Dejavu sería o no el cover correspondiente.

La tabla A7 presenta el resumen de estos resultados, tras reconocer fragmentos de música en bases de datos que varían en tamaño, y que efectivamente contienen las canciones que fueron analizadas. Conforme

a ésta, se comprueba que por sobre los 10 segundos, todos los archivos de audio fueron reconocidos correctamente, inclusive, aquellos covers que tenían su contraparte original en el sistema.

Para la segunda evaluación se eliminaron las canciones originales de los registros para corroborar la precisión del programa. De la misma manera que ocurrió en 3.3.4, Dejavu devolvió títulos de canciones incorrectos, cuando se esperaban respuestas vacías (véase tabla A8). No obstante, el valor de confianza máximo no es mayor a 10, coincidiendo con el umbral establecido con anterioridad para aceptar la respuesta de la aplicación como correcta.

3.4. Descripción del programa

3.4.1. Fiscalización de una radioemisora online

De acuerdo a la metodología implementada, tras escoger y configurar el algoritmo de reconocimiento acústico para el programa de fiscalización, se procede a desarrollar la plataforma, de tal forma que abarque solo la supervisión de una única radio online. Por tanto, para abordar este requerimiento se identificaron las tareas fundamentales del sistema para lograr su funcionamiento, las que se aprecian en el siguiente seudocódigo.

Algoritmo 1 Fiscalización de una radioemisora online

Entrada: *urlRadio, nombreRadio, segundosExtracto, fechaLimite, delay.*
Salida: archivo de texto plano.

```
1: craerArchivoSalida(nombreRadio)
2: Mientras (tiempo actual < fechaLimite) Hacer
3:     Acceder a urlRadio del streaming
4:     Transformar señal continua de música en discreta creando extractos de segundosExtracto
5:     Reconocer extracto de música con Dejavu
6:     Exportar respuesta a documento de fiscalización
7: Fin Mientras
```

En líneas generales, el programa requiere estar constantemente accediendo a la url del streaming cada cierto intervalo de tiempo, con tal de generar múltiples archivo de música, los que serán transmitidos a la función de Dejavu que se encarga del reconocimiento acústico. De este modo, es posible generar una lista de las canciones que están siendo emitidas en una determinada radio online.

El detalle de las variables de entrada del programa se presenta a continuación:

- **urlRadio**: cadena que almacena el link del streaming de la radio que se controla.
- **nombreRadio**: variable que identifica por un nombre corto la radio que se analiza, y cuyo valor será utilizado para nombrar el archivo de salida.

- **segundosExtracto**: variable entera que indica la duración que tendrán los extractos de música creados desde el streaming.
- **fechaLimite**: variable tipo fecha que puede ser definida de dos formas.
  - fechaLimite en minutos: esta variable contabiliza el tiempo desde que se corre el programa hasta que transcurran los minutos definidos por el usuario.
  - fechaLimite como día: el programa se ejecuta hasta el día, hora, minutos y segundos, que se le definieron.
- **delay**: variable entera definida en segundos, que indica cada cuanto tiempo se están creando extractos de música desde la radio, para transformar la señal continua en discreta, y hacer uso de Dejavu para su reconocimiento.

Cabe mencionar que la salida del programa corresponde a un archivo de texto, el cual es creado a partir del método *craerArchivoSalida(nombreRadio)*. Éste contiene el listado del nombre de las canciones, con sus respectivos tiempos cronológicos de emisión.

Para el ciclo iterativo (líneas 2 a 7) se desarrolló una función nombrada *fiscalizar*, la que se llama a sí misma cada cierto tiempo definido por la variable *delay*, hasta que se ha alcanzado el tiempo de fiscalización indicado por *fechaLimite*. La función *fiscalizar*, a su vez, llama a la función *generarExtracto*, pero en un nuevo hilo de ejecución. De esta forma, independiente que el algoritmo tarde mucho en reconocer alguna canción, seguirán creándose extractos según los segundos definidos, sin necesidad de esperar la respuesta de Dejavu para seguir con la ejecución del programa.

generarExtracto por su parte, corresponde a la función con más líneas de código, debido a que agrupa las tareas fundamentales del programa. El próximo seudocódigo especifica su labor.

Algoritmo 2 generarExtracto

Entrada: *urlRadio, nombreRadio, segundosExtracto, fechaLimite, delay.*

1: Determinar *nombreArchivo* del extracto de música de extensión .mp3 que se creará desde el streaming
2: Conexión streaming *urlRadio*
3: **Mientras** (segundos transcurridos < *segundosExtracto**2) **Hacer**
4: crear extracto *nombreArchivo*.mp3
5: **Fin Mientras**
6: *cortarCancion(nombreArchivo, segundosExtracto)*
7: cancion = dejavu.recognize(*nombreArchivo*)
8: *contenidoFila* = generar cadena texto para imprimir en archivo de salida, incluyendo nombre de la canción reconocida, fecha en que sonaba, y tiempo de respuesta de dejavu
9: *actualizarArchivoSalida(contenidoFila)*

Con respecto a la tercera línea de código del Algoritmo 3.4.2 es importante destacar que el valor de la variable *segundosExtracto* es duplicado, debido a que la duración real de la extracción de audio desde el streaming, varía en función de la memoria, tiempo de respuesta del servidor del streaming, etc, por lo que

en caso de no aumentar su extensión, estos archivos serán más cortos que el tiempo requerido. Por tanto, *cortarCancion* de la línea de código 6 es una función que se encarga de modificar la duración del archivo de música descargado según los parámetros definidos por el usuario.

Como se especifica en la cuarta línea de código, se crean archivos de audio de extensión .mp3 que contienen los extractos de música obtenidos desde la url del streaming de la radio, debido a que es el tipo de archivo que requiere como parámetro la función de reconocimiento acústico de Dejavu, y por ello, se hace indispensable transformar la señal continua de música en pequeños extractos. A su vez, el nombre de estos archivos contienen información sobre la fecha en la que fueron creados, por lo que será posible realizar una planilla que contenga la parilla musical de la radio fiscalizada, una vez el extracto de música sea reconocido por el algoritmo. Así, en la línea 7 del código, se define la variable *cancion* que almacena la respuesta de Dejavu en formato de diccionario.

Como resultado, se crea una cadena con todos estos datos y mediante el método *actualizarArchivoSalida* se imprime en una archivo de texto plano para generar la planilla de la radio, y de esta forma, se registra la música que estaba sonando en un determinado momento. Se debe mencionar además, que la respuesta de Dejavu puede conducir a tres situaciones. En primera instancia, la música será reconocida y dicha información será agregada a la planilla, sin embargo, en caso en que el diccionario *cancion* esté vació por no haber coincidencia de fingerprints, o el extracto de música no sobrepasa un umbral de frecuencia y genera un audio vacío, ambas respuestas son claramente señaladas en el archivo de salida con los respectivos tiempos en que las situaciones recién mencionadas ocurrieron.

3.4.2. Fiscalización paralela

La última etapa del proyecto consiste en probar el algoritmo anterior, modificando los aspectos necesarios para lograr fiscalizar múltiples radioemisoras online a la vez. Cabe destacar que, debido a la forma y estructura en que fue desarrollado el algoritmo anterior, fue sencillo modificar el método *fiscalizar* para generar mayor cantidad de hilos de ejecución y cumplir con el análisis paralelo de las radios.

A continuación se presenta el nuevo algoritmo, indicando las funciones agregadas al desarrollo anterior.

Algoritmo 3 Fiscalización paralela

Entrada: *archivoEntrada, segundosExtracto, fechaLimite, delay.*
Salida: archivos de texto plano con parilla musical.

1: listaNombreRadios, listaUrlRadios = *obtenerRadios(archivoEntrada)*
2: *crearArchivoSalida(listaNombreRadios)*
3: *fiscalizar(listaUrlRadios, listaNombreRadios, segundosExtracto, fechaLimite, delay)*

La primera diferencia es el método *obtenerRadios* el cual se encarga de leer un archivo de entrada de texto plano que contiene el nombre corto de una determinada radioemisora, con su respectiva dirección de

streaming. De esta forma, se generan las listas *listaNombreRadios* y *listaUrlRadios*.

Como se mencionó con anterioridad, la función fiscalizar, encargada de generar los extractos y reconocer la música, también fue ligeramente modificada para lograr revisar simultáneamente en varias estaciones de radio online, tal cual lo señala el algoritmo 4.

Dentro de este marco es importante señalar que el nombre de los archivos de audio generados son almacenados en una cola, de esta forma, es posible utilizar diversos threads para monitorear las radioemisoras paralelamente, los cuales obtendrán un archivo para comenzar con el reconocimiento, y una vez la cola se vacíe completamente, la aplicación dejará de ejecutarse.

Algoritmo 4 fiscalizar

Entrada: *listaUrlRadios, listaNombreRadios, segundosExtracto, fechaLimite, delay.*

1: **Si** (tiempo actual < *fechaLimite*) **Entonces**
2: *inicio* = definir tiempo de inicio
3: **Para** (urlRadio *en* listaUrlRadios) **Hacer**
4: *generarExtracto(urlRadio, nombreRadio, segundosExtracto, fechaLimite, delay)*
5: **Fin Para**
6: *final* = definir tiempo de finalización
7: *delayReal* = *delay* - (*final* - *inicio*)
8: *fiscalizar(listaUrlRadios, listaNombreRadios, segundosExtracto, fechaLimite, delayReal)*
9: **Si no**
10: Detener ejecución
11: **Fin Si**

De las líneas 3 y 4 del algoritmo 4, se desprende que se generan múltiples llamadas al método *generarExtracto* el cual, al estar dentro de un loop, se encarga de revisar cada radio disponible en el listado *listaUrlRadios*, de tal forma, de crear archivos de música para su posterior identificación, tal como se señala en el método 3.4.2 *generarExtracto* ya definido.

