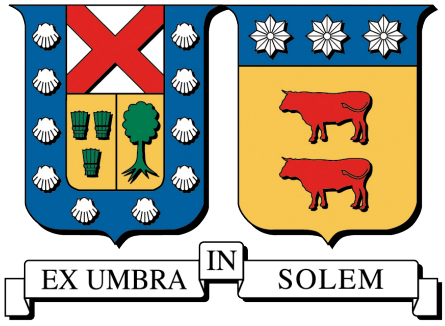
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE




# 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

XXXXXX 2017

# RESUMEN

El presente documento tiene como finalidad presentar una propuesta y una metodología para automatizar la fiscalización de los derechos de propiedad intelectual, abarcando específicamente la reproducción en radioemisoras online, con tal de detectar la frecuencia de reproducción de canciones de artistas chilenos, y llevar así, un control del uso de su música. Para ello, se diseñó una plataforma e implementó un algoritmo de reconocimiento acústico Finalmente, las pruebas realizadas demuestran tiempos de ejecución

**Palabras Clave.** Algoritmo de reconocimiento acústico. Fingerprints.

# ABSTRACT

El presente documento tiene como finalidad presentar una propuesta y una metodología para automatizar la fiscalización de los derechos de propiedad intelectual, abarcando específicamente la reproducción en radioemisoras online, con tal de detectar la frecuencia de reproducción de canciones de artistas chilenos, y llevar así, un control del uso de su música. Para ello, se diseñó una plataforma e implementó un algoritmo de reconocimiento acústico Finalmente, las pruebas realizadas demuestran tiempos de ejecución

**Keywords.** Algoritmo de reconocimiento acústico. Fingerprints.

# Índice de Contenidos

# Índice de Tablas

# Índice de Figuras

# Glosario

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

# 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 [Biblioteca Congreso Nacional de Chile. (2015). Fija porcentajes mínimos de emisión de música nacional y música de raíz folklórica oral, a la radio difusión chilena. 13 de junio, 2016, de Biblioteca Congreso Nacional de Chile Sitio web: ...] y desde entonces, las diversas radios del país se vieron en la necesidad de adecuar sus parillas de programación para cumplir con la nueva disposición.

Si bien esta propuesta fue promulgada con el fin de promover la diversidad y potenciar la industria musical chilena, se debe considerar que el único ente encargado de distribuir las ganancias a cada artista es la Sociedad Chilena de Derechos de Autor (SCD) que tiene por objetivo administrar derechos autorales generados por la utilización de obras musicales y fonogramas, vale decir, producciones musicales en cualquiera de sus formatos [http://scd.cl/ Sociedad Chilena de Derechos de Autor].

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 [Open Business. (2013). Desafíos de la gestión colectiva en Chile. 12 de junio, 2016, de Open Business. Sitio web: http://openbusinesslatinamerica.org/2013/04/22/desafios-de-la-gestion-colectiva-en-chile/] 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. 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.

Es por ello, que en este trabajo se realiza un análisis y diseño de una plataforma que permita a los artistas chilenos llevar un control del uso de su música a nivel radial para revisar simultáneamente en diversas radioemisoras online.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum,

erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

La estructura de la presente memoria es la siguiente:

- El capítulo 1 detalla la propuesta a desarrollar, las razones para hacerlo, y los objetivos que se desean cumplir con ello.
- El capítulo 2 muestra una indagación en la literatura abarcando principalmente las metodologías que existen para el reconocimiento de canciones digitalizadas, y los servicios y/o algoritmos más conocidos.
- El capítulo 3 presenta el diseño inicial, explicando la metodología para desarrollar la solución y detalla la forma en que se llevó a cabo la implementación del algoritmo de reconocimiento acústico.
- En el capítulo 4 se señalan las tareas realizadas para comprobar el rendimiento de la plataforma.
- El capítulo 5 muestra los resultados obtenidos tras llevar a cabo la etapa anterior, analizando los valores recopilados.
- Para finalizar, la Concusión resume el trabajo presentado, se comprueba el cumplimiento de los objetivos planteados, y se identifican los posibles trabajos a futuro.

# 1 | Definición del problema

## 1.1. Descripción

En Chile, la Sociedad de Derechos 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 encarga 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 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 cuenta con señal online [...]

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 [.../]. 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 [...] y cuyas características se señalan a continuación:

`http://transparenciaactiva.cultura.gob.cl/uploads/marcoNormativo/ebf456ab89212ce3af3173b8f2ebd18`
`PAg 3, las partes convienen CUATRO]`

Y cuyo costo, ascendió a un total de $290.850.000 pesos chilenos, de los cuales:

`[Transcribir: http://transparenciaactiva.cultura.gob.cl/uploads/marcoNormativo/ebf456ab89212ce3a`

`PAg 3, las partes convienen TERCERO]`

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. [... ]

Los dos elementos que contribuyen a este escenario es que, por un lado, (explicar del encargado de la SCD, estipulado por licitación). Adicionalmente, existe un gran número de artistas que no se encuentran

asociados a los registros de las SCD, tales como (buscar y nombrar artistas independientes).

