



**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INFORMÁTICA**

**GRADO EN INGENIERÍA INFORMÁTICA**

**TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN**

**TRABAJO FIN DE GRADO**

**PROPUESTA TECNOLÓGICA PARA APOYAR EL  
APRENDIZAJE DE UN INSTRUMENTO MONÓDICO  
MEDIANTE COMPARACIÓN DE PASAJES MUSICALES**

Eduardo Mora González

Septiembre, 2019





**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INFORMÁTICA**  
**TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN**

**TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN**

**TRABAJO FIN DE GRADO**

**PROPUESTA TECNOLÓGICA PARA APOYAR EL  
APRENDIZAJE DE UN INSTRUMENTO MONÓDICO  
MEDIANTE COMPARACIÓN DE PASAJES MUSICALES**

Autor: Eduardo Mora González

Director: José Bravo Rodríguez

Director: Iván González Díaz.

Septiembre, 2019

TFG

© Eduardo Mora González, 2019

Este documento se distribuye con licencia Creative Commons Atribución Compartir Igual 4.0. El texto completo de la licencia puede obtenerse en <https://creativecommons.org/licenses/by-sa/4.0/>.

La copia y distribución de esta obra está permitida en todo el mundo, sin regalías y por cualquier medio, siempre que esta nota sea preservada. Se concede permiso para copiar y distribuir traducciones de este libro desde el español original a otro idioma, siempre que la traducción sea aprobada por el autor del libro y tanto el aviso de copyright como esta nota de permiso, sean preservados en todas las copias.



TRIBUNAL:

Presidente: \_\_\_\_\_

Vocal: \_\_\_\_\_

Secretario: \_\_\_\_\_

FECHA DE DEFENSA: \_\_\_\_\_

CALIFICACIÓN: \_\_\_\_\_

PRESIDENTE

VOCAL

SECRETARIO

Fdo.:

Fdo.:

Fdo.:



*En la tormenta encontré el equilibrio*  
(Fangoria)





## Resumen

Está claro que la aparición y desarrollo de las Tecnologías de la información y comunicación en los últimos tiempos ha revolucionado la manera en la que percibimos e interactuamos con el mundo. La educación y la música también se han visto afectadas por las T.I.C.

La presente investigación supone un nuevo acercamiento entre la Educación Musical y la tecnología actual cuyo fin es la eficiencia en el aprendizaje de un instrumento monódico.

Para ello se ha diseñado un sistema que apoya el aprendizaje instrumental a través de la comparación de pasajes musicales, utilizando la técnica de *pitch tracking* para el procesamiento de sonido y el formato de escritura musical *MusicXML* para la lectura de las partituras a comparar entre lo interpretado por el alumno en su estudio instrumental, captado por un micrófono, y lo que debería haber sonado según la partitura original, corrigiendo así los errores durante el proceso de estudio.



### **Abstract**

Our vision and the way of interacting with the world have been changing in light of the new technologies. Education and music have also been affected by.

This investigation is a new approach between the Musical Education and the current technologies whose aim is being able to enhance the learning process of a monodic instrument.

Through the comparison of musical passages, using the *Pitch Tracking* technique for sound computer processing and the *MusicXML* musical writing format for reading the scores, this system is able to compare what the student is playing and what should have played. In order to collect the student's interpretation, a microphone is used. Using this system, errors can be corrected during the learning process.



# AGRADECIMIENTOS

---

Quisiera agradecer a todas las personas y entidades la ayuda que me han prestado en la realización de este Trabajo Fin de Grado.

En primer lugar, a mis padres, hermano y abuelos, que siempre han estado a mi lado apoyándome para seguir hacia adelante.

Quisiera hacer extensiva mi gratitud a José Bravo Rodríguez e Iván González Díaz, tutores de este T.F.G., por todo lo que me han aportado y ayudado a juntar mis dos grandes pasiones: la música y la informática.

Un agradecimiento muy especial a todos mis compañeros y amigos, que me han acompañado durante esta etapa de mi vida.

A todos ellos, muchas gracias.

*Eduardo Mora González*



# ÍNDICE GENERAL

---

<b>Índice de figuras</b>	<b>XIX</b>
<b>Acrónimos</b>	<b>XXI</b>
<b>1. INTRODUCCIÓN</b>	<b>1</b>
1.1. CONTEXTO . . . . .	1
1.2. MOTIVACIÓN . . . . .	3
1.3. OBJETIVOS . . . . .	3
1.3.1. OBJETIVO GENERAL . . . . .	3
1.3.2. OBJETIVOS PARCIALES . . . . .	3
1.4. ESTRUCTURA DE LA MEMORIA . . . . .	4
<b>2. FUNDAMENTOS Y ANTECEDENTES</b>	<b>5</b>
2.1. FUNDAMENTOS FISICOS . . . . .	5
2.1.1. EL SONIDO . . . . .	5
2.1.2. PITCH TRACKING . . . . .	7
2.2. FUNDAMENTOS MUSICALES . . . . .	10
2.2.1. MODOS BÁSICOS DE ORGANIZACIÓN TEMPORAL . . . . .	10
2.2.2. FORMA MUSICAL . . . . .	11
2.2.3. SISTEMA DE AFINACIÓN TEMPERADO . . . . .	12
2.2.4. CATEGORIZACIÓN DE LAS NOTAS MUSICALES . . . . .	13
2.3. ANTECEDENTES . . . . .	13
2.3.1. INFORMÁTICA Y MÚSICA . . . . .	13
2.3.2. APLICACIONES ACTUALES . . . . .	14
2.3.3. LÍNEAS DE INVESTIGACIÓN ACTUALES . . . . .	15
<b>3. METODOLOGÍA DE TRABAJO</b>	<b>17</b>
3.1. METODOLOGÍAS . . . . .	17
3.1.1. METODOLOGÍAS ÁGILES . . . . .	18
3.1.2. METODOLOGÍAS ÁGILES VS METODOLOGÍAS TRADICIONALES . . . . .	19
3.1.3. ADECUACIÓN DEL DESARROLLO ÁGIL AL PROYECTO . . . . .	19
3.2. PROTOTIPADO INCREMENTAL . . . . .	20

3.3.	JUSTIFICACIÓN DE LA METODOLOGÍA . . . . .	20
3.4.	PLANIFICACIÓN . . . . .	21
3.4.1.	PLANIFICACIÓN GENERAL . . . . .	21
3.4.2.	CALENDARIO DEL PROYECTO . . . . .	22
3.4.3.	PLANIFICACIÓN LOCAL: ESTADO DEL ARTE . . . . .	22
3.4.4.	PLANIFICACIÓN LOCAL: APLICACIÓN . . . . .	23
3.4.5.	PLANIFICACIÓN LOCAL: EXPERIMENTOS Y PRUEBAS . . . . .	29
<b>4.</b>	<b>RESULTADOS</b>	<b>31</b>
4.1.	VISIÓN GENERAL . . . . .	31
4.2.	GRABACIÓN DEL AUDIO . . . . .	31
4.2.1.	RECONOCIMIENTO DE LA SEÑAL DE AUDIO DE ENTRADA EN ANDROID	32
4.3.	PITCH TRACKING . . . . .	32
4.3.1.	DIVISIÓN DE LA SEÑAL EN VENTANAS . . . . .	34
4.3.2.	ANÁLISIS DE LA VENTANA . . . . .	35
4.3.3.	TRANSFORMACIÓN DE LA SEÑAL . . . . .	35
4.3.4.	AUTOCORRELACIÓN DE LA SEÑAL . . . . .	36
4.3.5.	DETENCIÓN DE LOS PICOS . . . . .	36
4.3.6.	OBTENCIÓN DE LA FRECUENCIA DE LA VENTANA . . . . .	37
4.4.	DISCRETIZACIÓN DE LAS FRECUENCIAS . . . . .	37
4.4.1.	OBTENER LA NOTA ASIGNADA A CADA FRECUENCIA . . . . .	37
4.4.2.	CALCULO DE LAS FRECUENCIAS DE LAS NOTAS . . . . .	38
4.4.3.	BÚSQUEDA DE LAS NOTAS A TRAVÉS DE UNA FRECUENCIA . . . . .	39
4.4.4.	DURACIÓN DE LAS NOTAS E IDENTIFICACIÓN DE LOS SILENCIOS . . . . .	39
4.5.	LEER ARCHIVO MUSICXML . . . . .	42
4.6.	COMPARAR ARCHIVO MUSICXML Y DAR FEEDBACK . . . . .	43
<b>5.</b>	<b>EVALUACIÓN</b>	<b>47</b>
5.1.	EVALUACIÓN PITCH TRACKING . . . . .	47
5.1.1.	EVALUACIÓN DE LA DIVISIÓN EN VENTANAS . . . . .	47
5.1.2.	EVALUACIÓN DE LA OBTENCIÓN DEL PICO MÁXIMO . . . . .	47
5.1.3.	EVALUACIÓN DEL CALCULO DE LA FRECUENCIA . . . . .	48
5.2.	EVALUACIÓN DISCRETIZACIÓN DE LAS FRECUENCIAS . . . . .	49
5.2.1.	EVALUACIÓN ASIGNACIÓN A CADA FRECUENCIA UNA NOTA MUSICAL	49
5.2.2.	EVALUACIÓN DE LA ASIGNACIÓN DE FIGURAS MUSICALES . . . . .	50
5.3.	EVALUACIÓN DE LA COMPARACIÓN DE PARTITURAS . . . . .	52
5.3.1.	EVALUACIÓN DE LA LECTURA DEL FICHERO MUSICXML . . . . .	52
5.3.2.	EVALUACIÓN DE LA COMPROBACIÓN DE PARTITURAS . . . . .	54