A su vez, el método *fiscalizar* se llama a si mismo una vez transcurrido el tiempo de espera definido por *delayReal*. Se ha incluido el cálculo de esta variable debido a que si la generación de múltiples hilos en la ejecución del loop provoca cierto retraso, el tiempo entre el extracto n-ésimo y extracto n+1 de una misma radio, mantendrá el *delay* original definido, sin considerar el tiempo utilizado para la generación de extractos de las otras radios.

Para finalizar, se debe recordar que la función *actualizarArchivoSalida* del algoritmo es la encargada de imprimir en un archivo de texto el listado de las canciones reconocidas con dejavu, con sus respectivos tiempos, de tal forma de generar automáticamente múltiples archivos con la parilla musical del listado de radios online.

4 | Experimentos

Indiscutiblemente, el proceso de testing dentro de la ingeniería de softaware juega uno de los papeles principales para brindar información relativa a la calidad de lo que se está desarrollando, pues resulta fundamental verificar y validar que el software satisface las necesidades del ususario y respeta su especificación. Sin embargo, existe una amplia rama que estudia y desarrolla metodologías para determinar la calidad del producto, y seguir este tipo de pautas muy elaboradas se transformaría en algo que se escapa del contexto del presente trabajo. Por ello, y dado el ámbito de esta publicación, se ha optado por implementar un plan básico de prueba, ya que de esta manera es posible simplificar el tipo de herramientas requeridas para llevar a cabo el análisis.

En vista de lo anterior, las actividades que se realizarán para testear la plataforma se enfocan principalmente en pruebas dinámicas, pues se requiere de la ejecución de la aplicación para poder efectuarlas. Adicionalmente, y teniendo en consideración que los experimentos se realizarán en la etapa final del desarrollo, se escoge la categoría de pruebas no funcionales.

4.1. Casos de prueba

Puesto que el contraste de la metodología no funcional generalmente considera el comportamiento externo del sistema, el modelo de caja negra es ideal para estudiar la plataforma cuando es sometida a una carga que actúa de forma concurrente, pues así no es necesario considerar los detalles de cómo fue desarrollada e implementada para estudiar las características de como trabaja el software.

Teniendo en consideración estas decisiones, en el presente capítulo se definirán los distintos ambientes de prueba, clasificándolos según algunas prácticas que existen en análisis de rendimiento. Cabe señalar que para todos los test descritos se utilizó el mismo equipo (véase la descripción de Equipo 1 en la tabla 4.1).

4.1.1. Pruebas de estabilidad

Tal como su nombre lo indica, este tipo de estudios se utiliza para determinar si la aplicación puede soportar una carga esperada continua con tal de comprobar que no existe degradación de la plataforma por un uso prolongado.

Como fue señalado en su momento, el 20 % de música chilena que la ley 20810 estipula, corresponde a menos de cinco horas diarias, por lo que en caso de utilizar una radio online, no sería posible asegurar cuántas canciones en la base de datos serán transmitidas durante el tiempo que se estará fiscalizando.

Teniendo esto en mente, y acotando la prueba para el presente desarrollo, se realizarán las observaciones sobre una radio personalizada diseñada especpificamente para esta situación y se ejecutará la aplicación durante veinticuatro horas. De esta forma, se podrá simular la pauta de una radio nacional, incluyendo a la transmisión de las canciones progrmas radiales con voces de locutores, con la adhisión de anuncios comerciales.

La recopilación de archivos utilizados para este experimento incluye siete programas radiales que se componen solo de voces de los panelistas, cuyo tiempo de duración es de 11:38:08. Con respecto a las canciones nacionales, el listado comprende 71 archivos de música que duran 5 horas, 6 minutos y 9 segundos. El resto de la transmisión incluye 110 canciones que no están disponibles en la base de datos.

Además de evaluar el comportamiento de la plataforma durante un día de funcionamiento, esta prueba permitirá estimar un rango o umbral entre confianza mínima de aciertos y confianza máxima de las respuestas incorrectas devueltas por el reconocimiento de Dejavu.

4.1.2. Pruebas de carga

Este tipo de análisis se realiza de tal forma de ir aumentando progresivamente la carga en el sistema para asegurar que la aplicación alcanza los objetivos establecidos en los requerimientos, bajo una cantidad de peticiones esperada. Por lo que según la situación descrita, para implementar este segundo ensayo será necesario constatar como se comporta la plataforma a medida que se incrementa la cantidad de redioemisoras que se fiscalizan.

Para lograr lo anterior, se ejecutará la aplicación de fiscalización durante aproximadamente veinte minutos, comenzando con el estudio de una sola radio online. Con posterioridad, se aumentará la cantidad hasta un límite de cuatro radioemisoras, con tal de analizar como evoluciona el tiempo de respuesta en función del número de transmisiones estudiadas.

Con el fin de llevar a cabo la idea previamente expuesta, se requiere contar anticipadamente con la programación musical, con tal de comprobar la veracidad de las canciones reconocidas por la plataforma. Por

tal efecto, nuevamente se transmitirá una listado de canciones nacionales en diversas radios online generadas específicamente para este análisis. Adicionalmente, al ser capaz de manipular el listado de archivos, será posible analizar el comportamiento del software bajo las circunstancias descritas en 3.3.4.D.

El catálogo es conformado por diez archivos, los que contemplan seis canciones originales y cuatro covers de éstas, música que ha sido previamente ingresada a la base de datos.

Cada uno de los equipos descritos en la sección 4.3 transmitirá la misma programación durante veinte minutos, transcurso de tiempo donde solo alcanzarán a sonar cinco canciones del total que están incluidas en la biblioteca musical. Adicionalmente, el equipo 1 también ejecutará el programa de fiscalización paralelamente a la emisión radial.

El propósito de esta prueba no es solo comprender el coportamiento del algoritmo a medida que aumenta la cantidad de radios a pesquisar, sino que además permitirá evaluar la factibilidad de fiscalizar más de una radio online a la vez, pues podrá obtenerse la razón a la que aumenta el uso de la memoria en función de los threads empleados.

Es importante aclarar que para cada uno de los test realizados en esta sección, el número de threads va en aumento. Al fiscalizar solo una radio, se tiene el hilo principal de ejecución *(función main)*,también el thread que monitorea la creación de extractos, un thread que se crea cada cietos segundos para descargar el fragmento, y por último un thread que se encarga de identificar la canción. Vale decir, en la instancia se puede llegar a tener un máximo de cuatro threads.

Para el caso de mas radioemisoras, se incrementa la cantidad de threads que identifican la música, a un valor igual a la cantidad de estaciones que se monitorean, y con ello, los threas temporales que descargan archivos de música desde las radios online. En otras palabras, al fiscalizar 2 radios, se puede llegar a tener un máximos de 6 threads. Para 3 radios, el valor aumenta a 8. Y para 4, la cantidad máxima es 10.

$$Max_{threads} = 2 + 2 \cdot Radios$$

4.1.3. Pruebas de estrés

En palabaras simples, este tipo de prueba busca forzar una falla del sistema a través del consumo excesivo de sus reccursos. En virtud de lo descrito, el testing a realizar incluirá un estudio del uso de la memoria por parte de la aplicación, cuando se ve sometida a analizar un amplio rango de radioemisoras a la vez. En una primera versión de esta prueba se van a monitorear 25 radios online, durante 20 minutos, luego el número de radios se extenderá a 50, 75 y finalmente 100.

Cabe señalar que para implementar este test es necesario crear un archivo de texto plano que contenga la URL a las 25, 50, 75 y 100 radioemisoras que se fiscalizarán. En este punto, es necesario hacer énfasis

en que al no disponer de la programación que se transmite en esas radios, lo esperable es tener una mínimo porcentaje de canciones reconocidas, por lo que esto resulta un buen ambiente para comprobar el contexto previamente descrito en 3.3.4.C.

4.2. Implementación

Antes de realizar los tres casos de prueba descritos en el apartado anterior, es necesario detenerse a explicar el proceso de carga del recopilado de canciones que se utilizaron para generar los fingerprints y almacenarlos en la base de datos. Cabe señalar que Dejavu extrae el título de la canción desde el mismo nombre del archivo para indicar el grupo al que pertenecen los fingerpirnts, y como resultado de esto, es necesario someter estas canciones por una etapa de procesamiento con el fin integrar datos como intérprete y título de la canción en cada archivo de formato mp3.

La conversión y limpieza de esta información se facilita con el uso de TagScanner (Versión 6.0.15) , un software libre editor de tags que se utilizó para renombrar los archivos de audio partiendo de sus etiquetas originales, permitiendo editar de manera masiva dentro de un mismo directorio. Inclusive, si las canciones no poseen esta información como metadatos, la aplicación presenta gran variedad de funcionalidades para facilitar el labor de rellenar los campos requeridos para el funcionamiento de Dejavu.

Luego de identificar los archivos cuyos nombres no cumplían con el formato deseado, y editarlos según fuese necesario, se procedió a ejecutar *crearBDD.py* para insertar los fingerpirnts en la base de datos. Como resultado, se generaron registros para 250 canciones, en aproximadamente cinco horas. Estos antecedentes se encuentran disponibles en la tabla 3.7.

Es también relevante especificar el método empleado para transmitir la progrmación diseñada, a través de las radios online personalizadas que se implementaron para este análisis. Esencialmente, el primer paso para montar una radioemisora es contar con con un servidor de radio srteaming. Para esta investigación, se utilizó Listen2MyRadio, compañía que ofrece el servicio gratuito de streaming de radio, video, y hosting de sitios web. Tras crear una cuenta, la plataforma nos brinda los elementos necesarios como IP, puerto, y constraseña, los cuales serán necesarios para transmitir desde el software que encola los archivos de audio que serán emitidos.

En segundo lugar, se procede a hacer la instalación de Winamp, un reproductor multimedia que contendrá la progrmación radial a transmitir, además de uno de sus plugin llamado Shoutcast DSP. Este último es el que permite conectarse con el servidor para efectuar el streaming. Tras la instalación de los programas descritos, solo resta configurar e ingresar los datos proporcionados por Listen2MyRadio.

