

UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Desarrollo de Formato de Data para Generación Procedural  
de Pistas Musicales**

Trabajo de graduación presentado por Diego Javier Álvarez Cruz para  
optar al grado académico de Licenciado en Ingeniería en Ciencias de la  
Computación y Tecnologías de la Información

Guatemala,

2022







UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Desarrollo de Formato de Data para Generación Procedural  
de Pistas Musicales**

Trabajo de graduación presentado por Diego Javier Álvarez Cruz para  
optar al grado académico de Licenciado en Ingeniería en Ciencias de la  
Computación y Tecnologías de la Información

Guatemala,

2022



Vo.Bo.:



(f)

\_\_\_\_\_  
Msc. Rodolfo Rodas

Tribunal Examinador:



(f)

\_\_\_\_\_  
Msc. Rodolfo Rodas



(f)

\_\_\_\_\_  
Msc. Douglas Leonel Barrios



(f)

\_\_\_\_\_  
Lic. Pablo Estrada

Fecha de aprobación: Guatemala, 9 de diciembre de 2022.





El presente trabajo nace de la curiosidad sobre la música y de la creciente necesidad de sistemas de datos robustos con facilidad de uso.

Me gustaría agradecer a mis padres, Carlos y Lorena, por el apoyo que me brindaron, y la paciencia que tuvieron durante mi camino universitario, ya que sin ellos estoy seguro que no habría llegado tan lejos en mi vida profesional, a Evelin Palencia, quien al día de hoy es la persona que más apoyo me ha brindado, y gracias a su paciencia, consejos y regaños me ha ayudado a formar la persona que soy hoy, y empujado a siempre hacer todo un poco mejor cada día. A mi grupo de amigos y de trabajo en el mundo de la ingeniería de datos, incluyendo a mi asesor Rodolfo Rodas, Erick Navarro, José Manuel Cortéz y Erwin Alarcón, quienes han apoyado de forma directa e indirecta mi rendimiento universitario, y quienes me inspiran a ser un mejor profesional cada día de trabajo. Envío un especial agradecimiento a Jorge Pérez, ya que el apoyó mucho en la realización de este trabajo, y junto a el tuvimos la idea de hacer un sistema de este tipo.

Agradezco también al ingeniero Douglas Barrios por todo el seguimiento y consejos, y por guiarme a través de toda mi carrera. De igual forma, al doctor Ismael Salazar por prestar su apoyo para mis mejoras personales, con el cual me ayudó a salir de situaciones muy difíciles.

Finalmente, agradecimientos por razones variadas Rodrigo Chávez, Kole Williams, Jac Sullivan, Thomas Gove, Adam, Nicholas G, Carlos W, Michael M, mi grupo de amigos y compañeros de trabajo de primer año y al grupo con quien tuve el privilegio de compartir este último semestre, y a todos los demás que no mencionaré para no extenderme, pero que saben la significancia que han tenido sobre mi carrera profesional.



<b>Prefacio</b>	<b>v</b>
<b>Lista de figuras</b>	<b>x</b>
<b>Lista de cuadros</b>	<b>xi</b>
<b>Resumen</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Antecedentes</b>	<b>3</b>
<b>3. Justificación</b>	<b>5</b>
<b>4. Objetivos</b>	<b>7</b>
4.1. Objetivo general . . . . .	7
4.2. Objetivos específicos . . . . .	7
<b>5. Marco teórico</b>	<b>9</b>
5.1. Generación procedural . . . . .	9
5.2. Teoría musical . . . . .	10
5.3. Ingeniería de datos y programación <i>data driven</i> . . . . .	12
<b>6. Marco metodológico</b>	<b>15</b>
6.1. Sistema de ingestión de datos . . . . .	15
6.1.1. Módulo de <i>pipelines</i> . . . . .	16
6.1.2. Módulo de obtención de metadata de LastFM . . . . .	16
6.1.3. Extracción de Ultimate Guitar y Music21 . . . . .	17
6.2. Diseño de base de datos . . . . .	18
6.2.1. Fase de <i>staging</i> y procedimientos almacenados . . . . .	19

6.2.2. Fase de almacenaje analítico . . . . .	20
6.3. Sistema orquestador . . . . .	21
6.4. Análisis adiconal . . . . .	22
<b>7. Resultados</b>	<b>23</b>
7.1. Formato de salida . . . . .	24
7.2. Encuestas de importancia y probabilidad . . . . .	26
7.3. Análisis de complejidad y costos . . . . .	27
<b>8. Análisis de resultados</b>	<b>29</b>
<b>9. Conclusiones</b>	<b>33</b>
<b>10.Recomendaciones</b>	<b>35</b>
<b>11.Bibliografía</b>	<b>37</b>
<b>12.Anexos</b>	<b>39</b>

1.	Explicación del GAN [7] . . . . .	9
2.	Ejemplo de un <i>height map</i> generado proceduralmente [7] . . . . .	10
3.	Flujo común de un ETL. [14] . . . . .	12
4.	Elementos incluidos generalmente en una arquitectura de <i>data warehouse</i> o <i>data lake</i> . [16] . . . . .	13
5.	Diagrama de flujo de módulo principal . . . . .	16
6.	Diagrama de flujo de módulo de LastFM . . . . .	17
7.	Diagrama de flujo de módulo de Ultimate Guitar y Music21 . . . . .	18
8.	Diagrama de vista general del sistema ELT . . . . .	19
9.	Diagrama entidad relación de fase <i>staging</i> . . . . .	20
10.	Diagrama entidad relación de repositorio final . . . . .	21
11.	Tabla de cantidad de composiciones por artista, para etiqueta <i>mathcore</i>	23
12.	Tabla de cantidad de artistas por etiqueta, para las etiquetas que se usaron como prueba . . . . .	23
13.	Tabla de ejemplo de lo que se obtiene para una canción, en este caso <i>One of Us Is The Killer</i> de The Dillinger Escape Plan . . . . .	24
14.	JSON con los datos crudos para una composición de <i>mathcore</i> , en este caso <i>One of Us Is The Killer</i> de The Dillinger Escape Plan . . . . .	25
15.	Promedio de notas utilizadas en una composición de <i>mathcore</i> , según las composiciones analizadas . . . . .	25
16.	Promedio de acordes utilizados en una composición de <i>mathcore</i> , según las composiciones analizadas . . . . .	26
17.	Importancia percibida de cada elemento del pentagrama al momento de escribir una composición musical, donde 0 es nada importante y 4 es casi esencial. . . . .	26
18.	Probabilidad de utilizar cada elemento del pentagrama para intentar categorizar una composición por género, humor o descriptor general . . . . .	27
19.	Costos mensuales del servicio de base de datos prestado por Microsoft Azure . . . . .	28
20.	Distribución de costos del servicio prestado por Microsoft Azure . . . . .	28

21.	Pronóstico de costos de mantenimiento de Microsoft Azure . . . . .	28
22.	Encuesta de importancia de elementos musicales, pregunta 1 . . . . .	39
23.	Encuesta de importancia de elementos musicales, pregunta 2 . . . . .	40

---

Lista de cuadros

---

1. Intervalos para algunos modos de escala. Los numeros indican cuantas notas de distancia cromática hay entre la nota actual y la siguiente. [4] 11
2. Tabla de complejidad para cada una de las funciones de los módulos de Python . . . . . 27





En la teoría musical, existe el concepto de los patrones, que pueden ser entendidos como el vocabulario utilizado para leer, escribir, y de cierta forma, hablar a través de la música. Los patrones en la música también permiten a un alto nivel, una capacidad de predicción, y como mínimo una capacidad de complementar o continuar una barra del pentagrama específica.

Dentro de las capacidades de la computación, en tiempos relativamente recientes se ha logrado la funcionalidad de predecir comportamientos, muchas veces con un nivel de exactitud bastante alto, a través de patrones que bajo circunstancias normales no son visibles al análisis de un humano. Estos patrones se detectan sobre repositorios de datos cuyo tamaño no solo añade complejidad a la exploración de los datos, sino también a la detección de dichos patrones. Sin embargo, muchas veces estos datos no están disponibles de una forma ordenada, centralizada y con acceso fácil para algoritmos; esto presenta el desafío principal de un *Data Engineer*, cuya tarea es preparar un sistema que permita obtener estos datos, ejecutar un proceso de limpieza sobre ellos, y colocarlos en un repositorio centralizado, ya sea en forma de *Data Lake* o *Data Warehouse*, o algún otro parecido, para que puedan ser utilizados por procesos de inteligencia o análisis.

Considerando lo anterior, este trabajo profesional explora la posibilidad de crear un repositorio centralizado, con capacidad de generar un formato accesible, de datos musicales obtenidos de fuentes MIDI, para facilitar el análisis y determinación de patrones que comparten varias composiciones que se puedan clasificar bajo un conjunto de categorías distintas.



Within musical theory, the concept of patterns can be understood as the vocabulary used to read, write and to some extent, speak through music. At a high skill level, patterns allow a musician to have a capability of prediction, and at the very least the skill to complement or continue a specific bar in a musical staff without any additional clues.

Within the set of computational capabilities, along the last 20 years, the functionality of predicting behaviors has been made possible, at times with a very high level of precision, all through the usage of patterns that a person wouldn't have been able to detect in normal circumstances. These patterns have been detected through the use of data stores whose size does not only add complexity to the exploration of data, but also the detection of any patterns within it. That said, in many cases the data is not readily available in an ordered and centralized manner, which presents the main challenge of a Data Engineer, whose task is to prepare systems that obtain and prepare data which is consequently stored in a centralized repository, lately in the form of a Data Lake or Data Warehouse, or any sort of storage that serves the same purpose so that it is then usable by analytics, intelligence or data science.

With that in mind, this work explores the possibility of creating a centralized data repository that can, in an accessible format, store musical data obtained from MIDI sources to facilitate the analysis and detection of patterns shared by many musical compositions that all fall within a set of categories or descriptors.