Es en base a esta situación que surge la idea de diseñar y desarrollar una plataforma nacional que permita fiscalizar las más de 30 radioemisoras locales y comprobar de manera automática las parrillas musicales de cada una de éstas, permitiendo también, a los artistas chilenos no asociados a la SCD, llevar un control del uso de su música a nivel radial.

## 1.2. Objetivos

### 1.2.1. Objetivo general

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

### 1.2.2. Objetivos secundarios

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

## 1.3. Alcances

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

# 2 | Estado del arte

Dado que la principal funcionalidad a desarrollar para esta memoria decae en el reconocimiento y tratamiento de archivos de audio, para su comparación y análisis, se estudió en primera instancia sobre el modo de operación de algunas aplicaciones móviles conocidas que permiten reconocer audio a partir de comparaciones con el catálogo de canciones de su base de datos.

Este análisis permitió comprender que dichas aplicaciones 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 en algunos casos, un tercer eje representa la intensidad del sonido.

## 2.1. Fiscalización de derechos de propiedad intelectual

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

### 2.1.1. Content ID

Sistema que escanea vídeos de YouTube y, si encuentra contenido con derechos música, imagen, etcétera notifica tanto al usuario como al poseedor legal del contenido. Actually licensed from a company called Audible Magic: http://www.audiblemagic.com/ [7]

### 2.1.2. Monitec

Comercialización de herramientas tecnológicas para la captura, reconocimiento y monitoreo de medios electrónicos y acústicos. Utiliza huellas o patrones digitales de las canciones y anuncios publicitarios que se quieren detectar y las utiliza como base para recorrer, en tiempo real, la señal de radio o televisión que captan sus antenas.

El sistema de monitoreo musical y publicitario de Monitec ofrece un software diseñado especialmente para uso de la industria discográfica, industria publicitaria, medios de comunicación y sociedades de gestión colectiva. Monitoreo de Publicidad: Control de Pauta detallado, con el conteo y el detalle exacto de las apariciones de su cuña o spot publicitario en cada medio, ya sea este radio, televisión abierta o televisión por cable.

### 2.1.3. Vericast

Vericast es un servicio de BMAT (España) global de identificación de música que monitoriza millones de canciones en aproximadamente 3000 radios y televisiones de más de 60 países.

La solución ofrece una identificación a tiempo real y reportes auditables basados en la huella digital o fingerprint del audio que es resistente a alteraciones de la señal como en un canal degradado o ruidoso, cuando la música se utilice de fondo. Bmat opera globalmente con más de 50 clientes en Europa [9], América y Asia. En 2011 incrementó sustancialmente sus ingresos respecto a 2010 y alcanzó los 1,1 millones de euros. Este año tienen previsto cerrar con 1,6 millones. Actualmente comercializa tres líneas de productos en una tecnología protegida por varias patentes internacionales.

## 2.2. Algoritmos de reconocimiento acústico

La principal metodología para la recuperación y reconocimiento de audio se basa en la huella digital acústica extraída de la canción en cuestión, la cual puede consistir en una muestra uniforme, o un resumen de los puntos de interés del espectrograma. Posteriormente, la información es comparada con una base de datos que consta de un catálogo de pistas de referencia para encontrar los mejores candidatos coincidentes. Sin embargo, para evitar hacer una comparación con todas las pistas de la base de datos, ésta puede ser particionada por Hash Directo, Locality Sensitive Hashing, u otras técnicas que se basan en la cuantización de vectores utilizados principalmente en datos comprimidos. De esta forma, es posible realizar una búsqueda más exacta en base a la vecindad de la solución [K. Mikolajczyk and C. Schmid. Performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1615 1630, 2005.]. Finalmente se realiza un último paso que permite tratar las coincidencias entregadas por la consulta de tal forma de eliminar de la respuesta los falsos positivos.

### 2.2.1. Paper 1

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit.

Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

## 2.3. Sistemas de reconocimiento acústico

### 2.3.1. Echoprint

Open source music identification system that allows anyone to build music fingerprinting into their application. It is powered by The Echo Nest.

### 2.3.2. ACRCloud

El servicio de reconocimiento de música ACRCloud ofrece acceso a una de las bases de datos de huellas de audio digitales más grande del mundo con uno de los mejores índices de reconocimiento de la indústria. Además, hemos integrado tecnologías de otros servicios musicales como Spotify, Deezer, iTunes, etc en nuestros resultados de reconocimiento, permitiendo a los desarrolladores proporcionar enlaces directos de estos servicios a sus usuarios, por ejemplo, para reproducir o comprar canciones inmediatamente.