Una vez transmitiendo las canciones por las radioemisoras, es posible proceder a relizar la fiscalización con el listado de canciones que se ha elaborado. En consecuencia, se debe diseñar el archivo de entrada que

contendrá los links de streaming, junto al nombre de la radio. De esta manera el programa creará un archivo de salida para cada radioemisora, donde imprimirá el resultado del reconocimiento musical. Adicionalmente, se debe configurar el archivo *cnf.SAMPLE* el cual contiene la información necesaria para establecer la conexión con la base de datos.

Un último factor a considerar antes de ejecutar el programa consiste en definir las variables que recibe el archivo .py, particularmente para el caso en que se monitorean múltiples radios paralelamente.

- **nombreArchivo:** ruta al archivo de entrada .txt que contiene nombre y url del streaming de la radio

- **delay:** tiempo que transcurre entre generación de extractos para una misma radio. Para las tres pruebas definidas, el valor utilizado es de 90 segundos. La elección de tiempo se debe a la estimación que se hizo de la duración promedio de una canción, que consiste en aproximadamente 3 minutos 30 segundos. Esto indica que en el mejor de los casos, si la canción es reconocida, el archivo de salida contendría tres líneas consecutivas con el mismo título de canción y artista.

- **segundosExtracto:** duración de los fragmentos generados a partir de la transmisión de la radio online, con un valor igual a 10. Esta cantidad fue escogida en base a los análisis de funcionalidad realizados a Dejavu project en el capítulo 2. Si se aumenta este valor, se tendrá mayor precisión en la respuesta, pero sería a costa de reducir la eficiencia en tiempo de cómputo y las posibilidades de escalabilidad del sistema completo.

- **hilosReconocimiento:** corresponde a la cantidad de threads utilizados para el reconocimiento. Para las pruebas, esta valor coincide con la cantidad de radios que se analizan.

- **tipoTiempo:** esta variable actúa en conjunto con *tiempo*. En el caso en que tipoTiempo se defina como *cronometro* el parámetro *tiempo* debe ser detallado como el número de horas que se desea fiscalizar. En caso contrario, si *tipoTiempo* es *calendario*, el campo *tiempo* debe definirse con formato *datetime*.

La ejecución del programa para examinar varias radios utiliza la siguiente sentencia:

```
python fiscalizarRadiosParalelo.py "entrada25Radios.txt" 90 10 25 cronometro 20
```

4.3. Características del equipo

La tabla 4.1 muestra las características principales de los equipos utilizados para las distintas etapas de testing a las que se sometió la plataforma desarrollada.

Tabla 4.1: Resumen de los equipos utilizados para testear la plataforma de fiscalización.

Equipo	1 Notebook Acer	2 Notebook Lenovo	3 Notebokk MSI	4 Netbook Samsung
Sistema operativo (Windows)	8.1	10 Home	10 Home	7 Starter
Tipo sistema	64 bits	64 bits	64 bits	32 bits
Tipo procesador	Intel Core i5	Intel Core i5	Intel Core i7	Intel Atom
Velocidad del procesador	2.20 GHz	2.50 GHz	2.60GHz	1.67GHz
RAM [GB]	4	8	8	1

4.4. Resumen de versiones

- Base de datos MySQL versión 5.7.
- Python 2.7.
- Módulos de Python
  - Pyaudio 0.2.9 para captar audio desde el micrófono.
  - Pydub 0.16.5 para la manipulación de audio
  - Numpy 1.11.1 para aplicar la transformada rápida de Fourier (FFT) al audio.
  - Scipy 0.17.1 utilizada para la búsqueda de peak en los fingerprints.
  - Matplotlib 1.5.1, usada para el trabajo en espectrogramas.
  - MySQLdb 2.0 para la interacción entre Python y la base de datos MySQL.
  - Sistema de control de versiones GIT 2.11.0.
  - Winmap 5.6.
  - ShoutCast 2.3.5.
  - TagScanner 6.0.15.
- Anaconda 4.1.1. Distribución de Python gratuita, que además incluye el entorno de desarrollo Spyder.
- Ffmpeg versión N-81555-g496d97f. Framework multimedia para la conversión de archivos de audio.

5 | Resultados

En este apartado se presentan los resultados obtenidos tras realizar las pruebas de rendimiento a la plataforma de fiscalización, las cuales fueron definidas en el capítulo 4. Dentro de las variables estudiadas se distinguen uso de memoria, tiempo de respuesta del algoritmo de reconocimiento Dejavu project, valores de confianza de las canciones identificadas, y clasificación de las respuestas según hubo o no coincidencia con los registros de la base de datos.

Se debe señalar que dada la diferencia de implementación de los diversos escenarios examinados, las métricas anteriormente señaladas no fueron aplicadas en todas las revisiones.

5.1. Pruebas de estabilidad

La primera prueba realizada involucra la creación de una radio online personalizada, debido a la importancia de conocer la programación emitida para comprobar la veracidad de las respuestas entregadas por la plataforma.

Como se señala en el capítulo 4 el tiempo de fiscalización comprende 24 horas, más un adicional de 10 minutos que permite corroborar que el sistema es capaz de funcionar por más de un día. Dentro de ese rango de 1450 minutos totales, se crearon 966 extractos de música cada uno de 10 segundos de duración, que según la programación diseñada, 203 de ellos corresponden a fragmentos obtenidos a partir de canciones nacionales. Los otros 763 concuerdan con música internacional o programas radiales.

5.1.1. Clasificación de respuestas obtenidas por la plataforma

Considerando la solución entregada por la plataforma, con los 966 extractos de la muestra se ha diseñado el gráfico de la figura 5.1. Como se puede apreciar, se ha clasificado la salida del programa en:

Con respuesta: hace referencia al mensaje devuelto por Dejavu project. Esto quiere decir que si hubo alguna coincidencia entre fingerprints, el algoritmo indica título y artista de la canción identificada.

Sin respuesta: abarca todos los fragmentos cuya búsqueda no tuvo concurrencia con los registros de la base de datos, y por tanto, Dejavu no logró identificar un nombre para la canción.

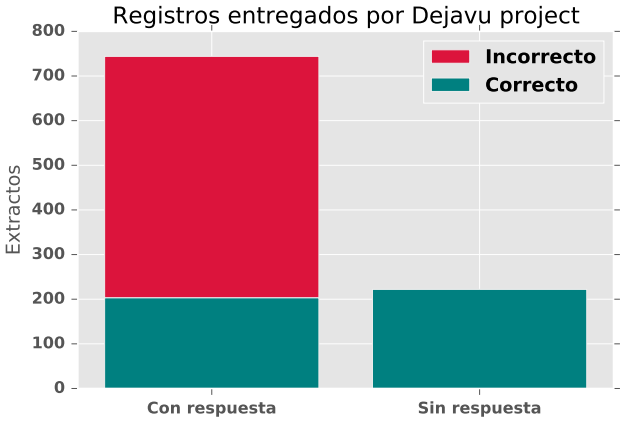

Figura 5.1: Clasificación de respuestas al fiscalizar 1 radio online.
El gráfico indica el número de impresiones al archivo donde Dejavu identificó canciones, y cuando no lo hizo.

De los datos obtenidos se desprende que un total de 744 respuestas de Dejavu contenían algún nombre de canción identificada. Se debe recalcar que 203 de éstos, son parte de canciones que existían en la base de datos y el identificador obtenido coincidía con la canción transmitida, en consecuencia, la respuesta es clasificada como correcta. Empero, las canciones de los cuales fueron extraídos los 541 restantes, no eran parte de la base de datos, vale decir, los 541 extractos deberían haber caído en la categoría de Sin respuesta, ya que esa música no era parte de la biblioteca de la plataforma. Dado que Dejavú asoció el título de alguna canción a estos extractos sin que existiera real coincidencia, se cataloga la respuesta como incorrecta.

Si bien este último valor parece alarmante para el objetivo de la plataforma, se deben considerar otros factores a la hora de evaluar la información obtenida, puesto que hay dos componentes adicionales que estudiar antes de aceptar la resolución entregada por Dejavu. Estos parámetros son la confianza y la cantidad de veces que el título de una canción se repite sucesivamente.

Tras una revisión parcial de la programación obtenida de la transmisión, se aprecia que para el caso en que la canción es identificada correctamente, el título y artista señalado en el archivo de salida se repite una cantidad de veces que depende de la duración de la canción. Para el caso de la música que no es parte de la base de datos, los nombres obtenidos son diferentes entre tiempos consecutivos, y además, la confianza de las respuestas tienen valores muy bajos. Por tanto, es fácil identificar la poca credibilidad de esa respuesta.

Para analizar cuantitativamente la certeza de la solución, se han elaborado gráficos de distribución para determinar entre qué rangos se sitúan los valores de confianza obtenidos por la plataforma, tanto de aciertos como de respuestas incorrectas. Ya que esto permite además, identificar un umbral sobre el cual se debería aceptar la respuesta de Dejavu como correcta.

5.1.2. Confianza

Referente a los aciertos, la imagen 5.2 señala que el 75 % de la muestra tiene valores de confianza por sobre el 100, mientras que del 25 % restante, el mínimo tiene un valor de certeza igual a 5. Al buscar dicho parámetro en el archivo generado por el programa, se ha identificado que aquel extracto corresponde a la parte final de la canción, y fragmentos consecutivos y previos a este en tiempo, presentan valores de confianza alta.

Respecto a los extractos considerados incorrectos el 100% de la muestra no supera un valor de confianza igual a 20.