MIT define generación procedural como un concepto bastante simple: Generación de datos por computadoras. Lo describen como una herramienta que permite producción de cadenas de datos posiblemente complejas, con un esfuerzo mínimo de parte del usuario. Estas cadenas de datos comprenden muchas áreas desde algo tan sencillo como texto, hasta objetos complejos como el foco central de este trabajo, música, y son usualmente utilizadas, pero no limitadas, para la creación de *assets* para videojuegos. Sin embargo, a pesar de que la utilización de la herramienta suele ser fácil y de poco esfuerzo, la producción de un algoritmo que genere estas cadenas de forma adecuada comprende dos desafíos principales: la obtención de datos que funcionen como una base de trabajo, y un nivel de entendimiento de los patrones existentes en estos datos, que depende muchas veces de una base de trabajo ordenada y fácil de acceder, ya sea por humanos o por un algoritmo.

Este trabajo, por lo tanto, se enfoca en el primer reto de crear un repositorio centralizado de datos y *metadata* de una colección de composiciones musicales que se obtengan a partir de archivos en formato MIDI (*Musical Instrument Digital Interface*), y presentarlo en forma de *Data Lake*, con la intención de permitir su obtención y análisis, acompañado de extractos pequeños de dichos MIDI.

Para ello, se emplea una plataforma ELT (*Extract, Load, Transform*) programada en Python 3.6 y procedimientos almacenados SQL, alojada en un sistema de base de datos Azure SQL combinada con Azure Blob Storage para extender las capacidades de esta y poder alojar información no estructurada. Adicionalmente, se apoya el funcionamiento de la plataforma para que pueda funcionar de forma autónoma, con una máquina virtual y procesos calendarizados a través de cron.

El ELT de python emplea la biblioteca music21 creada por Michael Cuthbert, para determinar los datos musicales de las composiciones.

El resultado final de la práctica se observa como un archivo JSON que contiene

los datos informativos de cada composición que cumple los requisitos del usuario, junto con los datos musicales, así como un histograma de notas y acordes, y la clave y modal que esta presenta.

En 2020, Tshepo [1] realiza un estudio sobre la clasificación musical por género utilizando modelos probabilísticos y *deep learning*; este estudio determinó que características como *chroma*, de la progresión de acordes no representan un peso en la determinación de género. Por otro lado, el histograma rítmico de la música sí tiene cierto peso. Sin embargo, este estudio se realizó utilizando el dataset GTZAN [2], el cual trabaja con datos musicales obtenidos de música en formato crudo, y presenta datos en forma de desviaciones y promedios. El presente trabajo busca obtener un repositorio similar al de GTZAN, pero con la intención de proveer a los interesados con datos más básicos, de forma que ellos puedan determinar lo que necesiten.

En 2021, Sarmiento [3] realiza un estudio similar al presente con DadaGP, donde explora las posibilidades de generar un repositorio con data de archivos Guitar Pro; Al igual que en este trabajo el enfoque se da solamente sobre las guitarras, pero trabajan directamente sobre archivos GP. Por otro lado, el enfoque no es de ingeniería de datos y se busca información más procesada de la que se busca en este trabajo.





Christer Kaitila [4] describe que la música en un juego tiene un efecto profundo en los jugadores; indica que es el pulido extra que queda en los recuerdos de los jugadores, inspira la imaginación y pone el ánimo de la situación. Un detalle importante que menciona es que, aún sin saber teoría musical, es posible generar música descrita como “sorprendentemente buena” con métodos procedurales siempre que se apliquen patrones, reglas y límites definidos según la temática que se quiera obtener.

Ann Waweru [5] describe el reconocimiento de patrones como la utilización de machine learning para identificar patrones, clasificándolos en base a su información estadística o conocimiento obtenido de estos y sus representaciones. Indica, además, que tiene una alta precisión de reconocimiento, con posibilidad de reconocer objetos no familiares y de varios ángulos distintos, que puede descubrir patrones parcialmente escondidos, y que incluso puede recuperar patrones en instancias de data faltante.

Regresando al artículo de Kaitila, explica que las reglas que nos provee son subjetivas y sobre simplificadas, y que hay información más avanzada que se puede observar. Siendo que machine learning es una parte esencial de la generación procedural, y que las reglas descritas por Kaitila son patrones, estudiados por *machine learning*, es natural pensar que existe una posibilidad de obtener reglas y patrones escondidos, no familiares, o poco conocidos en música utilizando un sistema de machine learning sobre un conjunto de datos de esta.

OpenAI [6] en 2019 publicó su herramienta MuseNet, una red neuronal que genera hasta 4 minutos de composiciones musicales con hasta 10 instrumentos, combinando estilos de varios compositores y artistas a lo largo de la historia. Indican que MuseNet no está programado explícitamente con conocimiento musical, sino con descubrimiento de patrones de armonía, ritmo y estilo aprendiendo a predecir el siguiente token utilizando más de mil archivos MIDI, con una tecnología de ML no supervisado conocido como GPT-2.

Una de las grandes limitaciones de MuseNet es que sus fuentes, y por lo tanto sus salidas, se limitan a archivos en formato MIDI. Aunque este proyecto no pretende desarrollar un sistema que pueda obtener directamente data de otros formatos, pretende desarrollar un sistema ELT genérico y escalable que permita a terceras personas desarrollar plug-ins que sí permitan esto. Finalmente, el propósito del proyecto es proveer un formato genérico que permita a usuarios con intenciones parecidas a OpenAI, obtener información digital de música de una manera accesible y fácil de utilizar.

#### 4.1. Objetivo general

Desarrollar un sistema ELT escalable que permita la obtención de un formato digital de música de una fuente, que a su vez permita proponer un formato genérico de información musical que contenga dos características musicales útiles para generación procedural.

#### 4.2. Objetivos específicos

- Determinar el formato digital musical más accesible para la obtención de datos.
- Seleccionar un repositorio web de música digital del cual se obtendrá el formato.
- Proponer la estructura del formato genérico que se adapte a la fuente utilizada.



## 5.1. Generación procedural

La generación procedural es definida por MIT[7] como dos palabras para un concepto simple: creación de datos por computadoras. Usualmente se utiliza para crear contenido en videojuegos o películas animadas, el cual incluye, pero no está limitado a diseños, objetos 3d, diálogo o animaciones. Una de las ventajas descritas para la generación procedural es el esfuerzo pequeño de parte de un humano en relación a la cantidad de contenido que se puede generar.

Un ejemplo general de generación procedural es un GAN, *Generative Adversarial Network*, definido por MIT como una red de generadores, que crean imágenes falsas, y una red discriminadora, que discrimina entre reales y falsas. Se hace una analogía de billetes falsos y un policía, donde la red discriminadora (el policía) cada iteración se vuelve mejor para detectar falsos, provocando que el fabricante de billetes falsos (la red generadora) debe generar un producto tan real que no se puede discriminar entre real y falso.

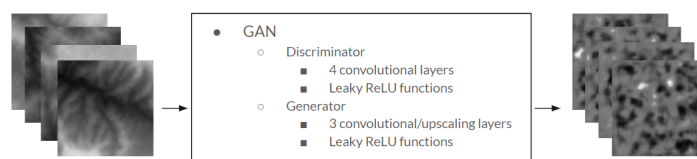


Figura 1: Explicación del GAN [7]

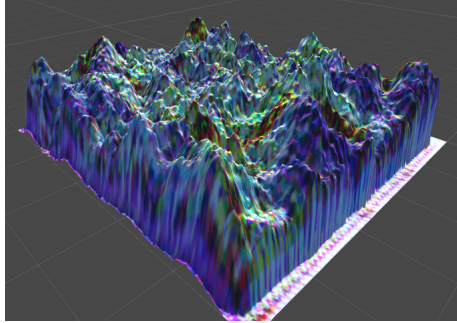


Figura 2: Ejemplo de un *height map* generado proceduralmente [7]

Uno de los requisitos más importantes para incluso iniciar a desarrollar un algoritmo de generación procedural, ya sea con Deep Learning o algo tan simple como cadenas de Markov es la recolección de data. El obtener la data a utilizar en sí puede representar un reto de dificultad variable, dependiendo de la disponibilidad de la data en records públicos o internos de un mismo proyecto. Sin embargo, el reto más grande es la preparación de la data para que, sin importar de donde se obtenga o el formato con el que se haya obtenido, este termine en un formato estándar que pueda ser utilizado fácilmente por el algoritmo de generación. [8]

Un uso interesante de la generación procedural existe en la música; existen casos, así como Steve Reich [9], quien usa técnicas de música algorítmica para generar música ambiental. En otros casos, aunque no es completamente procedural, algunas compañías de videojuegos como Bethesda Softworks, han utilizado algoritmos para selección de música según el ambiente y área en la que el jugador se encuentra en un dado momento. Otras compañías como Intelligent Systems han utilizado técnicas similares para modificar el tono e intensidad de una misma pista musical dependiendo de la situación del jugador. Sin embargo, uno de los mayores retos en cuanto a la música y la generación procedural viene en la forma del formato de los datos.

## 5.2. Teoría musical

Cuthbert [10] define la música notada como una colección de notas puestas una detrás de otra o simultáneamente en un penttograma; define las notas como el corazón de la música, a pesar de la existencia de otros elementos esenciales como la clave y la marca de tiempo. Define que, para tomar un acercamiento en forma de programación en la música, específicamente en formatos MIDI, hay que aprender las notas y como trabajar con ellas para llegar a cualquier parte.

El *Musical Instrument Digital Interface* [11], MIDI por sus siglas, es una especificación de hardware y software que permite intercambiar información entre varios instrumentos musicales u otros dispositivos, incluyendo sus notas, cambios de programa y control de expresión. La habilidad de transmitir y recibir esta data fue originalmente pensada para presentaciones en vivo, pero rápidamente tuvo un impacto grande sobre

estudios de grabación, producción de audio y video, y ambientes de composición.