### 2.3.3. Echonest

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

### 2.3.4. Dejavu project

Audio fingerprinting and recognition algorithm implemented in Python.

Dejavu can memorize audio by listening to it once and fingerprinting it. Then by playing a song and recording microphone input, Dejavu attempts to match the audio against the fingerprints held in the database, returning the song being played

### 2.3.5. Chromaprint

Chromaprint is the core component of the AcoustID project. It's a client-side library that implements a custom algorithm for extracting fingerprints from any audio source.

Note that the library only calculates audio fingerprints from the provided raw uncompressed audio data. It does not deal with audio file formats in any way. Your application needs to find a way to decode audio files (MP3, MP4, FLAC, etc.) and feed the uncompressed data to Chromaprint.

# 3 | Diseño de la solución

## 3.1. Metodología

Para desarrollar la plataforma y cumplir con los objetivos planteados en 1.2.1, 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

En base a la investigación realizada en el apartado **??** se establecieron XXXX aspectos para comparar los sistemas analizados, y determinar así, cual se ajusta mejor a los requerimientos del proyecto. Por ejemplo, una de las exigencias mas relevantes de la plataforma, según los objetivos secundarios planteados en 1.2.2, 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 **??** 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:** My caption

| Elemento | Dejavu | bmoquist/ Shazam | bfirsh/ python-echoprint | Music Identification based |
|---|---|---|---|---|
| Lenguaje | Python | Python | Python | c++ |
| Base de datos | Local | | Lista en el programa | |
| Entrada de datos | Archivo y micrófono | | Lista | chroma files of the songs |
| Última actualización | Abr-15 | Ene-14 | Nov-12 | Jul-13 |
| Sistema | Unix-Fedora | | | |

### 3.2.1. Descripción de Dejavu Project

Como se ha señalado en secciones anteriores, el proyecto Dejavu es un sistema open-source desarrollado en Python, que basa su implementación en fingerprints. Sin embargo, antes de abordar su funcionamiento y especificaciones, 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 primer concepto a interiorizar es la aplicación de la Transformada Rápida de Fourier (FFT de sus siglas en inglés: Fast Fourier transform) para el procesamiento digital de señales. Esto debido a que la música es codificada digitalmente como una larga lista numérica. En el caso de un archivo .wav la cantidad de números por canal es del orden de 44100 por segundo, donde los canales corresponden a la secuencia de muestras que un parlante de música puede tocar. Los más habituales son estéreo (2 canales) o mono (un canal). Por tanto, si se contara con una canción promedio de 3 minutos 30 segundos, la cantidad de muestras alcanzaría los:

$$210[s] * 44100 \left[\frac{muestras}{s}\right] * 2\,[canales] = 18{,}522{,}000$$

Por su parte, el valor 44100 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 Dejavu. Basta con señalar que este teorema matemático establece que existe un límite en la frecuencia máxima que es posible capturar con precisión durante la grabación de las señales de audio, y esta frecuencia indica que tan rápido muestreamos la señal, además de establecer que para asegurarse de tomar las muestras necesarias, es requerido aumentar al doble la frecuencia máxima.

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 44.100 muestras por segundo:

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

Cabe destacar que en el caso de los archivos .mp3, este formato comprime estos números con la finalidad de ahorrar espacio en el disco.

Una vez se aplica la transforma 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 3.1(b), al aplicar la FFT 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, Dejavu utiliza las herramientas de la biblioteca scipy para tratar el audio de entrada e identificar los máximos locales, tal como lo señala la imagen 3.2.b . De este modo, se extrae toda