---

<b>6. CONCLUSIONES</b>	<b>57</b>
6.1. CONSECUCIÓN DE OBJETIVOS . . . . .	57
6.2. COMPETENCIAS . . . . .	58
6.3. DIFICULTADES . . . . .	58
6.4. TRABAJO FUTURO . . . . .	59
 <b>A. DIAGRAMA DE GANTT</b>	 <b>61</b>
 <b>B. ARCHIVO MUSICXML DE LA PARTITURA ORIGINAL DE LAS PRUEBAS</b>	 <b>65</b>
 <b>C. DIAGRAMA DE COMUNICACIÓN</b>	 <b>71</b>
 <b>Bibliografía</b>	 <b>73</b>



# ÍNDICE DE FIGURAS

---

2.1. Representación del sonido con sus parámetros . . . . .	5
2.2. Representación de la duración del sonido . . . . .	6
2.3. Representación de la altura del sonido . . . . .	6
2.4. Representación de distintos tipos de timbres . . . . .	7
2.5. Representación de la intensidad del sonido . . . . .	7
2.6. Diferentes tipos de compases comunes. Figura extraída de [1] . . . . .	11
2.7. Esquema básico del Periodo . . . . .	12
2.8. Esquema básico de la Frase . . . . .	12
3.1. Representación Gráfica Ciclo de Prototipo Incremental. Figura extraída de [13]. . . . .	20
3.2. Calendario . . . . .	22
3.3. Calendario Estudio del Arte . . . . .	23
3.4. Calendario del reconocimiento de la señal de audio de entrada en <i>Android</i> . . . . .	23
3.5. Calendario del procesamiento de la señal . . . . .	25
3.6. Calendario de la obtención la nota asignada a cada frecuencia . . . . .	26
3.7. Calendario del calculo de la duración de las notas e identificación de los silencios . . . . .	27
3.8. Calendario de la renderización los resultados obtenidos . . . . .	28
3.9. Calendario del la comparación de las partituras . . . . .	28
3.10. Calendario experimentos y pruebas . . . . .	29
4.1. Fases del sistema . . . . .	31
4.2. Diagrama de clases del porcesamiento de la señal . . . . .	33
4.3. Diagrama obtención pico máximo . . . . .	37
4.4. Diagrama de clases de las frecuencias de las notas . . . . .	38
4.5. Diagrama de clases de las duración de las notas . . . . .	40
4.6. Diagrama obtención Lista de notas definitivas . . . . .	41
4.7. Diagrama de la comparación . . . . .	44
5.1. Comprobación número de ventanas . . . . .	47
5.2. Comprobación número de ventanas con otro tamaño . . . . .	47
5.3. Comprobación del pico máximo y su posición . . . . .	48

5.4. Comprobación de los máximos locales y del pico máximo junto con su posición . . .	48
5.5. Salida de la discretización de una frecuencia . . . . .	49
5.6. Salida de la discretización de varias frecuencias . . . . .	50
5.7. Salida de unir varias notas . . . . .	51
5.8. Salida de la duración de cada figura . . . . .	51
5.9. Salida de la figura de cada nota . . . . .	52
5.10. Partitura original comprobación notas y figuras . . . . .	52
5.11. Salida de la lectura de la partitura . . . . .	53
5.12. Partitura original comprobación notas y alturas . . . . .	53
5.13. Salida de la lectura de las alturas de las notas . . . . .	54
5.14. Partitura a comparar con si misma . . . . .	54
5.15. Salida de la comparación de dos partituras iguales . . . . .	54
5.16. Partitura modificada para ser comparada con la original . . . . .	55
5.17. Salida de la comparación de dos partituras con un pequeño cambio . . . . .	55
5.18. Partitura modificada 2 para ser comparada con la original . . . . .	55
5.19. Salida de la comparación de dos partituras con varios cambios . . . . .	56
5.20. Partitura modificada 3 para ser comparada con la original . . . . .	56
5.21. Salida de la comparación de dos partituras y una de ellas con menos notas . . . . .	56

# SIGLAS

---

**F.F.T.** Fast Fourier Transform. 8, 24, 33, 35

**I.S.O.** International Organization for Standardization. 13

**SW** software. 17–19, 21, 32, 37

**T.F.G.** Trabajo Fin de Grado. 1, 3, 5, 21, 57, 58

**T.I.C.** Tecnologías de la información y la comunicación. 2, 13, 22



# INTRODUCCIÓN

---

En este capítulo del proyecto pretendemos contextualizar todas aquellas innovaciones que este Trabajo Fin de Grado (T.F.G.) pretende resolver para ayudar al propósito de este, señalando las motivaciones que nos han llevado a su realización, concretando los objetivos y presentando la estructura que seguiremos para el desarrollo del mismo.

## 1.1. CONTEXTO

La búsqueda y elección de un tema para realizar un Trabajo Fin de Grado, puede resultar algo muy complejo, pues serían tantas las opciones disponibles que costaría tomar una decisión. Sabiendo que son muchas las posibilidades que esos temas ofrecen para investigar y que habría de partida en todos ellos varias ideas que servirían para innovarlos, dando un paso más para complementar conocimientos, no solo a la hora de estudiarlos y aprenderlos, sino también para enseñarlos en un espacio más práctico.

Hablar de innovación, en cualquier tipo de nuevo proyecto, es decir que se da comienzo, en sus primeros pasos, a cualquier nueva idea tecnológica que aparezca en un momento determinado y que al día siguiente de su aparición ya se está mejorando, en un proceso en el que este proyecto después de alcanzar su punto más álgido vislumbrará su fin, por haber sido superada por otra idea mejor y más innovadora, así sucesivamente, y que aunque se desconoce cuándo, ni cómo, esto sucederá.

Una vez puesta en marcha una idea innovadora, que puede tener aplicación en sectores de todo tipo (industrial, comercial, otros proyectos de investigación, etc.) dentro de los avances tecnológicos a nivel global, esto supone una carrera en la que la meta que se pretende alcanzar no es visible, pues se viaja hacia un futuro que no es posible definir, ni imaginar y que en este momento de la historia de la humanidad se realiza a una velocidad no conocida anteriormente, gracias a tantas nuevas ideas que no paran de surgir, generando a la vez una competitividad, que no permite distraerse, ya que poseer la tecnología más puntera supone tener el dominio comercial en los distintos sectores y el control de los mercados de inversión. También decir que todo va en beneficio de los avances de la Ciencia y que además repercute en el sostenimiento del planeta y a la postre, en la calidad de vida de los que lo habitamos.