Cuthbert expande sobre las capacidades de MIDI, demostrando en una sola nota toda la información que esta contiene. Datos como el nombre musical, el octavo, las tonalidades y datos especiales como el hecho de ser accidental, se pueden determinar de una sola nota en un espacio MIDI.

Powers [12] define el modo musical con tres aplicaciones principales, todas conectadas al significado de *modus*, del latín, una medición, estándar, método o camino. Las tres también denotan una relación entre la nota y su intervalo, sus escalas y su tipo de melodía. Indica que siempre se ha utilizado para designar clases de melodías y más recientemente para determinar ciertos tipos de normas o modelos para composiciones o improvisaciones. Desde una definición más física, lo define como una forma acústica para denotar un patrón particular de vibraciones en las cuales un sistema puede oscilar en una forma estable.

Kaitila [4] continúa la definición de modos musicales, pero se centra más en las partes que integran a un modo; los intervalos. Indica que los modos, o en sus palabras, los colores de las escalas, se basan en los intervalos, o la distancia cromática entre sus notas. A continuación se muestra una tabla adaptada de cómo explica el funcionamiento de los distintos modos:

Escala y modo	Descripción textual	Intervalo entre notas
Cromática	Aleatoria, atonal	1 entre todas.
Mayor	Clásica, feliz	2,2,1,2,2,2,1
Harmónica menor	Encantadora y siniestra	2,1,2,2,1,3,1
Doriana	Fresca, jazz	2,1,2,2,2,1,2
Diatónica	Bizzara, simétrica	2 entre todos.

Cuadro 1: Intervalos para algunos modos de escala. Los números indican cuantas notas de distancia cromática hay entre la nota actual y la siguiente. [4]

Expandiendo aún más sobre los intervalos y su relación con los modos, Cuthbert indica que muchas técnicas de análisis musical, como teoría tonal, modal y cromática se basan en el uso de intervalos. Su biblioteca music21, que tiene una capacidad amplia de obtener datos musicales de MIDI, tiene una colección de clases de intervalos utilizadas en módulos de análisis.

En 1990, Krumhansl y Shmuckler[13] proponen un método de detección de clave para los modos mayor y menor, derivando perfiles relativos a la importancia de los tonos en una escala cromática. Para determinar estas clases de tonalidad, se le solicitaba a personas que escucharan música y que calificaran tonalidades de sondeo en cuanto a que tanto funcionaban para un contexto musical. El algoritmo conocido como Krumhansl-Shmuckler fue propuesto, basado en la idea de que la distribución de clases de tonalidad de las notas en una pieza musical pueden revelar su tonalidad simplemente calculando la correlación de la distribución con cada uno de los perfiles, y prediciendo la clave con la mayor correlación.

### 5.3. Ingeniería de datos y programación *data driven*

Dremio[8] define la ingeniería de datos como el proceso de diseñar y construir sistemas que permiten a un usuario recolectar y analizar data cruda de múltiples fuentes y formatos, con la intención de que puedan encontrar aplicaciones prácticas de la data. En la misma línea, Dremio indica que el análisis de datos es un reto al tener data manejada por distintas tecnologías y estructuras, pero, como se indicó anteriormente, las herramientas de análisis, o generación en este caso, asumen que la data tiene la misma estructura y se maneja de la misma forma.

Dado que la cantidad de datos, fuentes de datos y tipos de datos crecen cada vez más en todo tipo de actividad digital, la importancia de análisis de datos, ciencia de los datos y machine learning también crece. Esto coloca una presión adicional sobre los ingenieros de datos, ya que es más necesario tener acceso rápido a datos limpios y claros que normalmente se encuentran de forma desordenada. Por ello, surge el paso de ETL, *Extract, Load, Transform*, que es un proceso o subproceso donde se extraen los datos de distintas fuentes, se transforman en algo que sea utilizable y confiable, y se carga en un repositorio final fácil de acceder y que tenga exactamente lo necesario para los usuarios finales. [14]

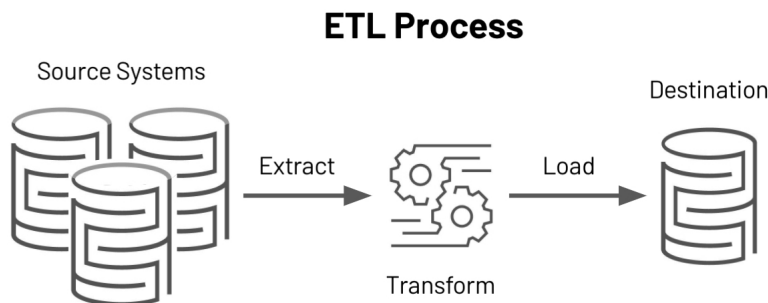


Figura 3: Flujo común de un ETL. [14]

Microsoft describe un problema en el proceso específico de ETL, con la cantidad de datos de entrada que se pueden llegar a manejar, y la necesidad de un recurso intermedio donde se transformen los datos. Por ello, se recomienda en casos, especialmente en lagos de datos donde puede llegar a existir un flujo exponencialmente de información, utilizar un ELT donde el proceso de transformación se haga sobre el mismo repositorio final.

*Data lakes* y *data warehouses* son repositorios de almacenamiento de datos utilizados principal y ampliamente para el uso de *big data*, pero a pesar de que sus usos son parecidos, no son intercambiables. De forma general, un *data lake* se describe como una piscina de datos en crudo, en distintos formatos y sin una estructura definida, cuyo uso no está escrito en piedra; sin embargo, es fácil de acceder y de modificar, y es específicamente útil para científicos de datos, quienes definen la forma de utilizar y



leer la data almacenada. Un *data warehouse*, por su parte, es un repositorio donde los datos están estructurados, filtrados, y procesados para un uso específico. Por las mismas razones, el acceso es más estricto, y las modificaciones se tornan más complejas y con costos mayores. Usualmente, son utilizadas para departamentos de análisis que requieren los datos de una forma específica.[15]

Considerando lo anterior, la arquitectura genérica de *data lake* y *data warehouse* sigue un patrón similar, que según Microsoft tiene la siguiente apariencia:

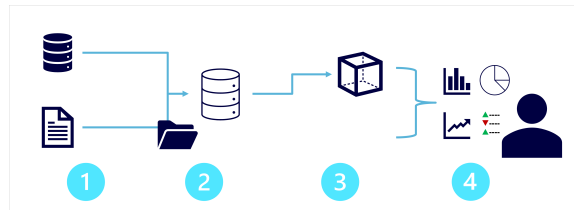


Figura 4: Elementos incluidos generalmente en una arquitectura de *data warehouse* o *data lake*. [16]

Respecto a la Figura 3, los pasos son los siguientes:

1. Ingestión y procesamiento de datos: Paso donde se cargan datos de uno o más repositorios de datos, archivos, corrientes en tiempo real o cualquier otra fuente, no necesariamente estructurada, a un *data lake* o *data warehouse* relacional. Usualmente implica un proceso ETL o ELT donde la data origen se limpia y filtra, y es reestructurada para el análisis deseado.
2. Repositorio de data analítica: El repositorio elegido en el paso anterior, ya sea un lago de datos o almacén.
3. Modelo de datos: Uno o más modelos que pre-agregan la data para facilitar el acceso y lectura de esta, usualmente descritos como cubos por el hecho de tener más de dos dimensiones (por ejemplo, ventas por región, por año).
4. Visualización de datos: La fase del procedimiento en la cual los datos se obtienen de los modelos para ser visualizados en reportes, *dashboards* y otro tipo de visualizaciones.

Bajo el alcance de esta práctica, los pasos a profundizar son los primeros dos. El paso de ingestión y procesamiento de datos consiste en la creación de un flujo de datos, a cual Microsoft se refiere como *pipeline*, que permite la orquestación de uno o varios ETLs. Estos *pipelines* consisten en actividades que operan sobre un *dataset* de entrada, que como fue mencionado anteriormente, puede ser cualquier fuente de datos no necesariamente estructurados, hacen un conjunto de operaciones sobre esta, y la cargan a un *dataset* de salida.

Amazon[17] define un *data lake* como un repositorio central que permite almacenar todos los datos estructurados y no estructurados a cualquier escala, puede estar sin cambios sin tener que estructurarla, y puede ser utilizada directamente para ejecutar

distintos tipos de procesos de análisis. Indica que las capacidades clave de un *data lake* son el movimiento de datos, el almacenamiento y catalogación segura de los datos, análisis de datos, y *Machine Learning*

La existencia de nuevas herramientas de datos permite el cambio a un ELT desde un ETL; en un ELT, las fuentes de datos se cargan automáticamente hacia una fase de *staging* en un estado normalizado, y se realizan transformaciones sobre los datos utilizando herramientas de transformación de datos. Este flujo de trabajo permite que analistas de datos puedan moverse más rápido de datos crudos a percepciones, y permite creación de *pipelines* robustas y repetibles en los datos fuente. [18]

El sistema propuesto y desarrollado consta de tres partes, dos de las cuales representan pasos específicos en el procedimiento ELT sugerido; el sistema de ingestión, que se comporta como el paso de extracción, un sistema de base de datos, que actúa como los pasos de carga y transformación de dicho proceso, y un *script* suelto que funciona para producir el formato final.

### 6.1. Sistema de ingestión de datos

El sistema de ingestión de datos se dividió en tres módulos, debido a la necesidad de obtener datos informativos (genero, etiquetas, nombres de canción, etc.) de varias composiciones, y los datos musicales de estas. El primer módulo es un orquestrador de flujo, el cual indica a los otros dos módulos los datos que deben buscar, es decir, es una especie de diseñador de *pipelines*. Los otros dos módulos consisten en uno que obtiene datos informativos y un listado de composiciones, y otro que utiliza composiciones en formatos MIDI, obtenidas a partir de las que se detectan en el módulo anterior, y las analiza utilizando la biblioteca *music21*.