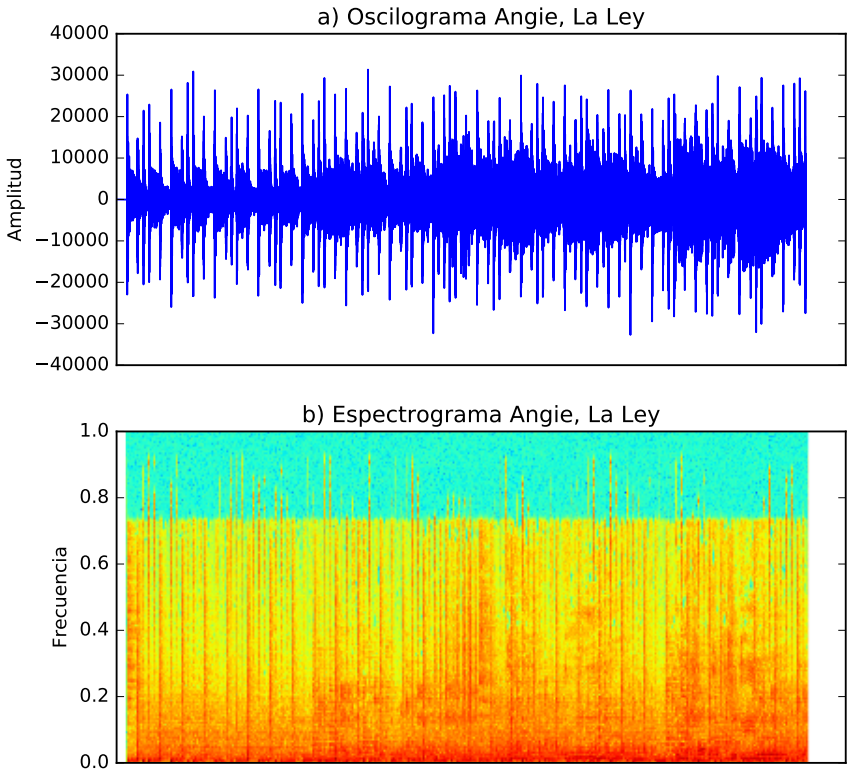



**Figura 3.1:** Modos de representación visual de un archivo de música.
La figura (a) corresponde a un oscilograma de los primeros 30 segundos de la canción Angie de La Ley que señala la variación de la amplitud respecto al tiempo. La imagen (b) es la representación gráfica de la misma canción, llamada espectrograma, donde la variable dependiente es la frecuencia, en función del tiempo.

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.

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 peak aledaño dentro del vecindario, como se observa en la figura 3.3. Específicamente, Dejavu [revisar código fuente]. 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.

Una vez comprendido los conceptos básicos que permiten caracterizar a cada una de las canciones, creando perfiles únicos para su identificación, es posible abordar el funcionamiento de Dejavu. En primer lugar, se debe contar con un sistema que permita almacenar los diversos fingerprints en una base de datos, puesto que ésta será utilizada 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

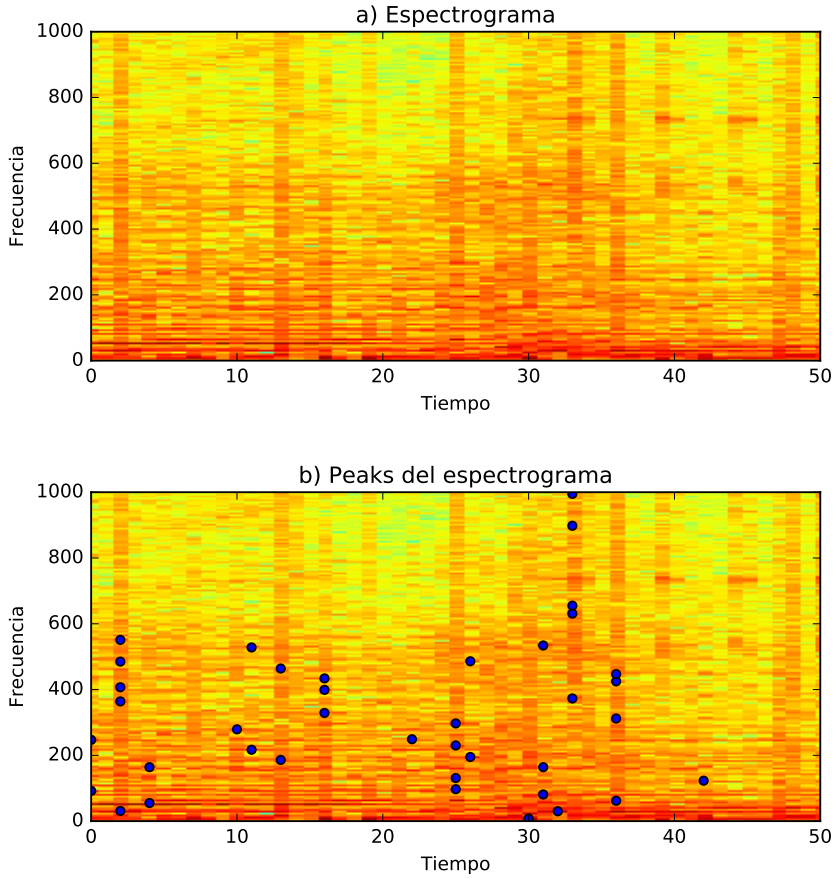



**Figura 3.2:** Espectrograma con sus peaks de audio.
La imagen (a) corresponde a una porción aumentada del espectrograma de los primeros 30 segundos de la canción Angie de La Ley. En (b) se observa el mismo espectrograma con los peaks de audio, cuyos valores de tiempo-frecuencia son utilizados para la creación de fingerprints.