No se concibe el desarrollo de nuevas ideas y emprender nuevos proyectos, independientemente del sector donde estos se realicen, sin que intervengan las aplicaciones de las tecnologías informáticas en cualquiera de sus áreas.

La música acompaña al hombre desde su existencia. En cualquier estudio antropológico de la humanidad, la música siempre ha estado presente y asociada directamente a su evolución.

Para muchos un arte, para otros una dimensión intangible; en todo caso, al igual que se puede leer y sentir una poesía, se puede leer y sentir una partitura musical sin necesidad de escucharla, basta con conocer la vida de Beethoven y su obra musical. Todo requiere tener conocimientos específicos, dentro del contexto de las distintas materias donde se pretendan aplicar nuevos avances para seguir innovando, sería como decir, comenzar a escribir un libro entre todos, que no tiene fin, en el que se va aportando y sumando nuevos, variados y distintos conocimientos, lo que hace que ese libro sea cada vez más rico en sabiduría, para el uso y disfrute de todos.

Era informática, era tecnológica, mundo de la información, de la comunicación. Actualmente el uso de las T.I.C. está presente en todos y cada uno de los diferentes ámbitos de la vida del ser humano. La mayoría de las personas, hacen un uso directo o indirecto de la tecnología y, por ende, de la informática. Los avances tecnológicos se están produciendo de una forma cada vez rápida en la actualidad; estos avances benefician a los distintos sectores de la sociedad: medicina, comunicación, seguridad, educación y dentro de esta, la educación musical.

La historia del ser humano, siempre ha estado unida a la música. Desde la prehistoria se tiene constancia del uso de instrumentos musicales en rituales y ceremonias, pero no fue hasta la antigua Grecia cuando apareció lo que ahora se conoce como Música Occidental y se crean los primeros tratados de música utilizados para la enseñanza de esta.

Hasta el siglo XVII, la enseñanza de música se desarrollaba principalmente en La Corte e Iglesias. En estos lugares, los músicos escribían tratados que servían para impartir docencia a los miembros de La Corte. Además, los músicos se dedicaban a componer obras con la finalidad de ser interpretadas en las fiestas y funciones religiosas, encargadas por los nobles y el clero. También y no menos importante, el pueblo llano poseía una tradición musical que era transmitida de padres a hijos de una forma oral y que eran usadas en los diferentes actos festivos.

En el siglo XVII, la creatividad musical adquiere un brillo muy importante y extraordinario en distintas partes de Europa y especialmente en Italia. De aquí nace la necesidad, por parte de la Iglesia de instrumentistas y cantantes altamente cualificados para sus grandes ceremonias.

Esta fue una de las principales causas del cambio que se hace en el uso de los conservatorios, donde originalmente se atendían a los niños desamparados, con el fin de proporcionarles un oficio. Se comienza a enseñar, de una manera más estructurada y teniendo en cuenta las cualidades individuales de cada niño, el estudio de un instrumento musical y particularmente del canto, instruyéndolos para participar en las funciones religiosas o para integrarse al servicio de La Corte.

En el resto de países y siguiendo el ejemplo de Italia, comenzaron a formarse músicos en los conservatorios y la didáctica aplicada para ello ha ido evolucionando hasta nuestros días, donde el proceso de enseñanza y aprendizaje musical está al alcance de cualquier persona que tenga inquietud por la música.



## 1.2. MOTIVACIÓN

Para el aprendizaje de la música se puede recurrir a una enseñanza reglada o usar un método autodidacta.

Existen diversos métodos para aprender la teoría de la música y la práctica de un instrumento en concreto. El problema principal que se puede observar es que todos estos tutoriales son pasivos, es decir, la persona que los sigue sólo obtiene la información teórica, sin saber a la hora de poner en práctica todo lo aprendido, si lo ha asimilado de manera correcta.

Por otro lado, como se ha mencionado anteriormente, existen centros específicos para el aprendizaje de música reglada, los conservatorios. En estos centros se ofrecen clases magistrales de enseñanza de la teoría musical y del instrumento.

Sin embargo, hay algunas veces que los alumnos no asimilan de manera correcta los contenidos trabajados en el aula. A la hora de estudiar y repasar el alumnado estos contenidos con el instrumento, de forma individual, pueden surgir errores en el aprendizaje que no son solventados hasta la siguiente sesión en el conservatorio, lo que conlleva un doble trabajo tanto para los docentes como el alumnado, ya que deben subsanar el concepto erróneo y cambiarlo por el correcto.

Teniendo en cuenta estas dos premisas e investigando sobre el tema, se observa la necesidad de desarrollar un sistema que ayude al alumno en el estudio individual de su instrumento, que sirva como apoyo, refuerzo y mejora de su aprendizaje.

Como es un proyecto amplio y ambicioso, será abordado en varias etapas.

## 1.3. OBJETIVOS

### 1.3.1. OBJETIVO GENERAL

El objetivo principal de este T.F.G. es desarrollar una herramienta con potencial aplicabilidad en el proceso de aprendizaje de un instrumento monódico con énfasis en el diseño e implementación de algoritmos para la comparación de interpretaciones.

### 1.3.2. OBJETIVOS PARCIALES

Los objetivos parciales del proyecto son:

- Implementación de un hilo de ejecución en segundo plano para la adquisición de la señal acústica proveniente del micrófono.
- Construir el procedimiento de “enventanado” de la señal acústica de la interpretación grabada y su preprocesamiento.
- Diseño e implementación del algoritmo de *Pitch Tracking* y pruebas de validación.
- Diseño e implementación del algoritmo de demarcación de notas musicales concretas (discretización en semitonos, estimación de duración y segmentación de silencios, estimación de intensidad/volumen de cada nota). Pruebas de validación del algoritmo.
- Diseño e implementación del algoritmo para mapear notas identificadas (y sus características) en notación *MusicXML*.

- Diseño e implementación del algoritmo para comparar las estructuras de árbol que albergan los archivos *MusicXML* (pasaje musical grabado Versus interpretación de referencia del pasaje musical).
- Proporcionar *feedback* al aprendiente de las diferencias encontradas tras la comparación de las interpretaciones de pasajes musicales.

## 1.4. ESTRUCTURA DE LA MEMORIA

A continuación, se detallan los capítulos que componen este documento y dentro de cada uno de estos se lleva a cabo una breve descripción de los mismos.

### Capítulo 2: Fundamentos y Antecedentes

En este segundo capítulo se muestran todos los fundamentos necesarios para la comprensión de este proyecto y también se exponen diferentes trabajos desarrollados en este campo de investigación, se analizan las relaciones entre ambos y se muestran las distintas ventajas e inconvenientes que ofrecen.

### Capítulo 3: Metodología de trabajo

En este capítulo se estudia la metodología de desarrollo que se ha utilizado para ejecutar el trabajo, así como la planificación estimada y su realización, describiendo las iteraciones del desarrollo.

### Capítulo 4: Resultados

Este capítulo, consta de un único apartado en el cual se confrontan los resultados obtenidos en cada iteración y la forma de llegar a estos.

### Capítulo 5: Evaluación

En el capítulo 5, se muestran la evaluación de los algoritmos.

### Capítulo 6: Conclusiones

En este ultimo capítulo, se muestran las conclusiones; se describen las dificultades encontradas durante el desarrollo y se presentan las propuestas de mejora del proyecto y de futuros trabajos.

# FUNDAMENTOS Y ANTECEDENTES

En el segundo capítulo se introducen todos aquellos fundamentos básicos tanto musicales como conceptos de la física aplicada a este campo, necesarios para comprender y abordar la solución al problema planteado. Además, se muestran distintas líneas de investigación respecto al tema principal.