Todos los módulos se hicieron con scripts de Python3.6 conectados a una base de datos AzureSQL alojada en los servicios web de Microsoft.

### 6.1.1. Módulo de *pipelines*

El módulo principal consiste en un script de Python y una tabla de procesos en AzureSQL. El flujo se define con el siguiente diagrama:

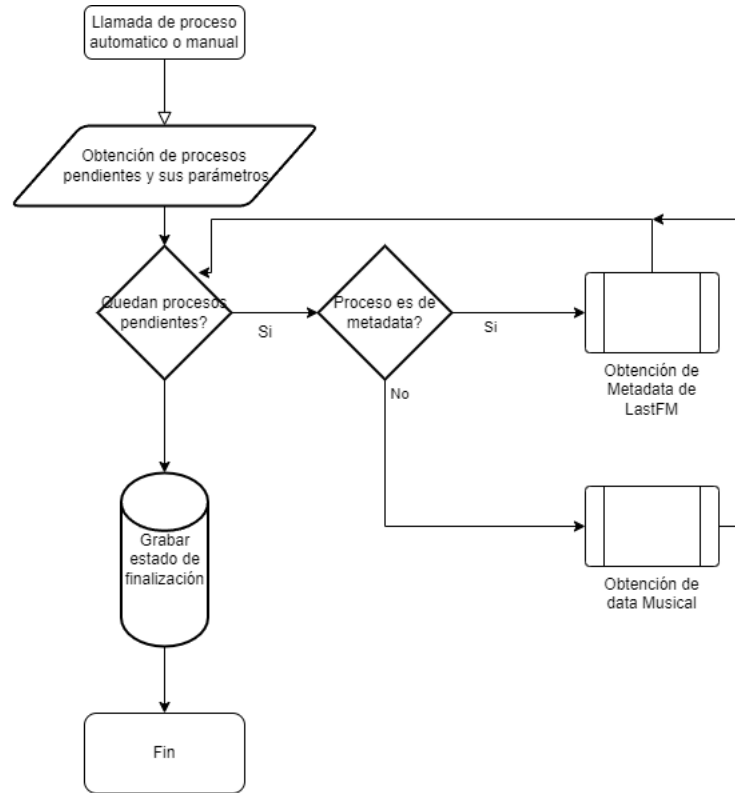


Figura 5: Diagrama de flujo de módulo principal

### 6.1.2. Módulo de obtención de metadata de LastFM

El sitio web Last.FM, cuyo objetivo principal es mantener un historial de las canciones que sus usuarios han escuchado, contiene la capacidad de categorizar no solo por artista y album, sino también le da a los usuarios la opción de colocar en cada canción, artista y album, una colección de etiquetas con las que se pueden describir. Estas etiquetas son altamente útiles en el objetivo de este trabajo, debido a que contienen la información necesaria para categorizar distintas composiciones ya sea por su género, el estilo, el humor, la época a la que recuerda, entre muchos otros descriptores.

Last.FM presenta un API con el cual se pueden obtener datos ya sea de un artista, un album, un *tag* o una canción, todo al mismo tiempo. Debido a la forma en la que está programado, es posible programar una función genérica que se pueda reutilizar para cada uno de estos datos.

De esta forma, se presentan el diagrama de flujo representativos de este módulo.

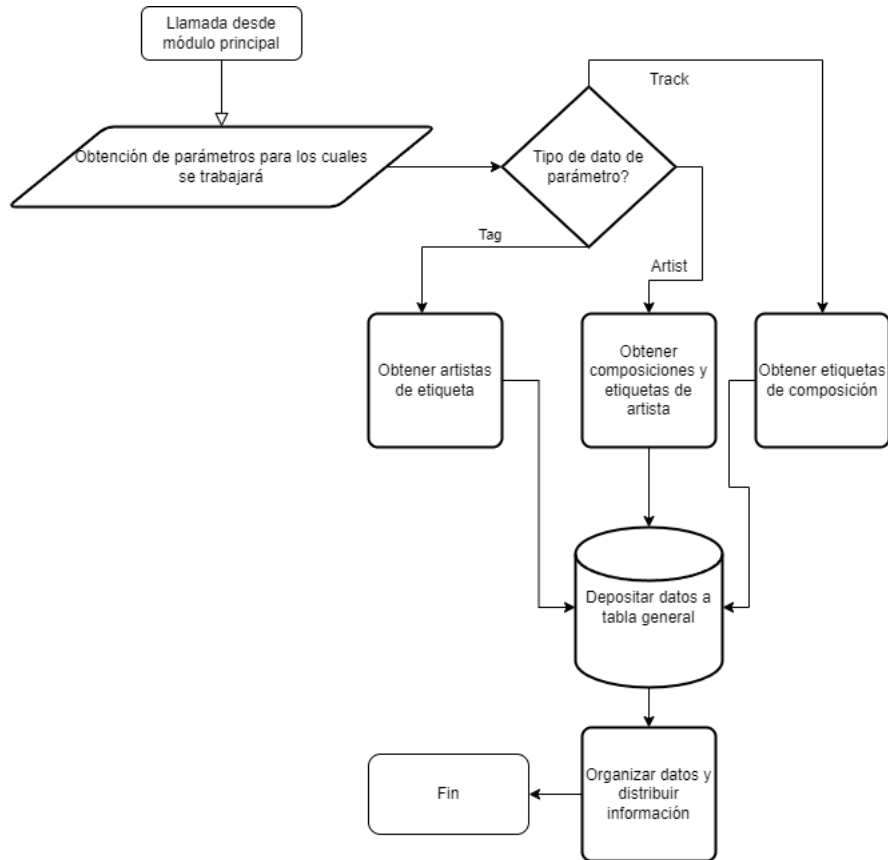


Figura 6: Diagrama de flujo de módulo de LastFM

### 6.1.3. Extracción de Ultimate Guitar y Music21

Para obtener los datos musicales, se utilizó la plataforma de Ultimate Guitar, la cual provee un amplio rango de archivos GP, es decir, MIDI con metadata adicional, categorizados por compositor. Estos archivos solamente contienen la información puramente musical. Para la extracción de estos archivos, se maneja un sistema manual con el listado de composiciones obtenidas por el módulo de Last.FM. El sistema manual consiste en colocar archivos GP en una plataforma Azure Blob Storage, la cual está conectada al sistema de base de datos, a través de una tabla que indica el estado (descargado/pendiente) de la extracción musical.

Estas composiciones son posteriormente analizadas por una librería sencilla de Python que permite la separación de archivos GP en distintas pistas, para obtener solamente las guitarras, y para separar en las secciones si existieran así como el coro y los versos.

Adicionalmente, el módulo cuenta con una funcionalidad prestada de la plataforma music21, citada ampliamente en el marco teórico, con la cual se obtienen los datos musicales para cada una de las composiciones y sus secciones enlistadas en formato GP. Esta plataforma emplea el método Krumhansl-Schmuckler para obtener la clave

y modo de la composición, y un metodo sencillo de detección de valor de nota de MIDI para determinar la nota o acordes. Bajo esta plataforma se obtienen los datos que finalmente se ingresan a la base de datos para el paso de transformación y almacenamiento analítico.

El flujo de trabajo se presenta en el siguiente diagrama:

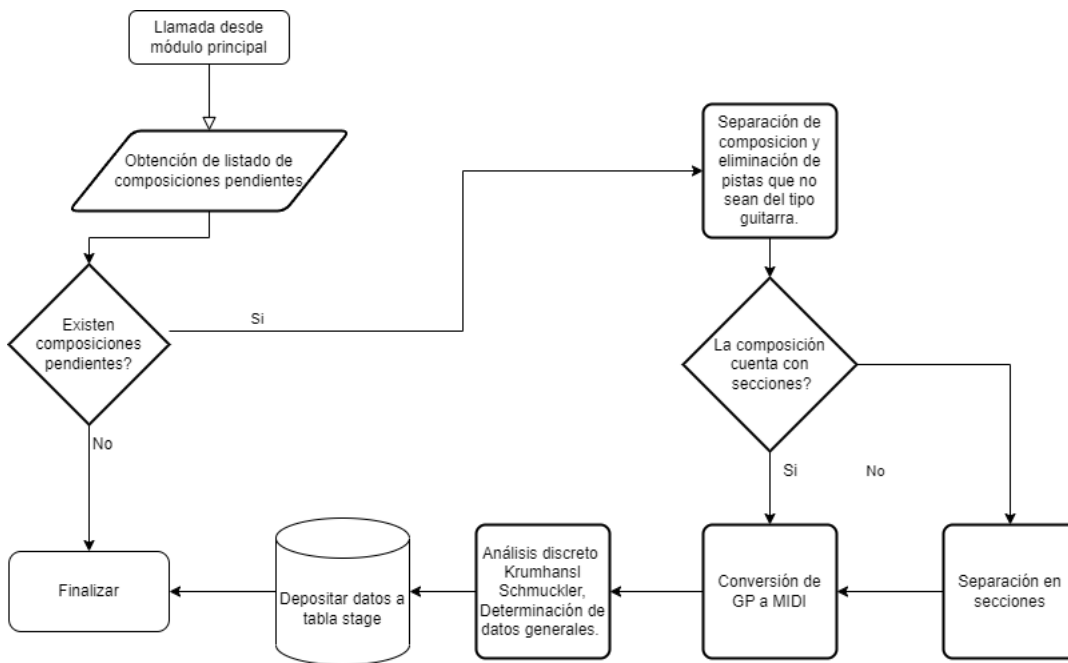


Figura 7: Diagrama de flujo de módulo de Ultimate Guitar y Music21

## 6.2. Diseño de base de datos

La base de datos, apodada FTBOT (Forty-Three Bot) por el canal 43% Artificial, citado al inicio, es una plataforma AzureSQL, la cual simula el comportamiento de un *Data Warehouse*, en cunato a la forma de trabajar una sección de *staging* para el manejo de los ELTs, pero con prácticas similares a las de un *Data Lake*, por la utilización de ELTs, los cuales están alojados directamente sobre la base de datos final, y por la existencia de datos no estrucutrados (MIDI) dentro de la misma. De esta forma, la base de datos se divide en tres módulos; uno de staging, uno de funciones como procedimientos almacenados, y uno de almacenaje final.