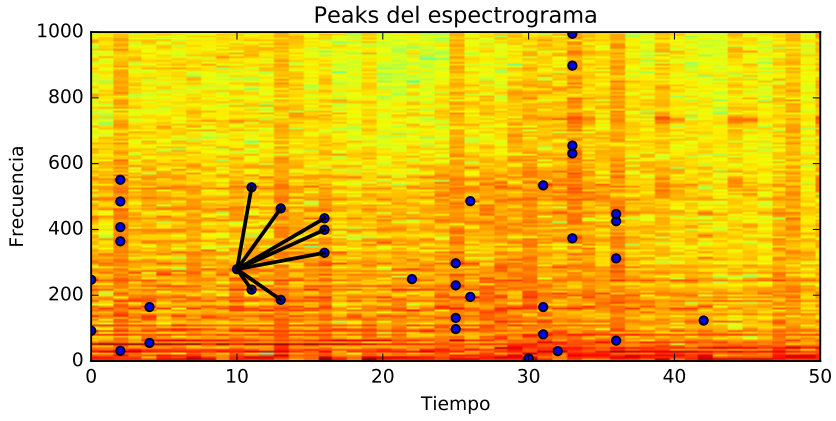



**Figura 3.3:** Espectrograma con sus peaks de audio.
La imagen (a) corresponde a una porción aumentada del espectrograma de los primeros 30 segundos de la canción Angie de La Ley. En (b) se observa el mismo espectrograma con los peaks de audio, cuyos valores de tiempo-frecuencia son utilizados para la creación de fingerprints.

y songs, con una clave foránea **song_id** autoincremental que las relaciona.

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

| Tabla | Dato | Tipo |
|---|---|---|
| songs | song_id | mediumint unsigned |
| | song_name | varchar(250) |
| fingerprints | fingerprinted | tinyint |
| | hash | binary(10) |
| | song_id | mediumint unsigned |
| | offset | int unsigned |

Teniendo en consideración que se crean muchos fingerprints por canción, el campo hash es un binary(10) con tal de reducir el espacio en la base de datos.

...

Se deben mencionar, además, los archivos .py que componen a Dejavu project. **_init_.py** contiene las defincciones 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 de el 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 arhivos 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 las 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**.

Y 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.

#### 3.2.1.1. 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 CONFIDENCIA 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 **??**.

### 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 7.2 se comparan las bases de datos creadas, observándose su tamaño, además de tiempos de creación. Se ha elaborado el gráfico de la figura 3.4, 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 **?**.

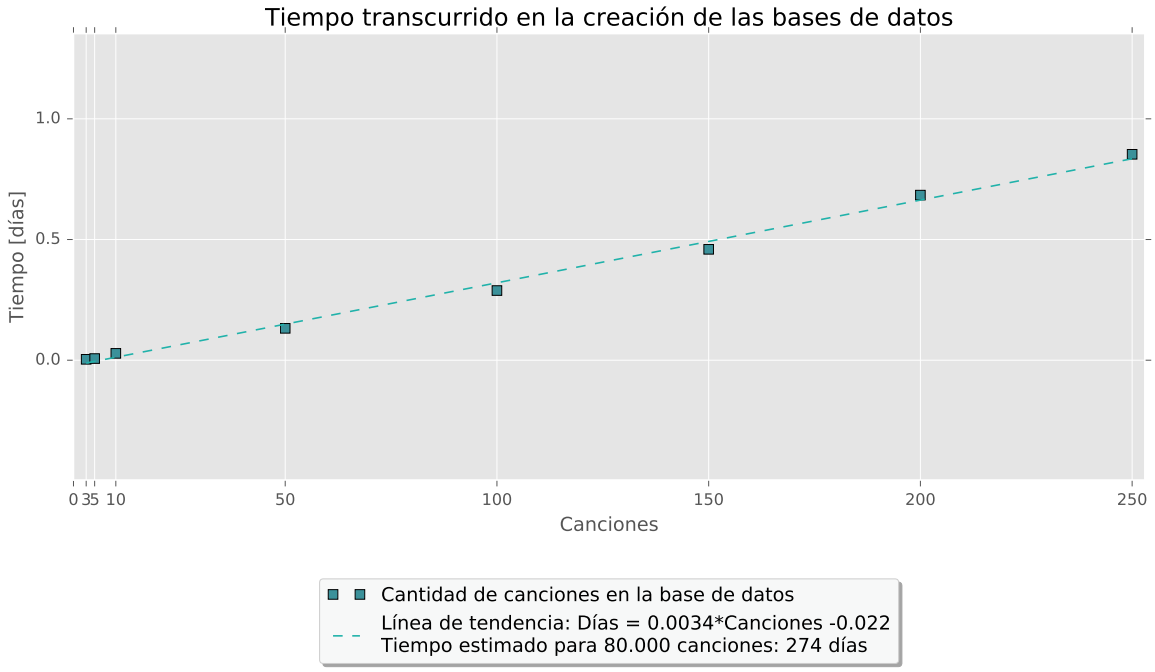



**Figura 3.4:** Tiempo en función del número de canciones con la configuración original de Dejavu.
La ecuación de la línea de tendencia, permite extrapolar el valor de los días necesarios para crear la base de datos de 80.000 canciones de la plataforma.