## 2.1. FUNDAMENTOS FISICOS

A lo largo de esta sección se explicarán los conceptos que la física aporta a esta disciplina, para poder comprender los contenidos que pretende este T.F.G. y así tener una visión y un conocimiento más amplio de los mismos en próximos capítulos.

### 2.1.1. EL SONIDO

El fundamento natural de la música es el sonido. **El sonido** se produce cuando una fuente sonora vibra. Dicha vibración es un movimiento de vaivén que obligará a moverse de la misma manera a las moléculas de aire que están próximas a la fuente. Esa oscilación, juntará o separará las moléculas, creando zonas de compresión y de depresión que a su vez empujarán a otras moléculas contiguas produciéndose ondas sonoras [14].

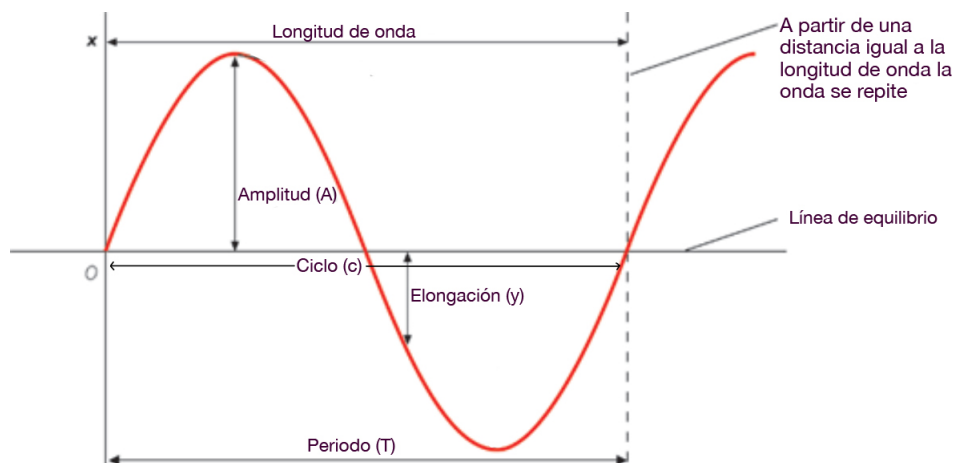


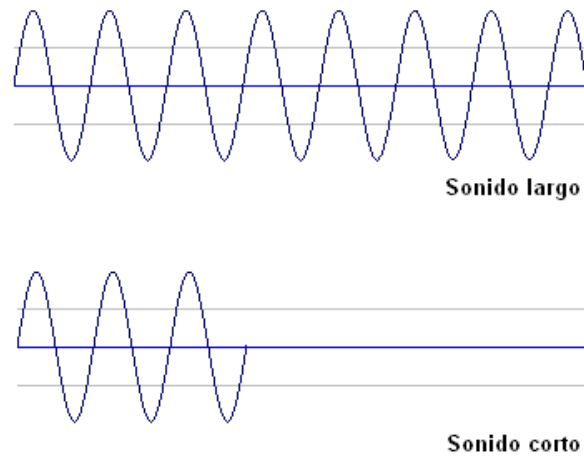
Figura 2.1: Representación del sonido con sus parámetros

Las Ondas Sonoras son definidas por la siguiente ecuación diferencial:

$$p(r, t) = p_0 + \frac{\Delta p}{r} \sin(2\pi vt - 2\pi \frac{r}{\lambda} + \phi_0) \quad (2.1)$$

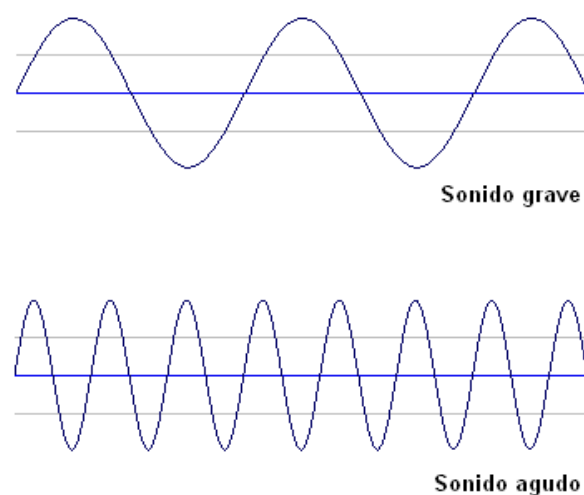
Las cualidades del sonido son las siguientes [14]:

- **Duración:** Se podría decir que la duración de una sensación sonora depende directamente de la duración del movimiento vibratorio que origina el sonido.



**Figura 2.2:** Representación de la duración del sonido

- **Volumen:** El volumen es una sensación que aumenta al incrementar la amplitud y disminuye al crecer la frecuencia.
- **Densidad:** Definimos densidad como la sensación que producen ciertos sonidos de ser más compactos, más llenos que otros.
- **Altura:** La altura de un sonido está determinada por la frecuencia de las vibraciones de un cuerpo sonoro, es decir, su carácter de Grave o Agudo. La altura depende del número de vibraciones por segundo que posee un sonido.



**Figura 2.3:** Representación de la altura del sonido

- **Timbre:** El Timbre define la diferencia en el color tonal de una nota tocada por diversos instrumentos o cantada por diferentes voces. El timbre depende de la forma de la onda sonora. El Sonido Puro (la vibración sinusoidal) sólo se puede conseguir electrónicamente o mediante el Diapasón.

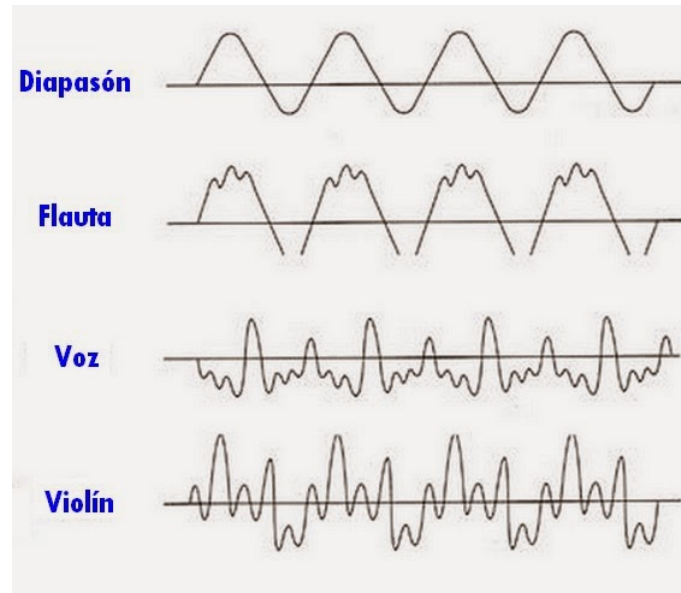


Figura 2.4: Representación de distintos tipos de timbres

- **Intensidad:** Es la cualidad del sonido que permite distinguir uno fuerte de otro débil y viceversa.