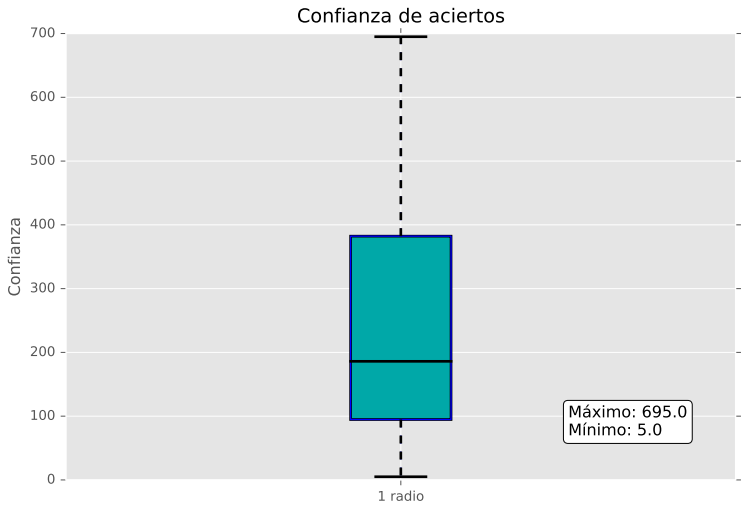

Figura 5.2: Valores de confianza de los aciertos al fiscalizar 1 radio online.
La imagen señala la cantidad de fingerprints coincidentes al realizar la prueba de estabilidad.

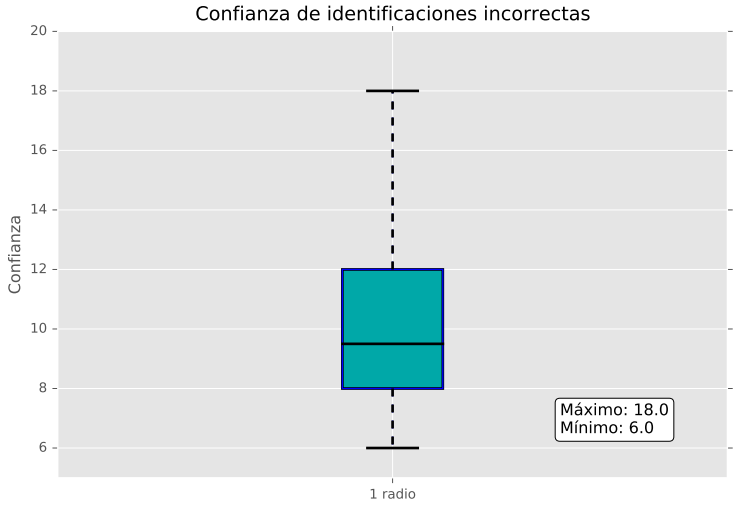

Figura 5.3: Valores de confianza de identificaciones incorrectas al fiscalizar 1 radio online.
El gráfico muestra la confianza para las respuestas consideradas incorrectas.

Con esta información recopilada, es posible generar la hipótesis que respuestas que tengan confianza mayor a 100 deben ser consideradas correctas. Para el caso en que la confianza es menor, se debe evaluar la cantidad de repeticiones sucesivas. Si dicho valor es mayor a 2, es esperable que esa canción haya sido identificada correctamente.

5.1.3. Tiempo de respuesta

Para finalizar, se ha incluido el análisis del tiempo que tarda Dejavu en entregar una respuesta. Como se aprecia de las figuras 5.4 y 5.5, no existe una diferencia evidente entre los resultados obtenidos. La mediana de los aciertos se ubica entre los $11 - 12$ segundos, mientras que para las respuestas consideradas como incorrectas, este valor disminuye hasta 10,1 segundos.

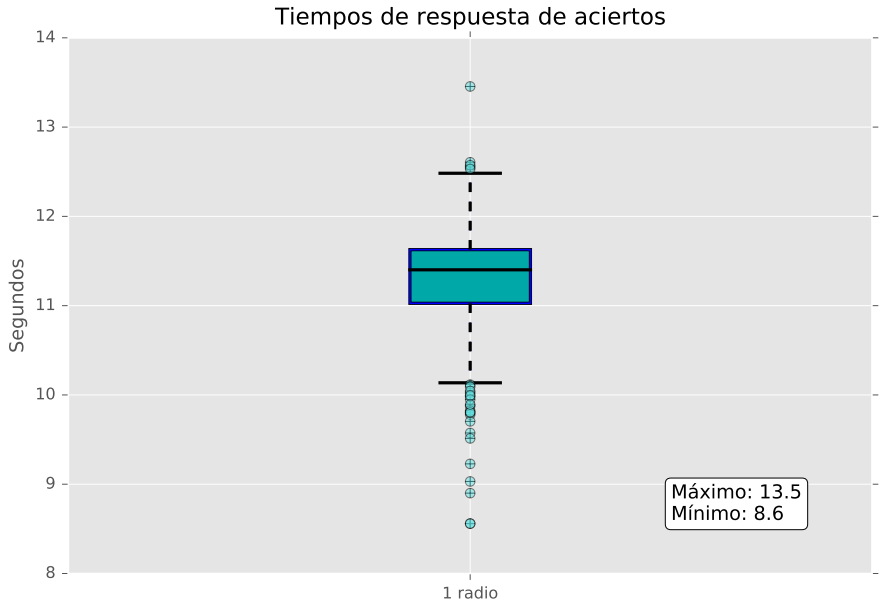

Figura 5.4: Tiempo de respuesta de los aciertos al fiscalizar 1 radio online.
La imagen indica la distribución de los tiempos de respuesta de las identificaciones correctas al realizar la prueba de estabilidad.

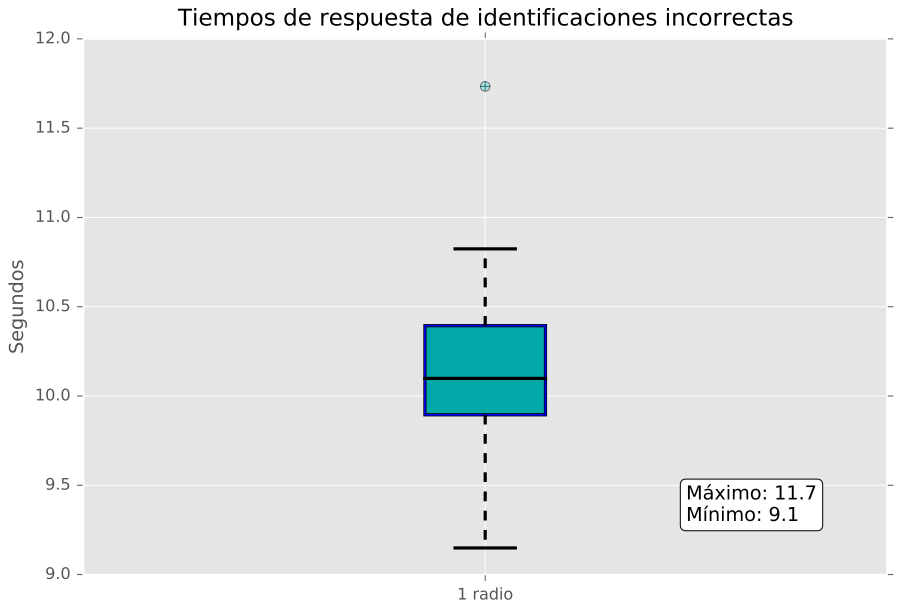

Figura 5.5: Tiempo de respuesta de identificaciones incorrectas al fiscalizar 1 radio online.
El gráfico representa los tiempos de respuesta para la prueba de estabilidad, de las soluciones consideradas incorrectas.

5.2. Pruebas de carga

El objetivo de este test consiste comprobar el rendimiento de la aplicación conforme aumenta la carga en el sistema. Con el propósito de analizar como evoluciona el comportamiento de los parámetros ya utilizados en la sección anterior, también se ha optado por utilizar radios personalizadas. Específicamente, la programación emitida solo incluyó canciones que existían en el repertorio de prueba de la base de datos.

El monitoreo se realizó por 20 minutos, durante los cuales se extrajeron 14 fragmentos de música. Tras comparar el listado de canciones emitidas con las respuestas entregadas por Dejavu, se ha comprobado un 100 % de identificaciones correctas para todas las pruebas, inclusive, para aquellas que involucraban covers.

5.2.1. Confianza

La confianza del algoritmo de Dejavu se define en base al número de fingerprints coincidentes, por lo que independiente de la cantidad de radios que se analicen, este valor no debe verse afectado. Un factor que puede influir en dicha cantidad es el intervalo de tiempo respecto a la canción original en que se creó el fragmento de música. Ésto, debido a que la conformación de tuplas que dan origen a cada fingerprint no tienen una distribución homogénea a lo largo de un archivo música, por ende, la cantidad de estos patrones característicos solo tendrán correlación con la escala de tiempo.

Como se percibe del gráfico 5.6, los valores de confianza no varían en función del número de radios analizadas. Si bien los extractos generados corresponden a las mismas obras musicales, el tiempo de su creación respecto al origen varía para cada revisión. Por consiguiente, las cantidades difieren.

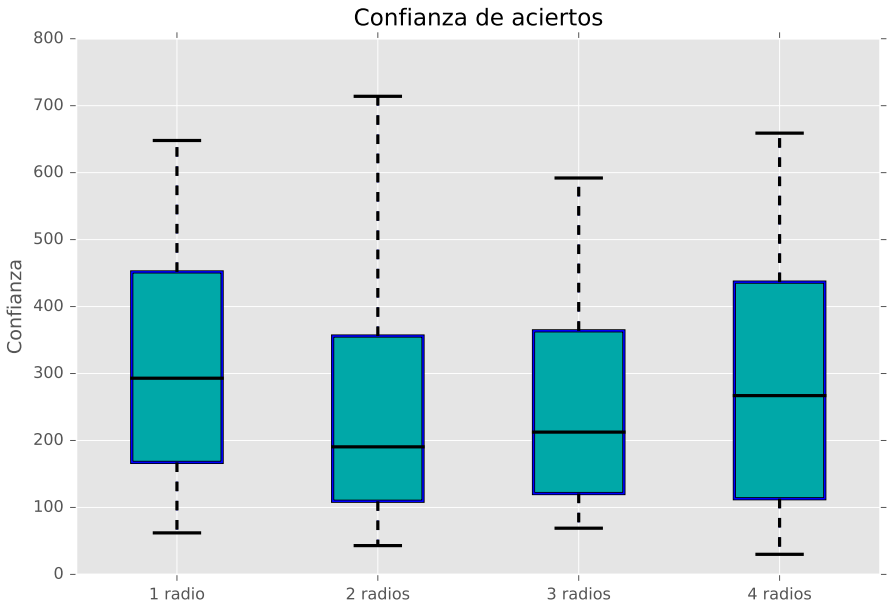

Figura 5.6: Valores de confianza de los aciertos al fiscalizar múltiples radios personalizadas. La imagen señala la cantidad de fingerprints coincidentes al realizar la prueba de carga.