La vista general del sistema ELT se presenta a continuación:

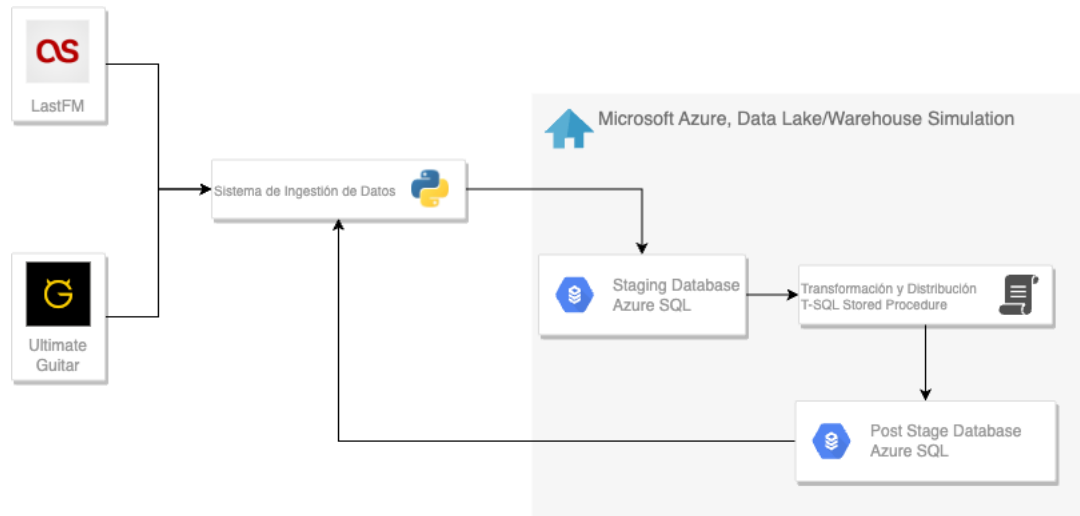


Figura 8: Diagrama de vista general del sistema ELT

### 6.2.1. Fase de *staging* y procedimientos almacenados

La fase de *staging* de la base de datos consta de una tabla principal llamada *incomingvalues*, que contiene todos los datos obtenidos del extractor LastFM sin importar el destino o el parámetro fuente; esta es de forma consecuente utilizada por un procedimiento almacenado, *SP\_DistributeValues* para enviar los datos según su fuente y su destino a las tablas *stage* de cada uno de los parámetros finales: etiquetas, artistas o albums. Las tablas de parámetros en su forma *staging* contienen una estructura similar a la final pero en casos eliminando o agregando campos puramente informativos. Adicionalmente, existen dos tablas de preparación de data para el módulo de Guitar Pro, donde se utilizan distintos campos informativos para poder obtener el ID de la canción investigada de forma que se pueda mantener una normalización adecuada en el esquema final.

A continuación se presenta el diagrama entidad relación de esta parte:



Figura 9: Diagrama entidad relación de fase *staging*

Los procedimientos almacenados son dos; uno para distribuir los valores del módulo de LastFM a sus tablas respectivas, y otro para limpiar y normalizar los ID de los datos musical del módulo Guitar Pro.

Adicional a ello, existen procedimientos almacenados intermedios que son utilizados para enviar los datos de las tablas *staging* a las tablas finales; estos solamente se aseguran de no agregar duplicados, actualizarlos en caso de ser necesario, y asignar ID donde sea necesario.

### 6.2.2. Fase de almacenaje analítico

La fase final del módulo de base de datos es el módulo de almacenamiento, que es donde se pueden acceder los datos. Las tablas existentes en esta fase son parecidas a las de la fase *staging*, pero en esta fase ya contienen sus ID finales y otra data informativa que puede ser útil para el análisis. A continuación se muestra el diagrama entidad relación.



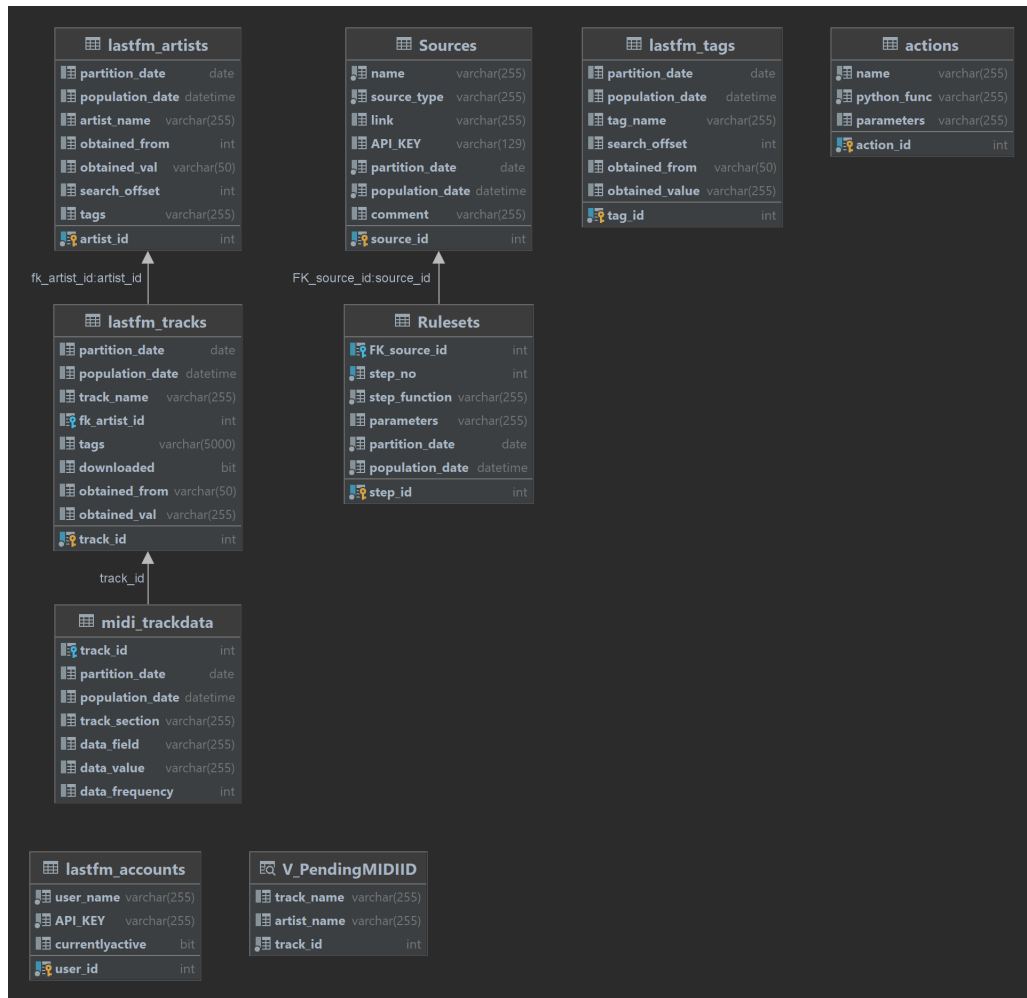


Figura 10: Diagrama entidad relación de repositorio final

Esta fase también cuenta con un procedimiento almacenado que provee los datos musicales de cada canción en formato de tabla, para que esta pueda ser organizada en un script de Python que la convierte al formato final JSON. Adicional a ello, en esta parte se incluyen las tablas del sistema orquestador del pipeline, el cual se describe en la siguiente sección.

### 6.3. Sistema orquestador

El sistema orquestador consta de un script de Python que toma datos de las tablas Rulesets y Actions de la base de datos, para determinar las acciones que deben realizarse automáticamente y con que periodicidad. Este sistema tiene la capacidad de agregar de forma modular nuevas fuentes de datos (por ejemplo, un sitio web como BitMidi) para los datos musicales o metadata informativa. Este funciona de una manera donde cada línea de la tabla Actions contiene un nombre de acción, una función

asociada y un conjunto de parámetros que deben ser utilizados para cumplir la acción. La tabla Rulesets por su parte contiene el orden en el que deben realizarse distintas acciones para terminar un procedimiento específico y los parámetros adicionales que puedan incluirse.

Este sistema permite gran escalabilidad en cuanto a la facilidad de asignar nuevos procesos y apoya en crear un sistema genérico, que no solamente puede ser utilizado para música sino para otros usos parecidos.

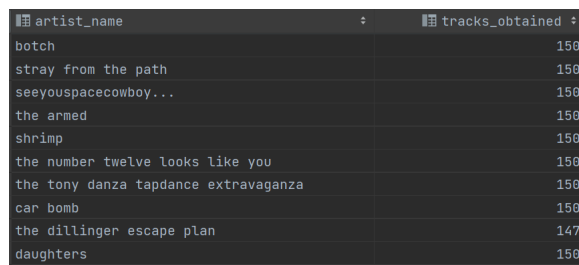
Sin embargo, el proceso tuvo que ser detenido por motivos de costos, los cuales se exploran en la siguiente sección.

## 6.4. Análisis adicional

Como parte adicional al objetivo, se realizaron dos análisis para la determinación de mejoras. Estos son el análisis de costos, que consistió en el uso de *forecast* de la herramienta de Microsoft Azure, y un análisis de complejidad sencillo, donde para cada función del sistema de Python se determinó un costo aproximado según la cantidad de iteraciones basado en un costo arbitrario por acción.