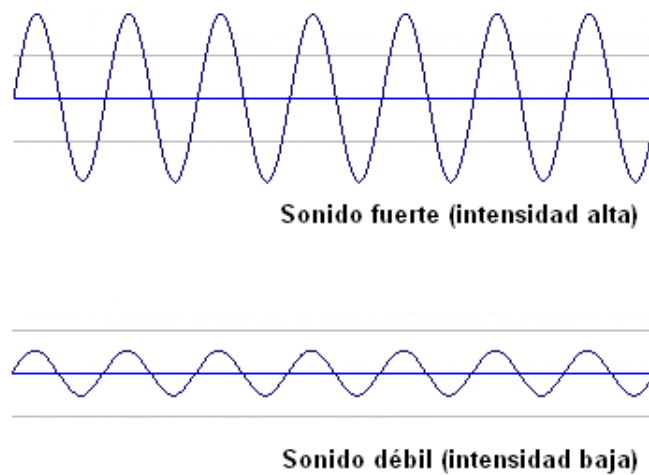


Figura 2.5: Representación de la intensidad del sonido

### 2.1.2. PITCH TRACKING

El *Pitch Tracking* es un algoritmo diseñado para estimar la frecuencia fundamental de una señal oscilante, generalmente de un tono musical. Dicho algoritmo tiene diferentes fases en las cuales se trata la señal para obtener las frecuencias. Esas fases son:

### ■ TRANSFORMACIÓN DE LA SEÑAL

Para procesar la señal, primero debemos transformarla. Para ello, usamos la *Fast Fourier Transform (F.F.T.)*, que cambia una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo [8]:

$$F\left(\frac{n}{NT}\right) = \frac{1}{N} \sum_{k=0}^{N-1} m(kT) e^{-j \frac{2\pi nk}{N}} \quad (2.2)$$

Donde **N** es el número de muestras de la ventana que se va a analizar, **T** es el periodo de muestreo (inverso a la frecuencia de muestreo que denominaremos 'f'), **n** es el Índice de la frecuencia cuyo valor queremos obtener y **m(kT)**, indica la muestra tomada en el instante **kT** (muestra Késima) de la ventana.

El valor del parámetro **n** determina la frecuencia concreta que se va a analizar, es decir, representa una de las frecuencias en las que se va a tratar de descomponer la señal de partida, de esta manera, para hacer el estudio con todas las frecuencias, usamos el rango completo de variación de **n=0, J, 2, ..., N-J**. Desarrollando la fórmula de la *Fast Fourier Transform* para los distintos valores de **n**, tenemos:

$$n = 0 \Rightarrow F(0) = \frac{1}{N} \sum_{k=0}^{N-1} m(kT) e^0 \quad (2.3)$$

$$n = 1 \Rightarrow F\left(\frac{1}{N}f\right) = \frac{1}{N} \sum_{k=0}^{N-1} m(kT) e^{-j2\pi \frac{k}{N}} \quad (2.4)$$

$$n = N - 1 \Rightarrow F\left(\frac{N-1}{N}f\right) = \frac{1}{N} \sum_{k=0}^{N-1} m(kT) e^{-j2(N-1)\pi \frac{k}{N}} \quad (2.5)$$

Con las ecuaciones anteriores se obtienen las siguientes nociones:

- La porción de la señal que se analiza se encuentra en el bloque de muestras **m(O), m(J), m(2), ..., m(N-J)**
- La frecuencia **f=0Hz**. (correspondiente a **n=0**) se halla haciendo la media aritmética de los valores de las muestras, por lo que representa la componente continua de la señal.
- El parámetro **n** actúa de índice para obtener las distintas frecuencias de estudio.
- Los valores que se obtienen para **0 ≤ n < N/2** coinciden con los obtenidos en el intervalo **N/2 ≤ n < N-J**, (con **n** par) por lo que es suficiente realizar los cálculos en una de las dos mitades. Según el *criterio de Nyquist*, el ancho de banda de la señal coincide con la mitad de la frecuencia de muestreo **j**, correspondiente a **n=0, J, 2, ..., (N/2)-J**.
- Los valores **n=J → 2n, n=2 → 4r, ..., n=N-J → 2(NJ)n**, indican las frecuencias de las señales sinusoidales y cosenoidales con las que se comparará la señal original, este concepto se explicará a lo largo del apartado.
- Si aumentamos el valor de **N**, conseguimos hacer el análisis con un mayor número de frecuencias (**0 ≤ n ≤ N-J**), pero a costa de un mayor tiempo para calcular las operaciones del sumatorio (**0 ≤ n ≤ N-J**)

## ■ AUTOCORRELACIÓN

Las funciones de autocorrelación son una herramienta útil en el tratamiento de las señales cuasi periódicas. La función de autocorrelación para una señal determinística en tiempo discreto se define usualmente como:

$$\phi(K) = \sum_{m=-\infty}^{\infty} x(m)x(m+k) \quad (2.6)$$

La definición apropiada para una señal aleatoria o periódica de la función de autocorrelación es:

$$\phi(K) = \lim_{N \rightarrow \infty} \left( \frac{1}{2N+1} \right) \sum_{m=-N}^N x(m)x(m+k) \quad (2.7)$$

De cualquier modo, que se defina la función de autocorrelación, la representación de la señal de esta función permite mostrar ciertas propiedades de la señal. Las propiedades más importantes de la función de autocorrelación son las siguientes:

- Si la señal es periódica con periodo de P muestras, entonces es fácil demostrar que para todo K, es decir, la función de auto correlación de una señal periódica es también periódica y con el mismo periodo.
- La función de auto correlación es una función par, es decir, para todo k.
- La función de auto correlación toma su valor máximo en k=0.
- El valor (0) es igual a la energía en el caso de señales determinística, o la potencia media para señales periódicas o aleatorias.

(...)De acuerdo con la definición de función de autocorrelación para una señal determinística en tiempo discreto se define esta en tiempo corto como:

$$R_n(K) = \sum_{m=-\infty}^{\infty} x(m)x(n-m)x(m+n)w(n-k-m) \quad (2.8)$$

Esta ecuación puede interpretarse como que cada segmento de señal de habla se multiplica por una ventana y después se aplica la definición de función de autocorrelación para señal determinística.

La función de autocorrelación en tiempo corto es una función par. Se puede considerar como un caso particular de una transformación en tiempo corto. Para ello, simplemente se escribe:

$$R_n(-K) = R_n(K) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)h_k(n-m) \quad (2.9)$$

Donde se define,

$$R_h(n) = w(n)w(n+k)$$

Dicho de otra forma, la función de autocorrelación en tiempo corto con salto de k muestras se obtiene filtrando la sucesión  $x(n) x(n-k)$  con un filtro de respuesta impulso  $h_k(n)$ .

La función de autocorrelación en tiempo corto puede reescribirse utilizando la ventana  $w'(n)=w(-n)$ , con lo cual se obtiene la fórmula:

$$R_n(K) = \sum_{m=-\infty}^{\infty} [x(n+m)w'(m)][x(m+n+k)w'(k+m)] \quad (2.10)$$

En esta expresión, el tiempo inicial de la sucesión de entrada se ha desplazado a la muestra n, entonces se ha multiplicado por la ventana w para seleccionar un segmento corto de habla. si

la ventana  $w$  es de longitud finita, entonces la función de autocorrelación en tiempo corto se puede representar por:

$$R_n(K) = \sum_{m=0}^{N-1-K} [x(n+m)w'(m)][x(m+n+k)w'(k+m)] \quad (2.11)$$

### ■ NORMALIZACIÓN

En estadística, normalizar es cambiar una distribución de ciertos datos a una curva normal o Gaussiana donde el promedio y la mediana coinciden en el mismo punto. En base a lo anterior, normalizar una señal es ajustar dicha señal a una curva normal. Cuando los datos tienen una distribución normal, es posible obtener mayor información de ellos que cuando estos datos presentan otro tipo de distribución, pues se cumple que los estimadores de dicha distribución tienen propiedades que se acercan más a la verdad.

Una de las técnicas más usadas de normalización es la *Normalización de Pico* en el que la ganancia se cambia para llevar el pico más alto de una señal.

## 2.2. FUNDAMENTOS MUSICALES

### 2.2.1. MODOS BÁSICOS DE ORGANIZACIÓN TEMPORAL

Para comprender los modos básicos que se utilizan en la organización temporal, previamente debemos delimitar los conceptos que intervienen en esta organización:

- **Pulso:** El pulso es cada estímulo idéntico e isócrono que se repite en una serie regular. Marca unidades iguales en el continuum temporal. En las poéticas palabras de Mathis Lussy:

*“En el tiempo, el hombre...clava sus jalones a fin de interrumpir la continuidad, la uniformidad, establecer puntos de apoyo para satisfacer sus necesidades rítmicas. Así nace la música y la poesía. Estos puntos de apoyo, estos límites son intangibles; son instantes fisiológicos y es su organismo, es su respiración la que los suministra. El ritmo es para la música lo que la simetría es para la arquitectura; es la división, la ruptura regular en la continuidad del tiempo, como la simetría es la división regular del espacio.”*<sup>[15]</sup>









Aunque el pulso se establece y sostiene mediante estímulos sensoriales objetivos, puede sentirse interiormente en la mente y el cuerpo, aun cuando el sonido haya desaparecido. Aunque por definición los pulsos han de ser exactamente iguales, la mente humana superpone inconscientemente una organización. Las más elementales son la organización binaria y la ternaria.