5.2.2. Tiempo de respuesta

Con respecto al tiempo de respuesta, se puede observar una clara diferencia entre cada muestra a partir de la imagen 5.7. Sin embargo, como fue señalado en 5.2.1, el moemento en que se produce el archivo de audio de búsqueda influye directamente en la cantidad de fingerpints que serán analizados, por ello, existe una gran diferencia entre los valores observados. Si es importante recordar, según lo observado en

No es posible realizar alguna otra aseveración salvo indicar que para las muestras de 1 y 3 radios, el tiempo de respuesta es menor, por lo que es esperable que las cantidades de fingerprints a los que se les buscó y encontró coincidencia también sea inferior. Al comparar las figuras 5.6 y 5.7 se reafirma esta idea.

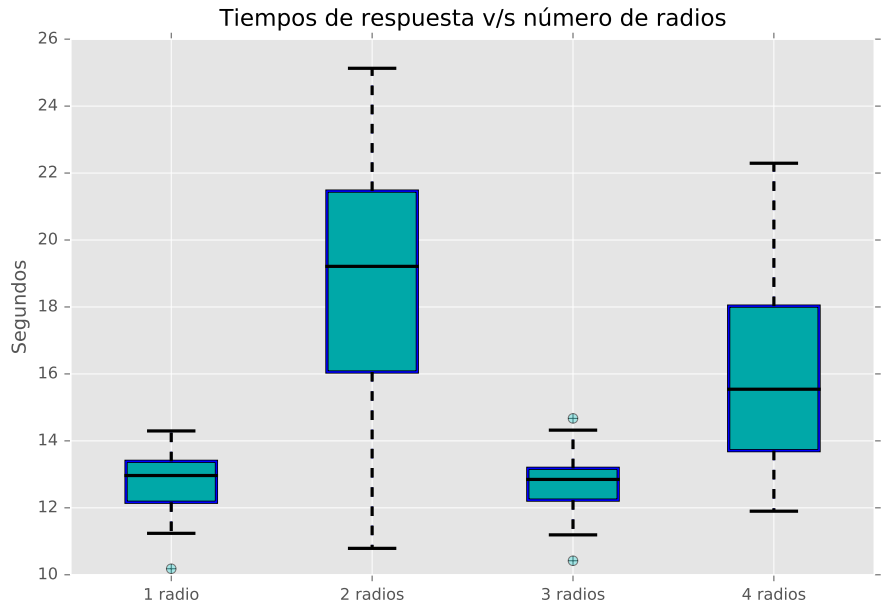

Figura 5.7: Tiempos de respuesta de los aciertos al fiscalizar múltiples radios personalizadas.
La imagen indica la distribución de los tiempos de respuesta de las identificaciones correctas al realizar la prueba de carga.

5.2.3. Uso de memoria

Una característica adicional que se ha decidido considerar es el uso de memoria durante la ejecución de la paltaforma. En la figura 5.10 se observan las cuatros pruebas cuya cantidad de radios va en aumento. Como se ha especificado previamente, la cantidad de máxima de threas para cada muestra son 4,6,8 y 10 respectivamente. El máximo de memoria utilizada depende estrechamente del extracto de música descargado por lo que no es factible manifestar alguna observación al respecto.

Lo que se debe destacar es el ligero aumento de los MB máximos utilizados, a medida que la cantidad de threads aumenta para cada muestra, cuya diferencia es cercana a las diez unidades. También se debe aclarar

que el valor denominado Δt en los gráficos hace referencia al intervalo que transcurre desde el momento en que se finaliza la extracción de audio, y la cola con archivos por reconocer se vacía totalmente. Como se aprecia a simple vista, este valor no supera los sesenta segundos. Esto quiere decir que desde el momento en que se genera el archivo .mp3 hasta que se logra identificar, el tiempo de procesamiento no supera los 90 segundos, pues en caso contrario, la cola no se vaciaría a tiempo, y luego de alcanzar la fecha límite para descargar el audio, la aplicación seguiría ejecutándose.

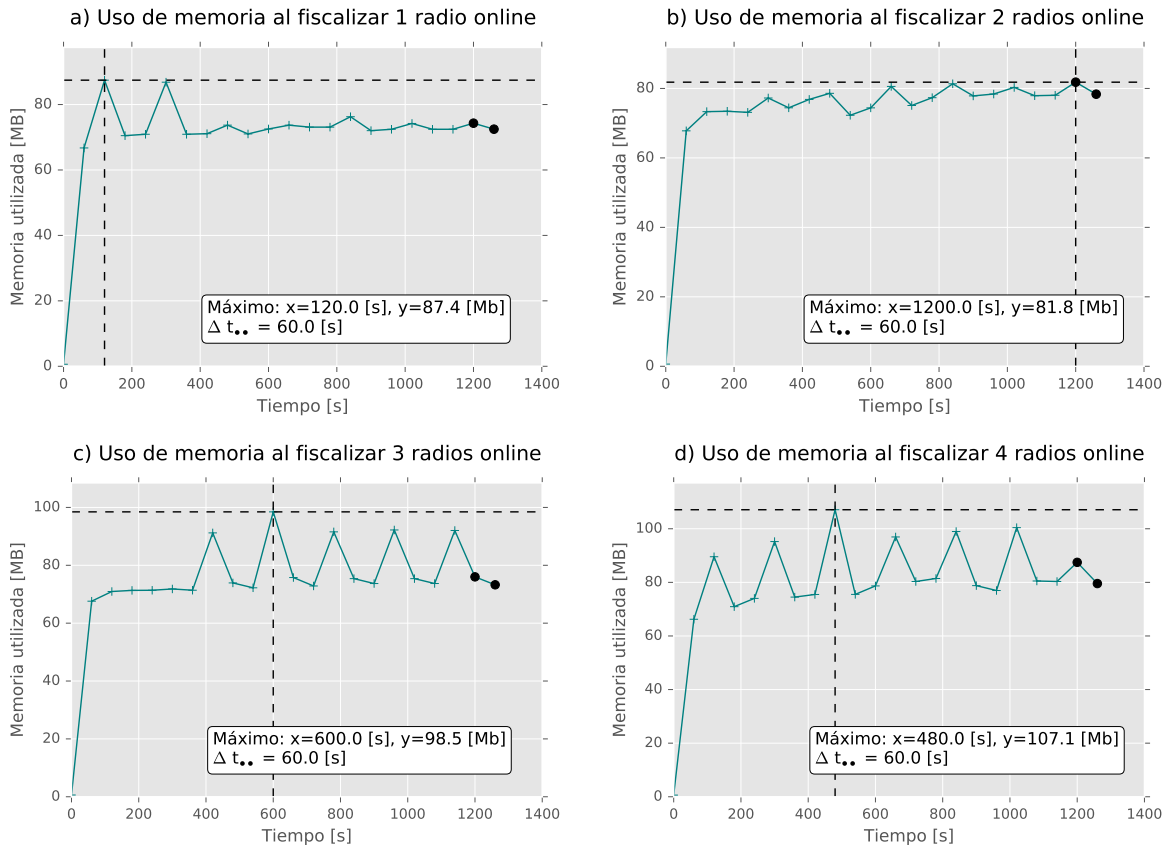

Figura 5.8: Uso de memoria al fiscalizar múltiples radios personalizadas.
Uso de memoria de la prueba de carga, según el número ascendente de radios monitoreadas.

5.3. Pruebas de estrés

Las últimas pruebas realizadas se diseñaron con la finalidad de encontrar el umbral por sobre el cual la aplicación no rendirá eficientemente. Se debe recalcar que dada la complejidad de comparar la programación desconocida de todas las radios analizadas con la respuesta obtenida, no es factible realizar un estudio sobre los valores de confianza; solo se examinarán los tiempos de respuesta y el uso de memoria.

5.3.1. Tiempo de respuesta

Tal como ocurre con los resultados de 5.2.2 los tiempos de respuesta varían notoriamente entre las muestras, imposibilitando encontrar alguna relación entre el número de radios y el tiempo transcurrido para encontrar una solución. En la figura 5.9 se perciben estos planteamientos. Es más, el comportamiento es tan inconsistente, que la muestra que fiscaliza 100 radios presenta la mayoría de sus tiempos de respuesta bajo los 20 segundos, y muchos de sus puntos son considerados outliers.