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: Análisis de atributos de configuración**

Según lo descrito en 3.2.1, 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 analizado influye en el número de fingerprints creados.

La tabla 7.4 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 3.5 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 fingerpirnts 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: determina el porcentaje de veracidad o certeza de la respuesta entregada por el algoritmo.
- Acierto: esta variable determina efectivamente si la salida del algoritmo corresponde al extracto de canción analizado.
- Tamaño del extracto: elemento 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 7.5 del apéndice, que en conjunto con los gráficos de la figura 6.1, permiten extrapolar el tiempo necesario para la plataforma de 80.000 canciones.

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 respuestas 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 extarcto de 15 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 de reconocimiento acústico.

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 sus reducidos tiempos

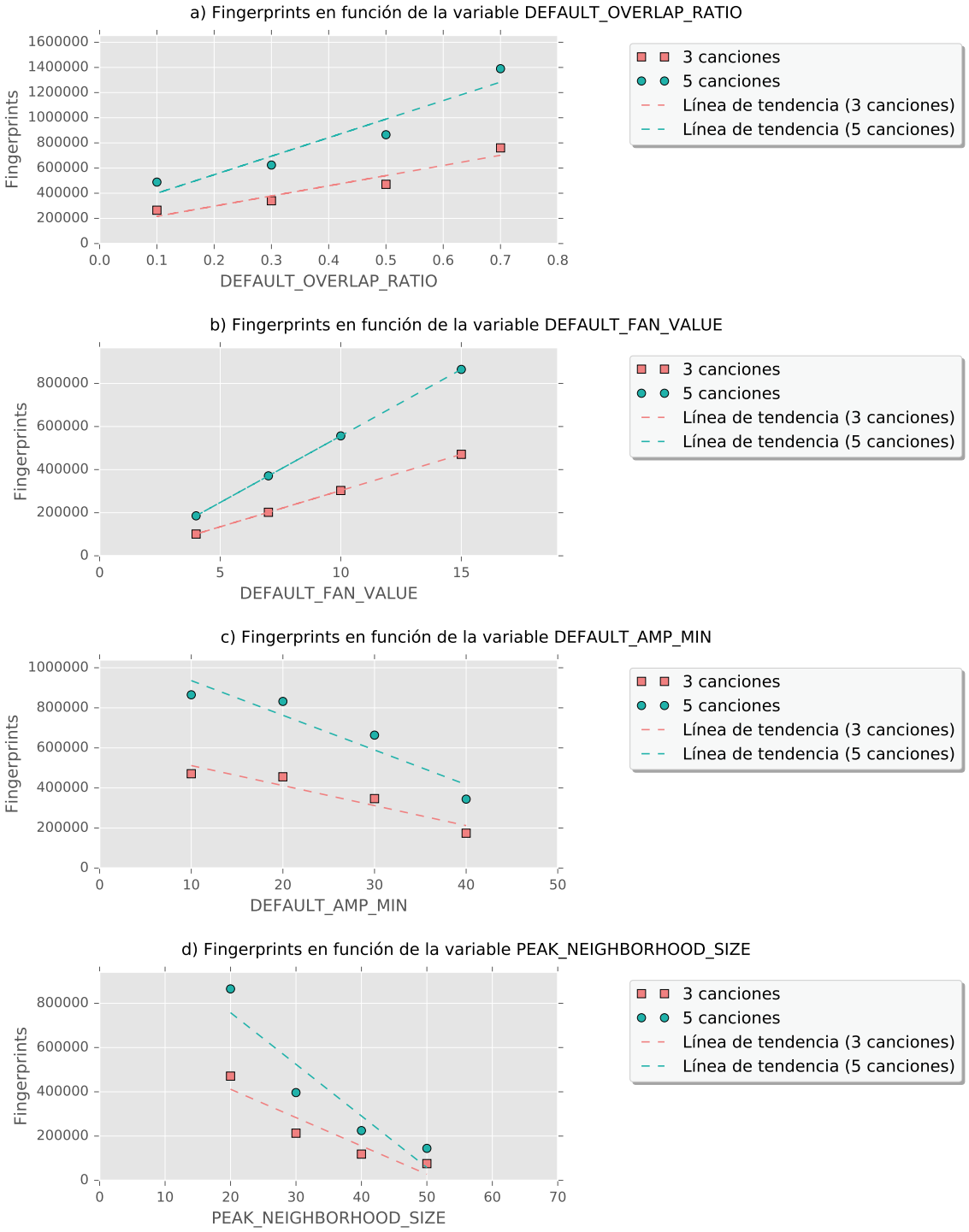



**Figura 3.5:** Fingerprints en función de diversos valores de configuración.
Los gráficos muestran la variación de la cantidad de fingerprints en función de los valores asignados a las variables de configuración: DEFAULT_OVERLAP_RATIO y DEFAULT_FAN_VALUE como rectas crecientes, PEAK_NEIGHBORHOOD_SIZE y DEFAULT_AMP_MIN como rectas decrecientes.

estimados en días, para la creación de la base de datos. Además presentan un buen rendimiento, al revisar el número de aciertos por el total de pruebas realizadas, siendo solo 1 de 9, de los extractos de 5 segundos el

que obtuvo una respuesta incorrecta.