- **Compás:** El compás es la medida del número de pulsos existentes entre los acentos que van apareciendo con una recurrencia más o menos regular<sup>[4]</sup>.

Como vemos se trata de un concepto asociado a la medida y por lo tanto es la métrica quien se encarga de los compases. El significado de ritmo es más amplio, engloba y rebasa al de compás. Sabemos, por ejemplo, que puede existir ritmo sin compás, como en el “ritmo libre” de algunos estilos del flamenco o del canto gregoriano.

Del mismo modo que no debemos confundir el ritmo con la métrica (el compás) también hemos de ser cuidadosos para no caer en el error de pensar que toda la organización temporal en la música está contenida en la barra de compás.



COMPASES			
Indicador	Unidad de tiempo	Duración del compás	Acentuación
2 4			
3 4			
4 4			
2 2			
6 8			

**Figura 2.6:** Diferentes tipos de compases comunes. Figura extraída de [1]

- **Ritmo:** El ritmo es un concepto muy amplio que sobrepasa incluso los límites de lo musical. No obstante, a este nivel, puede definirse como el modo en que una o más partes no acentuadas son agrupadas en relación con otra parte que si lo está. El ritmo es independiente del compás en dos sentidos.

Por una parte, puede existir ritmo aun cuando no haya un compás regular, como ocurre en el canto llano, en el recitativo secco o en los cantos libres del flamenco. Por otra parte, como el ritmo se manifiesta en los distintos niveles estructurales, pueden darse agrupamientos rítmicos no relacionados con el compás. Así una frase puede presentar un agrupamiento yámbico independientemente de que el compás sea binario o ternario.

### 2.2.2. FORMA MUSICAL

La forma musical presupone el acto generador de dar forma. Sólo el modelado consciente transforma una serie de notas en los más diversos tipos de manifestaciones inteligibles, crea relaciones entre partes o hace que se enfrenten ásperamente; en distintos momentos y de diferentes maneras, ese modelado apunta a una relación y a una coherencia tanto de los detalles musicales como del conjunto. Porque el tipo de conformación depende de la noción que se tenga de la forma: en las diversas transformaciones de un compositor, la forma musical representa modos históricos de pensamiento. Ambos factores son igualmente importantes: la propia idea formal, que se revela tanto en las ramificaciones internas como en la manufactura exterior de una obra, y la transformación histórica de esa idea [10].

Partiendo de la definición anterior, existen diferentes tipos de desarrollar una idea musical. Las más comunes son:

- **Fortspinnung:** Es un empujar constantemente hacia delante. Consta de tres partes bien diferenciadas:
  - Antecedente: se basa fundamentalmente por tener una precisión motívica y un ritmo armónico estable.
  - Fortspinnung: engancha de una manera sutil con el antecedente y usa los motivos para hacer el efecto de un empuje hacia delante.

- **Conclusión:** desarrolla el Fortspinnung y lo conduce hacia una cadencia bien estructurada y adornada.
- **Periodo:** Las relaciones armónicas de una idea llevan a terminar el pasaje musical en la tónica o primer grado de la tonalidad. El periodo está formado por dos partes: un antecedente, que presenta un tema principal y un consecuente, que hace una variación del tema original y lo termina.



Figura 2.7: Esquema básico del Periodo

- **Frase:** Al contrario que el Periodo, las relaciones armónicas de la idea llevan a terminar el pasaje musical en la dominante o quinto grado de la tonalidad. Las partes de la frases son las mismas que en el Periodo (antecedente y consecuente), pero en este caso, el consecuente tiene la función de llevar el tema del antecedente a otro nuevo tema, haciendo la función de puente entre un tema y otro.

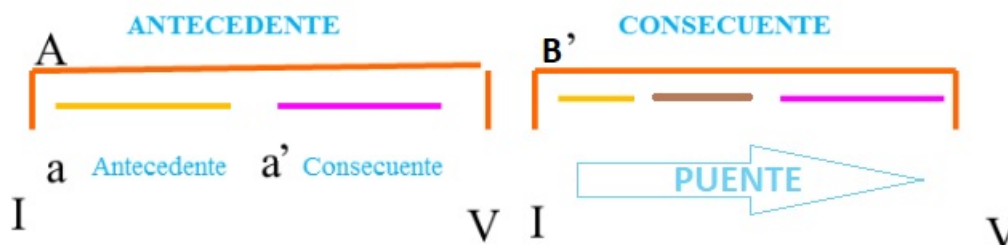


Figura 2.8: Esquema básico de la Frase

### 2.2.3. SISTEMA DE AFINACIÓN TEMPERADO

En la antigua Grecia, diversos matemáticos ya sabían la relación interválicas entre los sonidos, pero al intentar respetar las dos consonancias principales (8° y 5°) entre todas las notas, creaban una incompatibilidad entre otras consonancias (Sistema Pitagórico).

A lo largo de la historia, aparecieron otros sistemas de afinación para subsanar el problema del Pitagórico, algunos basados en la consonancia de 3° (Sistema Mesotónico), otros, en unidades interválicas más pequeñas que el semitono (Sistema Holder).

En el siglo XVI, surgió el sistema temperado que es una evolución del sistema de Pitágoras, ya al hacer este reparto en fracciones iguales, cada una de las quintas del círculo pitagórico resulta reducida en un doceavo de coma (2.12).

$$F_{n+1} = F_n 2^{\frac{1}{12}} \quad (2.12)$$

La comparación de los dos sistemas de afinación se muestra en la siguiente tabla:

Nota	ST	Intervalo	A. Pitagórica	A. Temperada
DO	0	Unisono	1	1
REb	1	Semitono	$2187/2048 = 1,06787$	$\sqrt[12]{2} = 1,05946$
RE	2	2ª Mayor	$9/8 = 1,125$	$\sqrt[12]{2^2} = 1,12246$
MIb	3	3ª menor	$32/27 = 1,18519$	$\sqrt[12]{2^3} = 1,18921$
MI	4	3ª Mayor	$81/64 = 1,26562$	$\sqrt[12]{2^4} = 1,25992$
FA	5	4ª Justa	$4/3 = 1,3334$	$\sqrt[12]{2^5} = 1,33484$
SOLb	6	4ª Aumentada	$729/512 = 1,51923$	$\sqrt[12]{2^6} = 1,41421$
SOL	7	5ª Justa	$3/2 = 1,5$	$\sqrt[12]{2^7} = 1,49831$
LAB	8	5ª Aumentada	$6561/4096 = 1,60181$	$\sqrt[12]{2^8} = 1,58740$
LA	9	6ª Mayor	$27/16 = 1,60181$	$\sqrt[12]{2^9} = 1,68179$
SIb	10	7ª menor	$16/9 = 1,7778$	$\sqrt[12]{2^{10}} = 1,78179$
SI	11	7ª Mayor	$243/128 = 1,89844$	$\sqrt[12]{2^{11}} = 1,88775$
DO	12	8ª Justa	2	2

**Tabla 2.1:** Comparación de los sistemas de afinación. Figura extraída de [3]

## 2.2.4. CATEGORIZACIÓN DE LAS NOTAS MUSICALES

Como se ha referido anteriormente, la evolución de los sistemas de afinación ha llevado a la variación de las frecuencias impuestas a cada nota. Hoy y para tener un consenso entre todos los instrumentos y músicos, se tiene como referencia desde 1975 el I.S.O. 16 [7], poniendo la frecuencia del La en 440 Hercios.