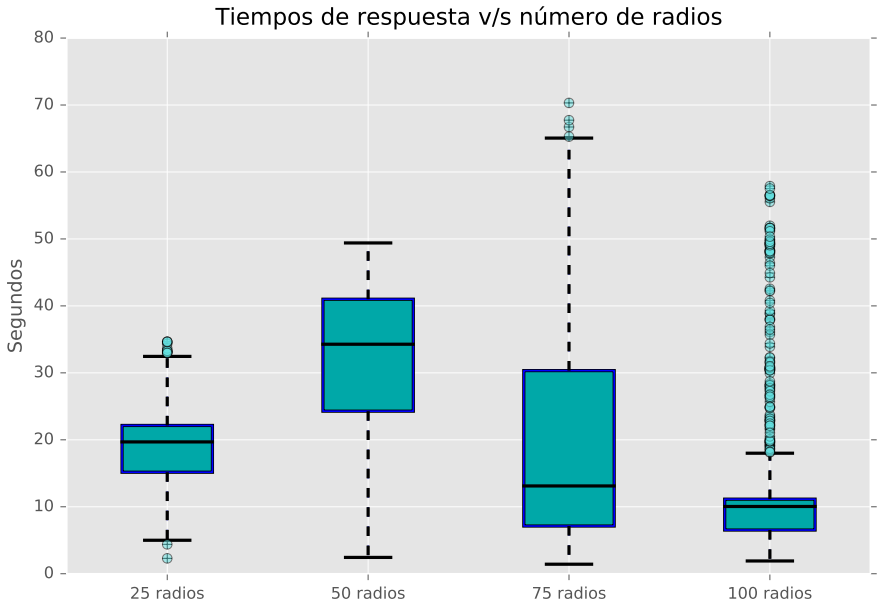

Figura 5.9: Tiempos de respuesta al fiscalizar múltiples radios nacionales.
El gráfico representa la distribución de tiempos de respuesta de la prueba de estrés, al aumentar el número de radios analizadas paralelamente.

5.3.2. Uso de memoria

La primera afirmación que puede establecerse con respecto a la figura 5.10 es que si existe una relación entre la cantidad de radios monitoreo y el tiempo de respuesta de Dejavu. Al fiscalizar 25 y 50 radios (52 y 102 threads máximos respectivamente), el tiempo que transcurre entre la finalización de obtención de audio y el vaciado total de la cola, es de 60 segundos. Por tanto, mantener la aplicación ejecutándose por tiempo indefinido no generará problemas de memoria. Inclusive, a simple vista se aprecia que luego de cada peak, el uso de memoria se estabiliza hasta que termina el *delay* definido.

Con respecto a las muestras donde se analizan 75 y 100 radios, se observa claramente que el tiempo entre la fecha límite de monitoreo y el vaciado total comienza a incrementar. El principal inconveniente con este comportamiento no es solo el tiempo que se debe esperar para obtener la programación de una radio online; también debe considerarse que los archivos .mp3 que se mantienen en la cola irán aumentando a lo

largo del tiempo, y por ende, el espacio en disco duro también. De no tener un control adecuado de este flujo, no solo la RAM se vería afectado, sino también el espacio para almacenamiento del equipo.

En sisntesis, y desde la perspectiva de los objetivos definidos en 1.3.1 y 1.3.2, este último análisis permite comprobar la factibilidad de utilizar el sistema desarrollado para el monitoreo de las radioemisoras online, cuando esta cantidad es cercana a 50.

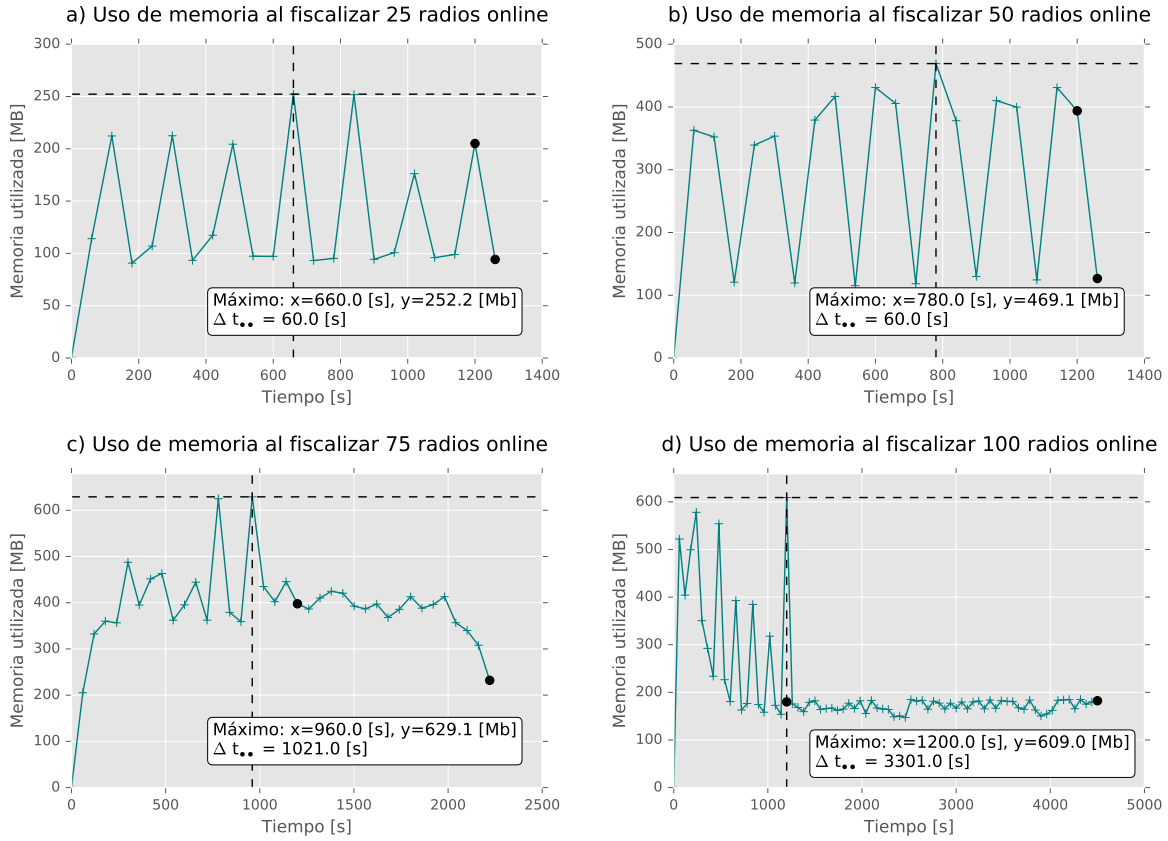

Figura 5.10: Uso de memoria al fiscalizar múltiples radios nacionales.
Uso de memoria de la prueba de estrés, según el número ascendente de radios monitoreadas.

Conclusiones

En este trabajo de investigación se ha elaborado una propuesta de diseño para la implementación de una plataforma de monitoreo radial, que sea capaz de fiscalizar diversas radios online nacionales de manera simultánea. Para desarrollar el prototipo se analizaron proyectos open-source con la finalidad de emplear un algoritmo de reconocimiento acústico, el cual permita elaborar un listado con la programación emitida por las estaciones examinadas.

El funcionamiento de estos algoritmos recae en las características intrínsecas que pueden obtenerse a partir de un archivo de audio, concepto denominado huellas acústicas (*Acoustic fingerprints*). La elaboración de estos identificadores de cada canción requieren la aplicación de la Transformada Corta de Fourier. Así, al obtener una señal discreta de la onda de sonido, es posible extraer los peaks de audio de una canción, los que se obtienen al procesar el espectrograma.

En consecuencia, el proceso para reconocer un extracto de música comprende una primera etapa de aprendizaje, mediante la cual, la plataforma es capaz de almacenar las propiedades inherentes de las obras musicales en una base de datos. Estos valores son utilizados en una segunda fase, que tiene el propósito de encontrar coincidencias al cotejar un archivo de audio con el repertorio de prueba.

Con la finalidad de deducir el periodo aproximado que tardaría la plataforma en cargar la biblioteca de fingerprints oficial, constituida por cerca de 80.000 canciones, se analizaron los tiempos transcurridos al generar bases de datos con una cantidad variable e incremental de canciones. Los primeros resultados no fueron favorables debido a que el plazo indicado superaba los nueve meses.

Tras optar por modificar las variables que conforman la estructura del algoritmo, se utilizó el mismo análisis anterior, para comparar el comportamiento de los tiempos de creación en función de los parámetros de Dejavu project, utilizado para el diseño de la aplicación. Estos son: amplitud umbral (*DEFAULT_AMP_MIN*) por sobre la cual un peak de audio es considerado para la formación de figerprints, superposición (*DEFAULT_OVERLAP_RATIO*) debido al tamaño de la función ventana utilizada, cantidad de peaks cercanos (*DEFAULT_FAN_VALUE*) para modelar las tuplas de frecuencias y tiempo, y tamaño del vecindario (*PEAK_NEIGHBORHOOD_SIZE*).

De la extrapolación de estos resultados se desprende la nueva configuración del programa de reconocimiento, la cual es capaz de elaborar el repertorio de prueba en un tiempo estimado de 64 días. Una vez realizadas las configuraciones del algoritmo de extracción de fingerprints, se recopilaron canciones nacionales, covers incluidos, para conformar la base de datos que se aplica para futuras validaciones.

Seguidamente se procede a desarrollar la plataforma de monitoreo. En primera instancia, los requerimientos abarcan fiscalizar una única radioemisora, y para lograrlo, se opta por la utilización de hilos (*Threads*), que se encargan de generar múltiples fragmentos de audio descargados intermitentemente desde el streaming radial. Simultáneamente, estos archivos que han sido previamente indexados a una cola, son captados por el hilo principal que se encarga de utilizar Dejavu para lograr la identificación del extracto musical.

A fin de abordar la solicitud de controlar paralelamente un conjunto de radios, se realizaron leves modificaciones al código fuente, que permite a la aplicación producir hilos de ejecución según las URL pasadas como parámetro de entrada. De este modo, el sistema se encarga de completar para las respectivas estaciones radiales, un archivo de texto plano que contempla las respuestas de Dejavu project con la posible distinción de título y artista de la canción sintonizada.

Una vez finalizado el desarrollo de la plataforma, se ejecutaron las pruebas de rendimiento con el propósito de corroborar el cumplimiento de los objetivos planteados. Debido a la necesidad de conocer la parilla programática de las radioemisoras analizadas, fue primordial diseñar radios personalizadas que facilitaran la comprobación entre la música emitida y la respuesta obtenida por el sistema.