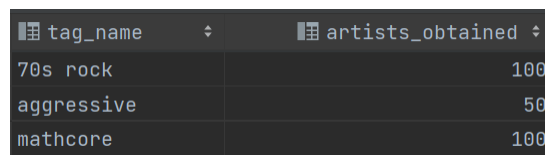
## Resultados

Como resultado principal, se determinó que el sitio web Last.FM es el más útil para la obtención de datos musicales por etiqueta o descriptor, a continuación se presentan los resultados generales de data obtenida en Last.FM para la etiqueta *mathcore*, con la cual se trabajó para este proyecto.



artist_name	tracks_obtained
botch	150
stray from the path	150
seeyospacecowboy...	150
the armed	150
shrimp	150
the number twelve looks like you	150
the tony danza tapdance extravaganza	150
car bomb	150
the dillinger escape plan	147
daughters	150

Figura 11: Tabla de cantidad de composiciones por artista, para etiqueta *mathcore*



tag_name	artists_obtained
70s rock	100
aggressive	50
mathcore	100

Figura 12: Tabla de cantidad de artistas por etiqueta, para las etiquetas que se usaron como prueba

## 7.1. Formato de salida

Siendo esta la información de humor, género o descriptores, se pudo para un conjunto de las canciones obtenidas para la etiqueta *mathcore*, obtener los datos musicales. Se presentan a continuación los resultados del módulo de Guitar Pro y Music21 en su forma más cruda, como se visualizaría en su forma de data lake:

track_id	partition_date	population_date	track_section	data_field	data_value	data_frequency
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Key	A	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Mode	minor	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Time Signature	4/4	8
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Time Signature	5/4	6
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	A2	38
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	C4	18
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	B3	22
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	A3	35
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Major Third	15
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	F4	17
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	E4	27
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	B-3	9
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	D4	25
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	quartal tri	151
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	major seventh	66
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	tritone-fourth	61
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Diminished Fifth with octave doublings	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Perfect Fifth with octave doublings	48
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	B-2	2
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	E-4	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	C#4	6
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Major Sixth	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Augmented Fourth with octave doublings	16
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	minor-second quartal tetra	44
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Perfect Fourth with octave doublings	64
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Diminished Sixth with octave doublings	12
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Augmented Third with octave doublings	6
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Diminished Sixth	2
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	minor triad	35
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	F#4	10
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	C#5	4
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	E3	24
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	major triad	12
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	F3	22
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	augmented triad	12
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	E-3	32
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	G#3	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	G3	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	F#3	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Major Third with octave doublings	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Minor Third	1
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Notes	E6	7
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Minor Seventh	4
1	2022-10-17	2022-10-17 02:33:23.467	NoSec	Chord	Diminished Fourth	1

Figura 13: Tabla de ejemplo de lo que se obtiene para una canción, en este caso *One of Us Is The Killer* de The Dillinger Escape Plan

Con esta estructura de datos tomada en cuenta, se determinó que el formato JSON es el más adecuado para poder mostrar los datos al usuario de una forma fácil de comprender para uso de máquina. El formato JSON se presenta a continuación.

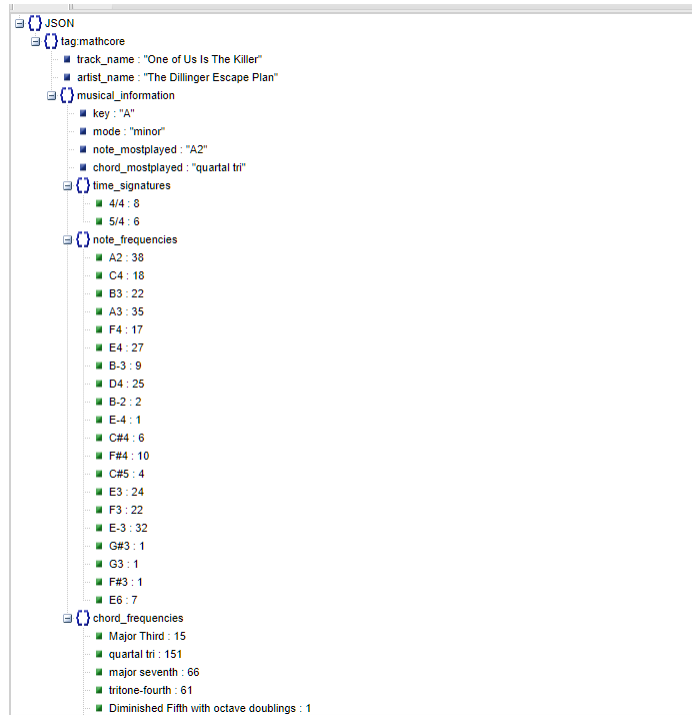


Figura 14: JSON con los datos crudos para una composición de *mathcore*, en este caso *One of Us Is The Killer* de The Dillinger Escape Plan

De la misma manera, se presentan diagramas de frecuencia para las notas y acordes más utilizados en la etiqueta *mathcore*, según las composiciones analizadas. Esto con la intención de darle al usuario una facilidad de lectura previa al uso.

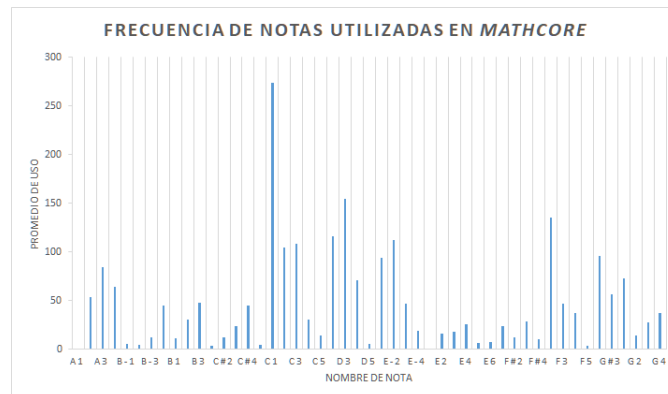


Figura 15: Promedio de notas utilizadas en una composición de *mathcore*, según las composiciones analizadas

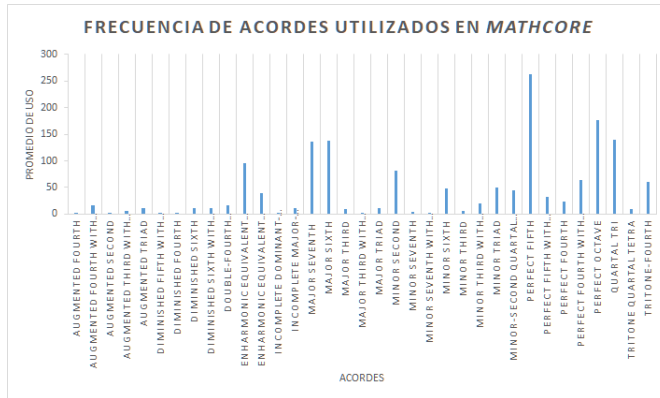


Figura 16: Promedio de acordes utilizados en una composición de *mathcore*, según las composiciones analizadas

## 7.2. Encuestas de importancia y probabilidad

Teniendo los resultados del formato ideal en cuenta se realizó un conjunto de 6 encuestas para determinar la utilidad de los datos presentados en el formato, añadiendo otros datos que se pueden encontrar en un pentagrama escrito. Los resultados de las encuestas son los siguientes:

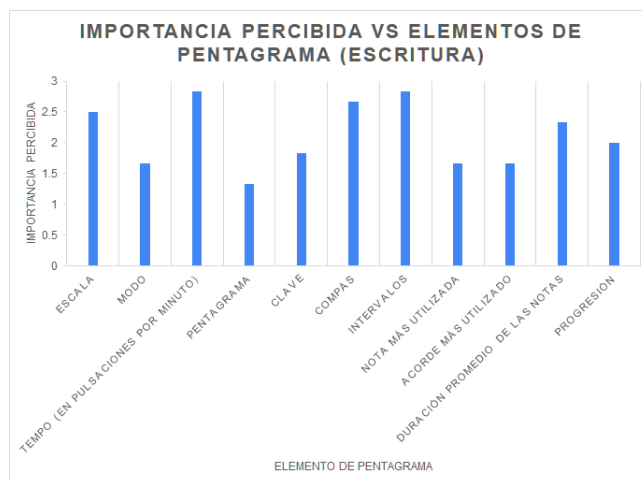


Figura 17: Importancia percibida de cada elemento del pentagrama al momento de escribir una composición musical, donde 0 es nada importante y 4 es casi esencial.

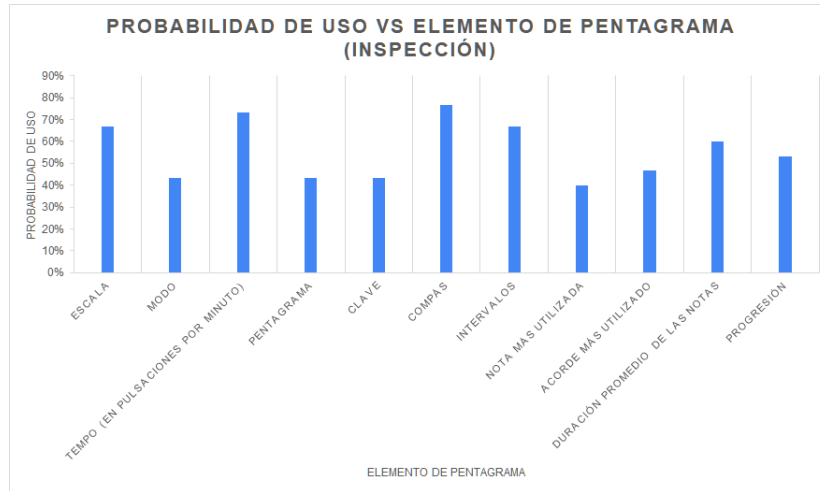


Figura 18: Probabilidad de utilizar cada elemento del pentagrama para intentar categorizar una composición por género, humor o descriptor general

### 7.3. Análisis de complejidad y costos

Función	Complejidad	Observaciones
lastfm_getinfo	$o(n)$	Se desconoce a nivel de API la complejidad. $n$ es la cantidad de etiquetas, composiciones o artistas a buscar.
lastfm_dumptodb	$o(n)$	$n$ es la cantidad de objetos que se insertan a la base de datos
db_getinfo	$o(n)$	$n$ es la cantidad de objetos que se buscan de la base de datos
getTrackDetails	$o(m)$	Se desconoce la complejidad interna de music21. $m$ es la cantidad de notas en cada composición, que se asume más grande que la cantidad de composiciones a procesar por lote.
db_postdetails	$o(n)$	$n$ es la cantidad de composiciones que se insertan a la base de datos.