## 2.3. ANTECEDENTES

La sociedad actual es testigo del cambio que se está produciendo en todos sus ámbitos. El uso de las T.I.C. es cada vez más relevante en la vida del ser humano. La utilización de ordenadores, móviles, tablets,.. se ha integrado plenamente en todos los ámbitos sociales. El ámbito educativo no puede quedarse al margen, aprovechando los beneficios que las T.I.C. aportan al proceso de enseñanza y aprendizaje del alumnado, lo podemos comprobar cómo ordenadores, pizarras digitales interactivas, etc. han sido introducidos en las aulas.

### 2.3.1. INFORMÁTICA Y MÚSICA

La enseñanza de la música relacionada con las nuevas tecnologías se enfrenta a dos desafíos: el primero integrar todo los avances que ha supuesto la introducción de la tecnología electrónica e informática en la música. El segundo desafío esta en los medios y en los métodos.

La primera vez que se empezó a usar el ordenador para le enseñanza de la música fue en 1970. Posteriormente, con la aparición de los ordenadores personales se empezaron a desarrollar nuevas estrategias para el aprendizaje:

- **Aprendizaje programado:** es la técnica más básica, ya que equivale a lo que actualmente conocemos como libro electrónico. En este tipo de aprendizaje, el sistema mostraba a través de la pantalla los conceptos de una manera guiada y gradual.
- **Aprendizaje interactivo:** como su propio nombre indica, en este tipo de aprendizaje se establece un diálogo entre el alumno y el ordenador. El programa que utiliza este método puede alterar el flujo de aprendizaje dependiendo de las respuestas que el alumno ofrezca. Para desarrollar este tipo de aprendizaje existen dos técnicas:
  - Técnica drill: Esta técnica se apoya en la proposición de un examen al alumnado por parte del sistema, y dependiendo de la calificación que se obtenga, se supera el nivel y se pasa al siguiente. Si no lo domina permanece en el nivel y se le proponen diferentes ejercicios de refuerzo y repaso para la superación de éste.
  - Técnica Tutorial: Es también conocida como clase particular. Consiste en presentar el sistema como un libro, pero mucho más flexible. Este método contiene teoría, ejemplos, partituras. . . , Con esta práctica el alumno puede manejar la información sobre el tema en el orden que desee, permitiendo retomar cualquier concepto.

Las aplicaciones desarrolladas para la enseñanza de música son principalmente las utilizadas para ejercitar del oído y practicar el solfeo. La técnica más común es “la técnica drill”, cuyo fundamento está basado en la presentación de los conceptos, y posteriormente desarrollar distintos exámenes con diferentes niveles para conocer la asimilación de estos.

Para el estudio de un instrumento, se emplea “la técnica tutorial” junto con “la técnica drill”, destacando los gráficos y los videos que utilizan para mostrar al alumno las diferentes técnicas, las digitaciones, posiciones, ritmo, fraseo. . .

### 2.3.2. APLICACIONES ACTUALES

Podemos encontrar en el mercado un gran número de programas y aplicaciones relacionadas con la música, entre los que destacamos:

- **Programas de edición de partituras:** Estos programas están diseñados para escribir música. Los más conocidos son:
  - Finale: Es el veterano de todos. Posee una elegante interfaz y nos permite un gran control sobre todos los aspectos de la creación de partituras incluyendo una completa paleta de herramientas de edición.
    - **Problema:** Finale es difícil de manejar para inexpertos dado que son tantas las posibilidades que ofrece, que puede llegar a desmotivar al usuario con todas las variantes de teclado y herramientas.
  - Sibelius: Es un programa mucho más fácil de manejar que Finale. Se utiliza para iniciarse en la escritura y está enfocado al campo didáctico.
    - **Problema:** Sibelius no ofrece tantas posibilidades como Finale.
  - Lilypond: Es el único editor que está disponible para UNIX.
    - **Problema:** No posee interfaz gráfica, sino que toda la edición se controla desde un editor de textos donde introducen los comandos necesarios para crear la partitura.
- **Programas de Aprendizaje:** Son aplicaciones que permiten estudiar la teoría musical y aprender un instrumento. Las más utilizadas actualmente son:

- Simply Piano: Es una aplicación que enseña al usuario a aprender a tocar el piano, empleando la técnica drill.
  - **Problema**: La aplicación está centrada en canciones específicas, además se desconoce hasta que nivel puede llegar a ser eficaz con otros tipos de instrumentos que no sean de tecla.
- Violy - Smart Violin Partner: Aplicación que detecta en el alumno de violín los fallos ha cometido y evalúa su interpretación.
  - **Problema**: al principio te pide permisos en el dispositivos que no son necesarios para la aplicación, como acceso a tus contactos. Luego la aplicación no es capaz de reconocer la partitura y mucho menos compararla con lo que has tocado.
- Music Tutor Sight Read: Aplicación dirigida para enseñar los fundamentos básicos musicales. Usa la técnica Tutorial.
  - **Problema**: aplicación solo útil para los primeros conceptos musicales.

### 2.3.3. LÍNEAS DE INVESTIGACIÓN ACTUALES

Actualmente, encontramos distintas líneas de investigación sobre el *Pitch Tracking* y el uso de la música y la informática, una de la más reciente es la de Liming Shi [16].

Esta investigación propone un algoritmo de seguimiento de frecuencia fundamental totalmente bayesiano basado en el modelo armónico y un modelo de proceso de Markov de primer orden y las conclusiones sacadas son que usando estos modelos de Markov, la estimación de frecuencia fundamental y los errores de detección de voz pueden reducirse.

La investigación de Emir Demirel [6], se centra en la aplicación de diferentes métodos computacionales para llevar a cabo un análisis armónico modal mediante el modelado del concepto de escalas de acordes.

Además, este trabajo propone diferentes enfoques computacionales para el reconocimiento del tipo de escala de acordes en una frase improvisada dado el contexto armónico.

Siguiendo esta misma línea, debemos mencionar la investigación de Anders Friberg [2] en donde se expone la creación de una red neuronal convolucional que utiliza la entrada de un sistema de estimación de tono polifónico para predecir la modalidad menor / mayor percibida en el audio de la música.

La entrada de activación de tono está estructurada para permitir que la primera capa de la red neuronal calcule dos cromas de tono enfocados en diferentes octavas. Las siguientes capas realizan análisis de armonía a través de escalas de tiempo y croma.



# METODOLOGÍA DE TRABAJO

---

En este capítulo se describe la metodología utilizada para poder alcanzar los objetivos propuestos.

## 3.1. METODOLOGÍAS

Utilizar todos los conocimientos y herramientas de una tecnología determinada no implica de partida el éxito del producto que se quiere obtener, para conseguirlo y obtener la máxima calidad del mismo, necesitamos estudiarlo con unos objetivos establecidos y utilizar una metodología de desarrollo adecuada.

En las dos últimas décadas las notaciones de modelado y posteriormente las herramientas pretendieron ser lo más sobresaliente para el desarrollo de *software* (SW), sin embargo, las expectativas no fueron satisfechas. Esto se debe en gran parte a que otro importante elemento, la metodología de desarrollo, había sido postergado. De nada sirven buenas notaciones y herramientas si no se proveen directivas para su aplicación.

Así, esta década ha comenzado con un creciente interés en metodologías de desarrollo. Hasta hace poco el proceso de desarrollo llevaba asociada un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada.

Este esquema "tradicional" para abordar el desarrollo de SW ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del "buen hacer" de la ingeniería del SW, asumiendo el riesgo que ello conlleva. En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto

Las metodologías ágiles son sin duda uno de los temas recientes en ingeniería de SW que están acaparando gran interés. Prueba de ello es que se están haciendo un espacio destacado en la mayoría de conferencias y workshops celebrados en los últimos años. Es tal su impacto que actualmente existen 4 conferencias internacionales de alto nivel y específicas sobre el tema. Además ya es un área con cabida en prestigiosas revistas internacionales. En la comunidad de la ingeniería del SW, se está

viviendo con intensidad un debate abierto entre los partidarios de las metodologías tradicionales (referidas peyorativamente como "metodologías pesadas") y aquellos que apoyan las ideas emanadas del "Manifiesto Ágil".