La primera verificación efectuada incumbe una prueba de estabilidad, sobre una radio online pesquizada por 24 horas y 10 minutos. De los 966 extractos de música extraídos de la transmisión, 203 conciernen a música nacional incluída en la base de datos, y la totalidad de éstos fue reconocida correctamente por la aplicación. Los 763 restantes debían ser resueltos como "Sin identificación", a pesar de ello, el 71 % fue asociado por el algoritmo a alguna canción de la biblioteca de prueba; mientras que solo 222 cayeron en esa categoría. Para contrarrestar este deficiente rendimiento, se evaluaron otros valores entregados por las resoluciones de Dejavu. La variable confianza, que representa la cantidad de fingerprints coincidentes en una búsqueda, permite identificar un umbral que debe superarse para que el nombre de la canción, determinada por el programa, sea considerada una respuesta correcta.

Conforme al test de carga, no es posible garantizar si existe alguna conexión entre el número de radios y los tiempos de respuesta del sistema. La arbitrariedad de los datos obtenidos no permiten establecer alguna relación, salvo la razón a la cual aumenta el uso de memoria conforme se añaden emisoras online a la fiscalización. Se debe enfatizar además, que los valores de confianza conseguidos entre las muestras de las distintas evaluaciones tenían una distribución similar.

Posteriormente, se llevan a cabo las pruebas de estrés en una ambiente donde se dificulta conocer la

programación del considerable número de estaciones revisadas, ergo, no es posible comparar las respuestas. En contraste con los estudios ya detallados, la ejecución de esta experimento se enfoca en los recursos utilizados, y no la veracidad de los títulos de canciones identificadas. Con 25 y 50 enlaces de streaming la plataforma tiene un rendimiento estable, es capaz de extraer, reconocer, y exportar la respuesta del reconocimiento musical en menos de 50 segundos. Sumando esta cantidad al intervalo de tiempo en que se producen los fragmentos de audio, limitado a 20 segundos, se confirma que el lapso en que se converge a una solución es inferior al retraso (*delay*) de 90 segundos definido para toda la experiencia. Por tanto, hay un flujo idóneo entre ingreso y salida de archivos desde la cola.

Por el contrario, los casos de prueba que involucran 75 y 100 radios presentan mayores tiempos de respuesta, debido a lo cual, los hilos encargados de llamar a la función de reconocimiento de Dejavu no logran encontrar la solución a la brevedad, provocando que sus ejecuciones perduren ininterrumpidamente, incluso superando el tiempo de la fecha límite establecida para el monitoreo.

Con las implicancias de este desarrollo, se establece que el margen ideal en el número de radios a fiscalizar es de 50 emisoras, siempre que se mantengan los parámetros previamente definidos. No obstante, la posible modificación de los valores de estos criterios permitiría encontrar nuevas configuraciones para aumentar la limitación de radios online controladas.

Trabajo futuro

Si bien en este trabajo se ha conseguido el objetivo propuesto, desarrollar un prototipo que permita automatizar la fiscalización de transmisiones online de radios nacionales, las limitaciones en el número de emisoras pueden ser superadas. La acción más factible de poner en práctica consiste en medir el comportamiento de la plataforma al alterar paulatinamente los valores de configuración.

Los posibles valores sujetos a cambio son, por una parte, el tiempo de los extractos de música. Estudios previos a la experimentación manifestaron que fragmentos de hasta cinco segundos producen respuestas satisfactorias. De utilizar mencionado valor, el tiempo máximo para la descarga de audio no sobrepasaría los 10 segundos, por ende, al sumar este intervalo con el tiempo empleado en la búsqueda de coincidencias, se estaría más cerca de alcanzar el flujo ideal obtenido en la prueba de 50 radios.

El segundo factor a considerar es aumentar la cantidad de hebras que procesan los archivos de audio de la cola, ya que en este desarrollo se optó por igualar esa medida con el número de radios. La deciSión fue tomada con la finalidad de controlar el acceso de los hilos a los recursos compartidos, como era el caso de los archivos de salida de cada URL examinada.

Un componente adicional que es apropiado investigar, corresponde a la cantidad definida para la variable *delay*. De aumentarla, se estaría otorgando un mayor tiempo de espera hasta generar el siguiente

extracto de música, y con esto, se brinda la oportunidad al sistema de encontrar una solución con un tiempo de respuesta inferior a este margen.

Para finalizar, se debe mencionar que una segunda línea de desarrollo para continuar esta herramienta constituye diseñar un modelo que analice los archivos que presentan la programación radial y cuantifique los porcentajes de música chilena emitida, logrando finalmente conformar la herramienta de monitoreo necesaria para fiscalizar el cumplimiento de la lay 20810.

Bibliografía

[1] Ministerios de Educación. Fija porcentajes mínimos de emisión de música nacional y música de raÍz folklórica oral, a la radio difusión chilena. https://www.leychile.cl/Navegar?idNorma=1076447, 2015. Visitada en junio de 2016. (document)

[2] Sociedad Chilena del Derecho de Autor. Funciones de la sociedad chilena del derecho de autor. http://www.scd.cl/www/index.php/socios-y-afiliados/. Visitada en noviembre de 2016. (document)

[3] Sociedad Chilena del Derecho de Autor. Posición ante comunicado radial. http://www.scd.cl/www/index.php/noticias/posicion-ante-campana-radial-de-archi/, 2014. Visitada en junio de 2016. (document)

[4] Open Business. Desafíos de la gestión colectiva en chile. http://openbusinesslatinamerica.org/2013/04/22/desafios-de-la-gestion-colectiva-en-chile/, 2013. Visitada en junio de 2016. (document)

[5] Teletrece online. Ley de música chilena. http://www.t13.cl/noticia/entretencion/cultura/archi-ley-de-musica-chilena-vulnera-libertad-de-expresion, 2016. Visitada en junio de 2016. (document)

[6] Sociedad Chilena del Derecho de Autor. Sistema de muestreo. http://www.scd.cl/www/index.php/nuestra-sociedad/scd-transparente/aspectos-corporativos/distribucion-de-derechos/sistema-de-muestreo/, 2015. Visitada en junio de 2016. 1.2

[7] Consejo Nacional de la Cultura y las Artes. Aprueba bases de licitación pública del fondo para el fomento de la música nacional para la adquisición de un sistema de monitoreo radial. http://transparenciaactiva.cultura.gob.cl/uploads/marcoNormativo/e209759403a33ec4f84e9bf1b3b4f1112453d364.pdf, 2011. Visitada en agosto de 2016. 1.2

[8] La Tercera Online. Los hitos, ganadores y pendientes de la ley que cambió el dial. http://www.latercera.com/noticia/el-ano-del-20-los-hitos-ganadores-y-pendientes-de-la-ley-que-cambio-el-dial/, 2016. Visitada en agosto de 2016. 1.2

[9] El Mercurio online. Música chilena en las radios. http://www.elmercurio.com/blogs/2016/04/16/40998/Musica-chilena-en-las-radios.aspx, 2016. Visitada en enero de 2017. 1.2

[10] Organización Mundial de la Propiedad Intelectual. Diario oficial nº 29.159. Artículo 2, punto viii, 1975. 2.1

[11] Dibam. Historia de la propiedad intelectual en chile. http://www.propiedadintelectual.cl/623/w3-propertyvalue-37351.html, 2010. Visitada en diciembre de 2016. 2.1

[12] Youtube-ContentID. Administración de derchos, content id. https://support.google.com/youtube/answer/2797370?hl=es-419&ref_topic=2778544, 2012. Visitada en octubre de 2016. 2.1.1

[13] SoundCloud. Identification system. https://blog.soundcloud.com/2011/01/05/q-and-a-content-identification-system/, 2011. Visitada en febrero de 2017. 2.1.2

[14] Twitch. Twitch music library. https://help.twitch.tv/customer/portal/articles/1824967-twitch-music-faq, 2016. Visitada en febrero de 2017. 2.1.3

[15] Monitec. Servicios monitec. http://monitec.com/, 2014. Visitada en abril de 2017. 2.1.4

[16] BMAT. Vericast. http://www.bmat.me/es/products/vericast-es/, 2012. Visitada en abril de 2017. 2.1.5

[17] Pedro Cano, E Batle, Ton Kalker, and Jaap Haitsma. A review of algorithms for audio fingerprinting. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 169–173. IEEE, 2002. 2.3

[18] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *Journal of VLSI signal processing systems for signal, image and video technology*, 41(3):271–284, 2005. 2.3.1.1

[19] Avery Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006. 2.3.1.1

[20] Shumeet Baluja and Michele Covell. Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern recognition*, 41(11):3467–3480, 2008. 2.3.1.1

[21] Jaap Haitsma, Ton Kalker, and Job Oostveen. Robust audio hashing for content identification. In *International Workshop on Content-Based Multimedia Indexing*, volume 4, pages 117–124. University of Brescia, 2001. 2.3.1.2

[22] Stanford University. Sonification seminar. https://ccrma.stanford.edu/courses/120-fall-2003/lecture-5.html, 2015. Visitada en abril de 2017. 2.3.1.2

[23] Juergen Herre, Eric Allamanche, Oliver Hellmuth, and Thorsten Kastner. Robust identification/fingerprinting of audio signals using spectral flatness features. *The Journal of the Acoustical Society of America*, 111(5):2417–2417, 2002. 2.3.2

[24] Eloi Batlle, Jaume Masip, and Enric Guaus. Automatic song identification in noisy broadcast audio. In *IASTED International Conference on Signal and Image Processing*, 2002. 2.3.2

[25] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 2.3.2

[26] Yan Ke, Derek Hoiem, and Rahul Sukthankar. Computer vision for music identification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 597–604. IEEE, 2005. 2.3.2

[27] La Tercera online. Scd lanza nueva base de datos de música chilena para radios. http://www.latercera.com/noticia/ley-del-20-scd-lanza-nueva-base-de-datos-de-musica-chilena-para-radios/, 2016. Visitada en agosto de 2016. 3.3.4

Apéndices

Tabla A1: Valores originales de los parámetros de Dejavu.