Cuadro 2: Tabla de complejidad para cada una de las funciones de los módulos de Python

Finalmente, se presentan los costos de manejo de la base del sistema de base de datos con Azure durante los tres meses que estuvo activa previa a la redacción del documento. También se presenta un pronóstico de costos de parte de la herramienta de Microsoft Cost Analysis para los próximos tres meses, si se siguiera utilizando la misma cantidad de datos.

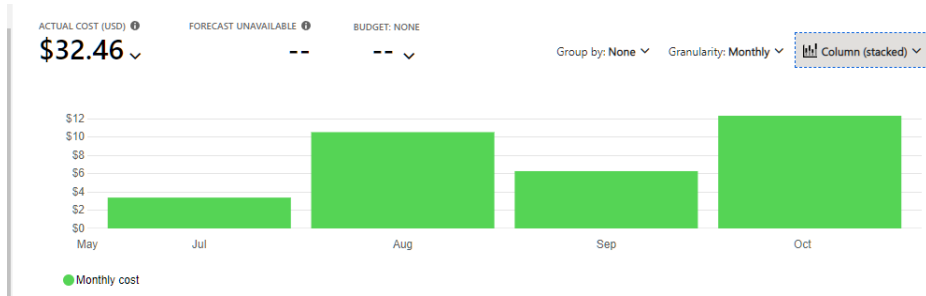


Figura 19: Costos mensuales del servicio de base de datos prestado por Microsoft Azure

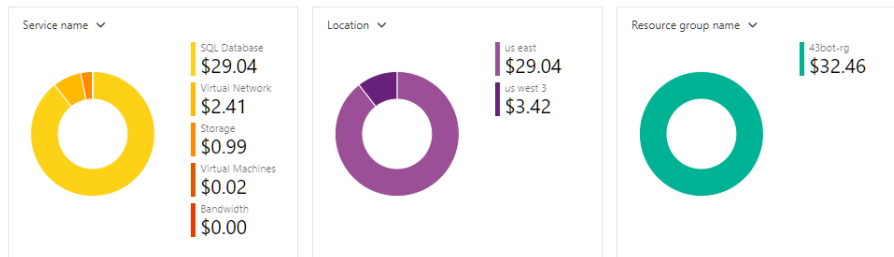


Figura 20: Distribución de costos del servicio prestado por Microsoft Azure

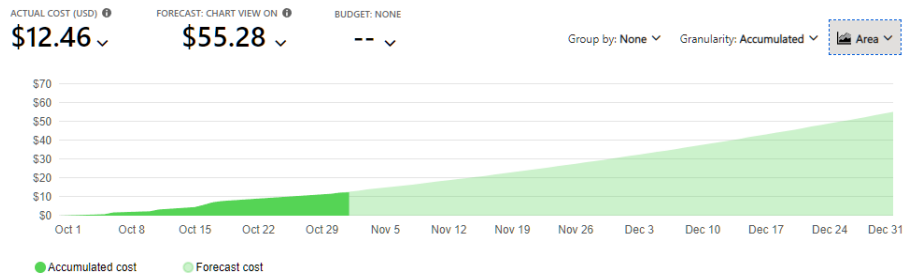


Figura 21: Pronóstico de costos de mantenimiento de Microsoft Azure



---

## Análisis de resultados

---

Analizando los resultados presentados en la sección anterior, se comparan los objetivos a cumplir contra la información que se pudo determinar a través de los resultados. De esta forma, las figuras 11 y 12 presentan un fuerte argumento para la utilización de Last.FM como uno de los dos repositorios web de música, indicados en uno de los objetivos. Esto ya que, con solamente una etiqueta, de naturaleza bastante esotérica por el público objetivo reducido para el cual se suelen componer canciones categorizables como *mathcore*, se pudieron obtener 100 compositores en una o dos ejecuciones del API (no se quisieron realizar más, por motivos de sobrecarga del mismo y para evitar sobrecargar costos; este punto se analizará con más detalle para discutir las figuras 19, 20 y 21, al igual que el Cuadro 2).

De estos 100 compositores, tomando en cuenta las mismas consideraciones de costos y manejo de información, se pudo obtener hasta 150 composiciones de 10 de ellos. Estas incluyen, en casos, versiones alternas, en vivo o con algun invitado especial, para las cuales suele cambiar de alguna forma la composición en general, aunque mantienen un cierto respeto al sonido usual del compositor. Tomando esto en cuenta, la utilidad de Last.FM como herramienta para obtención de datos es muy valiosa, dado que en muchos casos las versiones alternas no se pueden encontrar en repositorios de música oficiales como Apple Music, Spotify o Deezer, y son enviadas como *scrobbles* a Last.FM por usuarios que las obtienen de grabaciones o medios alternativos.

Para continuar dentro de la selección del repositorio del que se obtiene el formato, se encuentra la página Ultimate Guitar, la cual contiene una gran variedad de archivos Guitar Pro, que a su vez contienen información valiosa en formato MIDI, junto con una capacidad de poder separar por secciones de pentagrama (así como el coro o los

versos), y de cierta forma lo más útil, una posibilidad de filtrar por calificación cada una de los archivos Guitar Pro enviados por los mismos usuarios. Esta última opción permite que al momento de elegir los archivos a analizar por el sistema, se puedan elegir explícitamente los que cumplan con cierto nivel de calificación, evitando así popular la base de datos con información errónea.

Agregando a este último punto, se encuentra una gran desventaja respecto a la herramienta Guitar Pro: dado que los archivos disponibles son casi todos enviados por usuarios, en varios casos y especialmente para etiquetas esotéricas como la que se utilizó para la mayor parte de este trabajo, no existen archivos para muchas de las composiciones que se podría encontrar en Last.FM. Sin embargo, dado la naturaleza de la intención del trabajo, la cual permite la búsqueda de información por más de una etiqueta, se considera que las composiciones disponibles en formato Guitar Pro son suficientes para el propósito.

De esta forma, se pudo determinar que para los propósitos del trabajo, el formato Guitar Pro, y en su defecto MIDI, los cuales incluyen la mayor cantidad de datos musicales en un formato de bajo peso y con fácil acceso. Teniendo esto en cuenta, se consideró trivial la separación de las composiciones en secciones, ya que para la gran mayoría de composiciones, las características musicales no suelen cambiar de sección a sección, más allá de las características de tiempo, que usualmente tampoco ven mayor cambio. La razón por la cual se presentó la opción de separar por secciones, se encuentra en la existencia de algunas composiciones del género *progressive*, las cuales suelen modificar estos valores drásticamente, pero de igual forma presentan una gran complejidad a la hora de categorizarlas.

El formato digital propuesto (Figura 14) se determinó en base a la suposición de que la detección de patrones es de los puntos más importantes en la generación procedural. Dado esto, considerando que la clave de una composición y el modo en el que esta composición se presenta, se determinaron en la investigación como los principales patrones en la música escrita, se incluyeron como datos principales la clave y el modo. Posteriormente, también se determinó que los factores de tiempo, así como el compás (*time signature*) y el tempo podrían ser datos útiles para determinar patrones. Sin embargo, por limitaciones técnicas el tempo no fue tomado en cuenta para este estudio. Finalmente, se incluyen las notas y acordes más utilizadas en cada composición, con la intención de poder determinar intervalos, los cuales tampoco se obtuvieron por limitaciones, pero pueden ser obtenidos a través de la clave y las notas que se consideran centrales en la composición, y determinar el tipo de sonido que se pretendía transmitir con la composición original.

También se presenta la frecuencia de notas y acordes, con la intención de cumplir con el objetivo principal de un *data lake*, que es contener todos los datos posibles en un formato crudo, añadido a presentar al usuario final la capacidad de ejecutar un algoritmo Krumhansl Schmuckler con sus propios parámetros sobre los datos, y así determinar por su cuenta la clave y modo. Por otra parte, también se puede utilizar para generar diagramas de frecuencia como los de las figuras 15 y 16, donde es fácil observar las notas y acordes más utilizados en composiciones que cumplan con un

descriptor.

Agregando a la toma de decisiones del formato final, se toma en cuenta el estudio antecedente de Tshepo, quien encontró que más allá de los datos de tiempo y croma de la música (progresión de acordes), el contraste entre las frecuencias auditivas de cada sonido en una composición y la información de tonalidad suelen ser los más importantes. De esta forma, se presentó mayor importancia sobre esto.

Finalmente, respecto al formato de salida, se tienen las encuestas de importancia percibida de elementos, y la probabilidad de uso para inspección. En estas, tres críticos de música y tres compositores independientes fueron presentados con dos preguntas. La primera, referente a la importancia que consideran que tiene cada elemento al momento de escribir una composición nueva, y la segunda referente a la probabilidad de investigar un elemento específico si se les preguntara qué género, humor o descriptor se podría asignar a una composición solo leyendo el pentagrama.

En la primera pregunta, Figura 17, se observa un peso para el tempo, compás, e intervalos, seguido de la escala, la duración promedio y la progresión. De forma interesante, el modo, que desde un inicio se consideró como uno de los más importantes, no parece tener mayor peso. Sin embargo, dado que un intervalo es posible de determinar a través del modo y vice versa, y que el formato presenta la capacidad de mostrar modo, la utilidad se presenta como intercambiable. Como comentario interesante, uno de los encuestados indica que al momento de escribir música, no utiliza realmente ninguno de estos elementos sino que escribe lo que se le venga a la mente. Sin embargo, esto presenta una respuesta ambigua en cuanto a si esto significa que todos tienen la misma importancia, o ninguna, por lo que esa respuesta específica fue descartada.