### C: Prueba de extractos de música inexistentes 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 7.6 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, pues las canciones no existían en las base de datos analizadas.

Para ahondar en el estudio de este comportamiento, se analiza la fila de confianza máxima de las respuesta 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 Dejavu 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 7.5 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 7.6 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 3.3

**Tabla 3.3:** 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 |

### 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 otro músicos,

creando nuevas versiones de una obra musical.

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.

Resultados...

## 3.3. Descripción del programa

### 3.3.1. 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.

> Agregar requerimientos: conexión a internet. URL streaming.

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 que contiene el listado del nombre de las canciones, con sus respectivos tiempos cronológicos de emisión, el cual es creado a partir del método *craerArchivoSalida(nombreRadio)*.

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*. *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 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 6 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.

### 3.3.2. 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 agregas al desarrollo anterior.

**Algoritmo 3** Fiscalización paralela

**Entrada:** *archivoEntrada, segundosExtracto, fechaLimite, delay.*

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 **??**.

**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 **??**, 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 2 *generarExtracto* ya definido.

A su vez, el método *fiscalizar* se llama a si mismo una vez a pasado 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

## 4.1. Características del equipo

## 4.2. Implementación

# 5 | Resultados

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in,

fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

# Conclusiones

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

# 6 | Figuras

## 6.1. 1

Mira mira 6.1

Mira mira 6.2

Mira mira 3.4

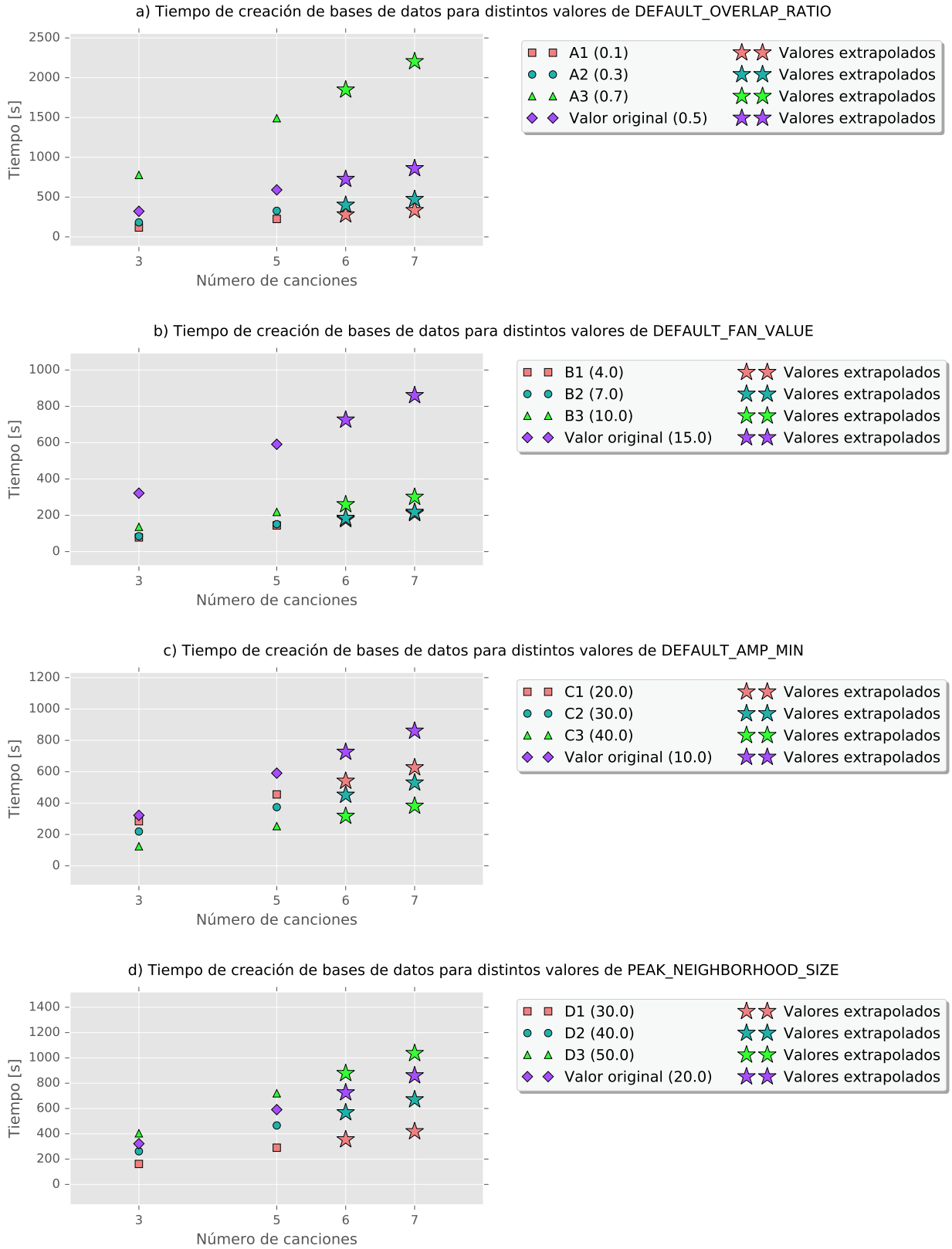



**Figura 6.1:** Tiempo de creación de bases de datos al modificar valores de configuración.
Tiempo transcurrido en la creación de las bases de datos de tres y cinco canciones, con dos valores extrapolados para cada grupo de variables, utilizados para estimar cuanto tardaría una base de datos de 80.000 canciones.

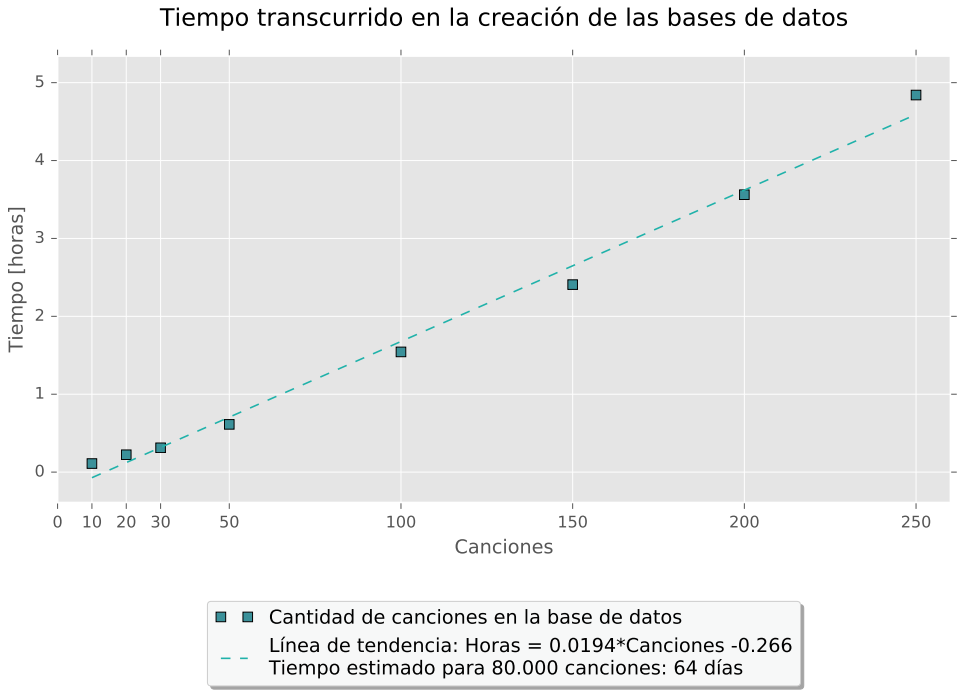



**Figura 6.2:** Tiempo en función del número de canciones de la nueva configuración de Dejavu.
Tiempo transcurrido en la creación de las bases de datos con la nueva configuración de Dejavu. La ecuación de la línea de tendencia, permite extrapolar el valor de los días necesarios para crear la base de datos de 80.000 canciones de la plataforma.

# 7 | Tablas

## 7.1. 1

**Tabla 7.1:** 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 7.2:** 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 7.3:** Tiempo de creación de bases de datos, con los valores de configuración.

| Número de canciones | Tiempo [seg] | Tiempo [min] | Tiempo [hr] |
|---|---|---|---|
| 10 | 396 | 6,6 | 0,1 |
| 20 | 802 | 13,4 | 0,2 |
| 30 | 1124 | 18,7 | 0,3 |
| 50 | 2206 | 36,8 | 0,6 |
| 100 | 5555 | 92,6 | 1,5 |
| 150 | 8666 | 144,4 | 2,4 |
| 200 | 12821 | 213,7 | 3,6 |
| 250 | 17428 | 290,5 | 4,8 |

**Tabla 7.4:** 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 7.5:** 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 7.6:** 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 7.7:** 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 7.8:** 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 7.9:** 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 |