Parámetro	F3_3	F3_5
DEFAULT_OVERLAP_RATIO	0,5	0,5
DEFAULT_FAN_VALUE	15	15
DEFAULT_AMP_MIN	10	10
PEAK_NEIGHBORHOOD_SIZE	20	20
PEAK_SORT	True	True
Segundos	322	591
Fingerprints	470.938	865.039
Tamaño [MB]	43,16	82,25

Tabla A2: Tiempo de creación de bases de datos, con los valores originales de Dejavu.

Número de canciones	Tiempo [seg]	Tiempo [min]	Tiempo [hr]
3	322	5,4	0,1
5	591	9,9	0,2
10	2.448	40,8	0,7
50	11.411	190,2	3,2
100	24.938	415,6	6,9
150	39.689	661,5	11,0
200	59.094	984,9	16,4
250	73.706	1.228,4	20,5

Tabla A3: Valores de configuración de las bases de datos de 3 y 5 canciones.

La tabla señala las 26 bases de datos para las cuales se modifica un solo parámetro de configuración, con sus respectivos tiempos de creación, cantidad de fingerprints y tamaño.

Bases de datos	Variable modificada	Valor	Segundos	Fingerprints	Tamaño [MB]
A1_3	DEFAULT_OVERLAP_RATIO	0,10	118	264.963	26,09
A1_5		0,10	226	488.378	47,17
A2_3		0,30	183	340.149	30,11
A2_5		0,30	328	623.970	58,19
A3_3		0,70	779	760.517	66,22
A3_5		0,70	1492	1.389.934	119,31
B1_3	DEFAULT_FAN_VALUE	4	78	101.020	11,06
B1_5		4	144	185.543	19,09
B2_3		7	85	202.019	21,09
B2_5		7	151	371.056	37,16
B3_3		10	137	302.977	27,11
B3_5		10	218	556.456	48,17
C1_3	DEFAULT_AMP_MIN	20	284	455.623	41,16
C1_5		20	455	831.922	84,25
C2_3		30	219	346.760	30,11
C2_5		30	374	663.643	56,19
C3_3		40	125	174.205	17,06
C3_5		40	253	344.135	31,11
D1_3	PEAK_NEIGHBORHOOD_SIZE	30	162	213.056	21,09
D1_5		30	290	396.462	38,14
D2_3		40	262	118.504	13,06
D2_5		40	466	224.778	23,09
D3_3		50	404	75.614	7,06
D3_5		50	720	144.452	14,06
E1_3	PEAK_SORT	FALSE	100	24.725	3,06
E1_5		FALSE	240	46.162	6,06

Tabla A4: Resumen de los resultados de Dejavu al probar 27 extractos de música.

Las pruebas se realizaron con cada base de datos que solo contiene 3 canciones, generando extractos de 5,10, y 15 segundos para realizar el reconocimiento acústico.

Salida		Segundos	A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3	E1	F1
Respuesta Dejavu	Aciertos	5	9	9	9	9	8	8	8	8	8	8	8	8	2	8
		10	9	9	9	9	9	9	9	9	9	9	9	9	1	9
		15	9	9	9	9	9	9	9	9	9	9	9	9	3	9
	Errores	5	0	0	0	0	1	1	1	1	1	1	1	0	0	1
		10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sin respuesta		5	0	0	0	0	0	0	0	0	0	0	0	1	7	0
		10	0	0	0	0	0	0	0	0	0	0	0	0	8	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	6	0
Tiempo mínimo de búsqueda [seg]		5	1	1	4	1	1	1	1	1	1	1	1	1	3	3
		10	2	3	9	2	2	2	3	2	2	2	2	2	3	8
		15	3	4	14	3	3	3	5	4	3	3	3	1	10	13
Tiempo máximo de búsqueda [seg]		5	4	1	6	1	1	1	2	2	1	1	1	2	11	5
		10	7	3	12	2	2	2	5	3	2	2	2	3	7	11
		15	11	5	18	3	3	3	8	6	3	3	3	3	11	16
Confidencia mínima (aciertos)		5	1	3	1	1	39	56	78	39	2	30	10	5	1	78
		10	7	15	20	24	42	71	87	43	18	24	9	3	3	88
		15	7	15	20	68	126	179	243	215	62	89	29	20	1	243
Confidencia máxima (errores)		5	-	-	-	-	1	1	1	1	1	1	1	-	-	1
		10	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		15	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Días estimados para la creación de 80000 canciones			100	134	660	62	62	76	158	328	118	248	436	660		250

Tabla A5: Resumen de los resultados de Dejavu al probar 18 extractos de música.

Los extractos de 5, 10, y 15 segundos, se generan de 2 canciones aleatorias que no son parte de las bases de datos.

Salida		Segundos	A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3	E1	F1
Respuesta Dejavu	Aciertos	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Errores	5	6	6	6	6	6	6	6	6	6	6	5	5	0	6
		10	6	6	6	6	6	6	6	6	6	6	6	6	0	6
		15	6	6	6	6	6	6	6	6	6	6	6	6	0	6
Sin Respuesta		5	0	0	0	0	0	0	0	0	0	0	1	1	6	0
		10	0	0	0	0	0	0	0	0	0	0	0	0	6	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	6	0
Tiempo mínimo de búsqueda [seg]		5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
		10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
		15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Tiempo máximo de búsqueda [seg]		5	1	1	6	1	1	1	3	2	1	1	2	2	-1	6
		10	2	3	12	2	2	2	5	4	2	2	3	3	-1	11
		15	3	5	18	3	3	3	8	6	3	3	3	3	-1	17
Confidencia mínima (aciertos)		5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		10	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		15	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Confidencia máxima (errores)		5	3	3	3	4	4	4	4	4	3	2	2	1	-	4
		10	4	4	4	4	4	4	4	4	3	3	2	1	-	4
		15	4	4	6	4	4	4	4	4	3	3	3	2	-	4
Días estimados para la creación de 80000 canciones			100	134	660	62	62	76	158	328	118	248	436	660		250

Tabla A6: Resumen de los resultados de Dejavu al probar 45 extractos de música.

Las pruebas se realizaron con cada base de datos que solo contiene 5 canciones, generando extractos de 5,10, y 15 segundos para realizar el reconocimiento acústico.

Salida		Segundos	A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3	E1	F1
Con reconocimiento	Aciertos	5	14	15	15	15	14	14	14	14	12	14	14	14	3	14
		10	15	15	15	15	15	15	15	15	15	15	15	15	3	15
		15	15	15	15	15	15	15	15	15	15	15	15	15	4	15
	Errores	5	1	0	0	0	1	1	1	1	3	1	1	1	0	1
		10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sin reconocimiento		5	0	0	0	0	0	0	0	0	0	0	0	0	12	0
		10	0	0	0	0	0	0	0	0	0	0	0	0	12	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	11	0
Tiempo mínimo de búsqueda [seg]		5	1	1	4	1	1	1	1	1	3	1	1	1	3	4
		10	3	3	9	2	2	3	3	3	7	3	2	2	3	9
		15	5	5	14	3	5	5	6	6	12	5	3	3	11	14
Tiempo máximo de búsqueda [seg]		5	3	3	7	1	2	3	3	3	5	2	1	1	11	7
		10	5	5	13	2	4	5	6	5	10	5	2	2	7	13
		15	9	8	19	3	7	8	9	8	15	7	3	3	11	19
Confidencia mínima (aciertos)		5	1	2	1	1	11	15	9	3	26	4	2	1	1	19
		10	7	15	19	24	42	66	87	43	18	24	7	3	2	88
		15	7	15	20	43	69	104	143	109	62	31	8	4	1	138
Confidencia máxima (errores)		5	5	-	-	-	1	1	1	1	3	1	1	1	-	1
		10	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		15	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Días estimados para la creación de 80000 canciones			100	134	660	62	62	76	158	328	118	248	436	660		250

Tabla A7: Resumen de los resultados de Dejavu al probar 216 extractos de covers.

Los extractos de 5, 10, y 15 segundos, se generan de 24 covers presentes en las bases de datos.

Salida		Segundos	Covers30	Covers50	Covers100	Covers150	Covers200	Covers250
Con reconocimiento	Aciertos	5	61	61	60	60	60	60
		10	71	71	71	71	71	71
		15	72	72	72	72	72	72
	Errores	5	4	4	5	5	5	5
		10	0	0	0	0	0	0
		15	0	0	0	0	0	0
Sin reconocimiento		5	7	7	7	7	7	7
		10	1	1	1	1	1	1
		15	0	0	0	0	0	0
Tiempo mínimo de búsqueda		5	1	1	1	1	1	1
		10	1	1	1	1	1	1
		15	1	1	1	1	1	1
Tiempo máximo de búsqueda		5	6	6	7	7	7	7
		10	6	6	6	7	7	7
		15	6	7	7	7	7	7
Confidencia mínima (aciertos)		5	2	2	8	8	8	8
		10	7	7	7	7	7	7
		15	53	53	53	53	53	53
Confidencia máxima (errores)		5	3	3	4	4	4	4
		10	-	-	-	-	-	-
		15	-	-	-	-	-	-

Tabla A8: Resumen de los resultados de Dejavu al probar 27 extractos de covers.

Los extractos de 5, 10, y 15 segundos, se generan de 3 covers que no están presentes en las bases de datos.

Salida		Segundos	Covers10	Covers20
Con reconocimiento	Aciertos	5	0	0
		10	0	0
		15	0	0
	Errores	5	7	7
		10	9	9
		15	9	9
Sin reconocimiento		5	2	2
		10	0	0
		15	0	0
Tiempo mínimo de búsqueda		5	1	1
		10	1	1
		15	1	1
Tiempo máximo de búsqueda		5	5	7
		10	5	6
		15	3	4
Confidencia mínima (aciertos)		5	-	-
		10	-	-
		15	-	-
Confidencia máxima (errores)		5	6	6
		10	6	6
		15	6	6