Para la segunda pregunta, Figura 18, se observa un patrón parecido en las más importantes, pero en este caso los intervalos tuvieron una menor importancia que el compás.

Tomando todos estos aspectos en cuenta, no es posible obtener una respuesta concreta en cuanto a cuáles de los elementos musicales son los que se deberían tomar en cuenta para la generación procedural. Sin embargo, si queda claro que los intervalos (modo y nota más utilizada, en su defecto) y el compás son de los más importantes para utilizar. De esta forma, se considera que el formato de salida generado cumple con más de dos elementos clave para la generación procedural de música y su estudio. Adicional a ello, la falta de claridad en cuanto al resto de los elementos propone un argumento fuerte para la utilización de un *data lake* que contenga toda la información posible sobre datos musicales, ya que con la cantidad suficiente de datos, y un análisis profundo de estos (el cual se sale del alcance del trabajo), existe la posibilidad de encontrar patrones escondidos.

Dentro de las limitaciones para el formato, como ya fue mencionado anteriormente, se encuentra que no se obtuvieron datos como tempo, longitud de notas, o intervalos. Específicamente para el dato de intervalos, se cubre la necesidad de determinarlo al tener el modo y la nota más repetida (que se considera la central), pero definitivamente

existe una posibilidad de determinarlos dentro del mismo algoritmo. Adicionalmente, el algoritmo utilizado para determinar la clave, Krumhansl-Shmuckler, solamente calcula el modo menor o mayor, por lo que la información modal puede no estar del todo completa.

Añadiendo a las limitaciones, se presentan los datos de complejidad y costos, con las figuras 19, 20 y 21. La principal limitante para poder obtener resultados mas concretos fue la utilización financiera de la base de datos en sí; se observa que para el mes de agosto, donde se crearon la mayor parte de los datos (aproximadamente 3000 líneas de tablas), se observa un crecimiento de costos de aproximadamente el triple, llegando a \$10, y para el mes de octubre, en el cual se hicieron la mayor cantidad de pruebas también se ve un aumento parecido. Suponiendo que se tenía planeado hacer un estimado de 10 veces la cantidad de datos que se obtuvieron en cuanto a metadata, y una cantidad aún más grande de datos musicales (de los cuales sólo se obtuvieron para 20 canciones por las mismas limitaciones), estos costos pudieron haber llegado a niveles que no se habían considerado en la planificación. Esta misma limitación provocó que el sistema automático no se pudiera implementar completamente y fue el factor limitante en cuanto a la cantidad de canciones analizadas.

Tomando en cuenta el Cuadro 2, donde se observan las complejidades de cada función, se observa la posibilidad de reducir la complejidad de todas, idealmente a una complejidad logarítmica para poder reducir la carga sobre una máquina virtual donde estas se estarían ejecutando. Sin embargo, para los costos de bases de datos, esta complejidad es trivial y no presenta ningún efecto real.

Como conclusión general del análisis, es interesante notar que no existe una respuesta clara respecto a las características más útiles en la composición procedural, pero que utilizando las características definidas es posible determinar otras que las personas encuestadas y los estudios antecedentes consideran importantes. Sin embargo, existe una gran dificultad en cuanto a los costos de mantenimiento y ejecución de un sistema de este tipo, lo cual presenta conflicto respecto a la escalabilidad del mismo desde una perspectiva financiera, a pesar de que la escalabilidad en términos de modularidad sí es un punto fuerte a favor.

Se logró desarrollar un sistema ELT, con escalabilidad en términos de los módulos que se le pueden asociar, que permite la obtención del formato Guitar Pro, de la fuente Ultimate Guitar, que a su vez permitió proponer un formato genérico de información musical que contiene el modo, clave y la nota más utilizada por una composición que cumple con un descriptor específico. De esta forma, se cumple con la necesidad de por lo menos dos características importantes en la generación procedural, y se pueden derivar otras características que se determinaron como importantes.

De la misma forma, se logró determinar que el formato digital más accesible para la obtención de características musicales es Guitar Pro, y en su defecto MIDI, ya que estos contienen datos clave para el análisis y están escritos en un lenguaje de máquina con la intención de ser legibles por humanos.

Adicionalmente, se seleccionó el sitio web Ultimate Guitar como el repositorio de música digital del cual se pueden obtener archivos Guitar Pro, gracias a su capacidad de filtrar por banda y por calificación de usuarios. A este sitio web, se le acopla la utilización del sitio Last.FM, que contiene la metadata necesaria para determinar las composiciones a estudiar en base a un descriptor.

Finalmente, se logró proponer una estructura JSON con el formato genérico que contiene los datos de modo, clave, nota y acorde más utilizado, y la frecuencia de utilización de notas y acordes de una composición específica.



## CAPÍTULO 10

---

### Recomendaciones

---

Sobre la selección del formato a utilizar, se recomienda en futuros estudios la obtención de más datos, específicamente la duración promedio de las notas y el tempo de las composiciones, aunque se sugiere incluir todo lo posible para completar las necesidades del *data lake*. También se recomienda la utilización de un algoritmo más avanzado que el de Krumhansl-Shmuckler, para la determinación específica de los modos musicales.

Sobre las limitantes que se encontraron, se recomienda hacer una planificación adecuada de presupuesto para los recursos que se utilizaran. Esto para permitir una escalabilidad adecuada, del punto de vista financiero, y poder obtener la mayor cantidad de datos sin incurrir en costos imprevistos.





- 
- [1] T. Nkambule, “Classification of Music by Genre using Probabilistic Models and Deep Learning Models,” Tesis doct., University of the Witwatersrand, 2020.
  - [2] A. Olteanu, *GTZAN Dataset - Music Genre Classification*, 2019. dirección: <https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification>.
  - [3] P. Sarmiento, A. Kumar, C. Carr, Z. Zukowski, M. Barthelet e Y.-H. Yang, *DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models*, 2021.
  - [4] C. Kaitila, *Procedural Music Generation*, 2017. dirección: <https://www.procjam.com/tutorials/en/music/>.
  - [5] A. Waweru, *Understanding Pattern Recognition in Machine Learning*, 2021. dirección: <https://www.section.io/engineering-education/understanding-pattern-recognition-in-machine-learning/>.
  - [6] C. Payne, *Musenet*, 2019. dirección: <https://openai.com/blog/musenet/>.
  - [7] J. Van Brummelen, *Procedural Generation: Creating 3D Worlds with Deep Learning*, s/f. dirección: [http://www.mit.edu/~jessicav/6.S198/Blog\\_Post/ProceduralGeneration.html](http://www.mit.edu/~jessicav/6.S198/Blog_Post/ProceduralGeneration.html).
  - [8] DremIO, *Introduction to Data Engineering*, s/f. dirección: <https://www.dremio.com/resources/guides/intro-data-engineering>.
  - [9] S. Reich, *Biography of Steve Reich*, 2022. dirección: <https://www.steverreich.com/>.
  - [10] M. Cuthbert, *Music21*, 2013. dirección: <http://web.mit.edu/music21/doc/about/about.html>.
  - [11] MIDI, *MIDI Specification*, s/f. dirección: <https://www.midi.org/specifications/midi1-specifications>.
  - [12] H. S. Powers, “Mode (from Lat. modus: ‘measure’, ‘standard’ ; ‘manner’, ‘way’),” *Grove Music Online*, vol. 93, págs. 1-52, 2001.
  - [13] S. T. Madsen, G. Widmer y J. Kepler, “Key-finding with interval profiles,” en *ICMC*, 2007.

- [14] Databricks, *Extract Transform Load (ETL)*, s/f. dirección: <https://www.databricks.com/glossary/extract-transform-load>.
- [15] Talend, *Data Lake vs Data Warehouse*, s/f. dirección: <https://www.talend.com/resources/data-lake-vs-data-warehouse/>.
- [16] Microsoft, *Describe data warehousing architecture*, s/f. dirección: <https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/2-describe-warehousing?ns-enrollment-type=learningpath&ns-enrollment-id=learn.wwl.azure-data-fundamentals-explore-data-warehouse-analytics>.
- [17] Amazon, *What is a data lake?* s/f. dirección: <https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/>.
- [18] S. Spediacci y A. Deora, *Speeding Time to Insight with a Modern ELT Approach*, s/f. dirección: [https://www.databricks.com/session\\_eu20/speeding-time-to-insight-with-a-modern-elt-approach](https://www.databricks.com/session_eu20/speeding-time-to-insight-with-a-modern-elt-approach).

Elementos musicales

En esta sección se le presentarán varios elementos de una composición escrita en pentagrama. Por favor indique: al momento de analizar o componer una canción, ¿qué tan importante considera cada elemento?.

Elementos \*

	Nada importante	Poco importante	Neutral	Importante	Muy importante
Escala	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Modo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tempo (en pulsaciones por minuto)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pentagrama	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Clave	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Compás	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Intervalos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Nota más utilizada	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Acorde más utilizado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Duración promedio de las notas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Progresión	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figura 22: Encuesta de importancia de elementos musicales, pregunta 1

Determinación de ánimo, género o descriptores.

A continuación se le presentarán los mismos elementos de la sección anterior. Sin escuchar una composición y solo teniendo el pentagrama, ¿Qué tan probable sería que utilice estos elementos para categorizar la composición en género, o para describir el ánimo de la misma?

Elementos \*

	Nada probable	Poco probable	Neutral	Probable	Muy probable
Escala	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Modo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tempo (en pulsaciones por minuto)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pentagrama	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Clave	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Compás	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Intervalos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Nota más utilizada	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Acorde más utilizado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Duración promedio de las notas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Progresión	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figura 23: Encuesta de importancia de elementos musicales, pregunta 2