La curiosidad que siente la mayor parte de ingenieros de SW, profesores, e incluso alumnos, sobre las metodologías ágiles hace prever una fuerte proyección industrial. Por un lado, para muchos equipos de desarrollo el uso de metodologías tradicionales les resulta muy lejano a su forma de trabajo actual considerando las dificultades de su introducción e inversión asociada en formación y herramientas. Por otro, las características de los proyectos para los cuales las metodologías ágiles han sido especialmente pensadas se ajustan a un amplio rango de proyectos industriales de desarrollo de SW; aquellos en los cuales los equipos de desarrollo son pequeños, con plazos reducidos, requisitos volátiles, y/o basados en nuevas tecnologías[11].

### 3.1.1. METODOLOGÍAS ÁGILES

Las metodologías ágiles son un tipo de metodología de trabajo que comparten unos principios y valores, que permite desarrollar SW rápidamente y responder a cualquier cambio que pueda surgir a lo largo del proyecto.

En febrero de 2001, en una reunión en Utah-EEUU aparece por primera vez el término "ágil". En dicha reunión se creó *The Agile Alliance* y participaron 17 expertos de la industria SW en la creación.

*The Agile Alliance* es una organización, sin ánimo de lucro, fundamentada en el fin de promover los principios del desarrollo ágil ayudando a las organizaciones para adaptar dichos conceptos.

#### El Manifiesto Ágil

Para exponer al mundo todo lo tratado en la reunión, los integrantes de esta crearon un manifiesto conocido como Manifiesto Ágil, en los que se exponen todos los principios sobre los que están basado este método alternativos en cuatro postulados [12]:

- *Individuos e interacciones sobre procesos y herramientas.*
- *software funcionando sobre documentación extensiva.*
- *Colaboración con el cliente sobre negociación contractual.*
- *Respuesta ante el cambio sobre seguir un plan.*

Aunque el Manifiesto Ágil da más prioridad a los conceptos de la izquierda no implica que se descuiden los de la derecha.

#### Principios del Manifiesto Ágil

Los valores anteriormente descritos inspiran los doce principios del manifiesto. Son las principales características que diferencian el proceso de desarrollo ágil de los tradicionales [12]:

- I *Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.*
- II *Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*



- III *Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
- IV *Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.*
- V *Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.*
- VI *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
- VII *El software funcionando es la medida principal de progreso.*
- VIII *Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.*
- IX *La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.*
- X *La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
- XI *Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*
- XII *A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.*

### 3.1.2. METODOLOGÍAS ÁGILES VS METODOLOGÍAS TRADICIONALES

Aunque durante todo este punto se ha desarrollado profundamente todos los beneficios de las metodologías ágiles, hay que mencionar que no es ni mejor ni peor que las tradicionales, solamente que una metodología es mas eficaz en un proyecto que en otro:

- Usamos metodologías tradicionales en proyectos con un problema conocido y una solución al mismo bien definida. En este entorno es fácil analizar, diseñar y ejecutar una solución. Es ideal si sabemos que los requisitos que se establecen al principio del proyecto no van a cambiar. Esto puede aplicar en proyectos cortos donde el riesgo sea más limitado.
- En un entorno muy cambiante donde no está claro el problema a solucionar, ni la forma de hacerlo son más eficaces las metodologías ágiles ya que incorporan mecanismos de gestión del cambio que implican un menor esfuerzo [5].

### 3.1.3. ADECUACIÓN DEL DESARROLLO ÁGIL AL PROYECTO

La elección de una metodología ágil para este proyecto se debe al hecho de que al dividir el problema principal en pequeñas iteraciones, éste se hace más flexible a los cambios que puedan surgir durante el desarrollo.

Otras de las razones es que, con este tipo de metodologías se detecta los errores de una manera más rápida, la razón es que se hace revisión de manera más frecuente.



Cada etapa corresponde con una iteración, y en cada iteración sigue los pasos de la Figura 3.1, que son:

- **Requisito/Modulo *software*:** cada requisito está marcado por un objetivo parcial tal y como se puede observar en el primer capítulo.
- **Diseño del módulo:** cada iteración tiene un diseño concreto que debe ser pensado para que sea escalable para próximos módulos.
- **Codificación:** es dar vida a lo diseñado anteriormente.
- **Validación:** comprobar que lo codificado anteriormente funciona de forma correcta.

Las tres últimas etapas se repiten para cada una de las iteraciones. Por el modo de operar de esta metodología y por la dependencia de cada iteración con la anterior, es la metodología idónea para el tipo de proyecto que se está llevando a cabo.

### 3.4. PLANIFICACIÓN

A continuación se describirá brevemente cada una de las iteraciones, así como los objetivos más importantes propuestos para cada una de ellas.

#### 3.4.1. PLANIFICACIÓN GENERAL

Para completar el desarrollo se han definido cuatro actividades generales, respondiendo a las funcionalidades que el sistema debe ofrecer. La actividad de desarrollo cuenta con una mayor carga de trabajo, es por ello que se ha dividido en subtarefas, cada una de las cuales hace referencia a los distintos módulos SW requeridos para su funcionamiento, los cuales funcionarán de forma independiente y podrán ser desarrollados de forma prototipada.

- **Estado del arte:** Se realiza una búsqueda de toda la información y documentación relevante acerca de los fundamentos teóricos y tecnológicos que son necesarios para el desarrollo de este T.F.G. y búsqueda de otros trabajos o líneas de investigaciones anteriores y/o actuales que sirvan como antecedentes del mismo.
- **Desarrollo de la aplicación:** El desarrollo de la aplicación, como se ha mencionado antes se hará en diversas subtarefas, que son:
  - Grabación del audio
  - *Pitch Tracking*
  - Discretización de las frecuencias
  - Leer archivo *MusicXML*
  - Comparar archivo *MusicXML* y dar *feedback*
- **Experimentos y pruebas:** Se realizan sesiones de pruebas para comprobar la funcionalidad del sistema y su efectividad con casos reales.
- **Memoria del T.F.G.:** Se recopilan y analizan toda la documentación generada durante el desarrollo.

### 3.4.2. CALENDARIO DEL PROYECTO

En el siguiente calendario se puede observar la duración temporal del proyecto. Se señala en rojo coincide con la Semana Santa, periodos de exámenes ordinarios y periodo de exámenes extraordinarios respectivamente.

A continuación las actividades definidas en la planificación global son detalladas y divididas en iteraciones, dando lugar a la planificación local de cada actividad.

#### Calendario 2019

Febrero							Marzo							Abril						
Lu	Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
				1	2	3					1	2	3	1	2	3	4	5	6	7
4	5	6	7	8	9	10	4	5	6	7	8	9	10	8	9	10	11	12	13	14
11	12	13	14	15	16	17	11	12	13	14	15	16	17	15	16	17	18	19	20	21
18	19	20	21	22	23	24	18	19	20	21	22	23	24	22	23	24	25	26	27	28
25	26	27	28				25	26	27	28	29	30	31	29	30					

Mayo							Junio							Julio						
Lu	Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
			1	2	3	4						1	2	1	2	3	4	5	6	7
6	7	8	9	10	11	12	3	4	5	6	7	8	9	8	9	10	11	12	13	14
13	14	15	16	17	18	19	10	11	12	13	14	15	16	15	16	17	18	19	20	21
20	21	22	23	24	25	26	17	18	19	20	21	22	23	22	23	24	25	26	27	28
27	28	29	30	31			24	25	26	27	28	29	30	29	30	31				

Figura 3.2: Calendario

### 3.4.3. PLANIFICACIÓN LOCAL: ESTADO DEL ARTE

#### Iteración EA1:

Se realiza un estudio e investigación exhaustiva acerca de los conceptos físicos, musicales y tecnológicos necesarios para implementar la aplicación. Se recopila la información más relevante para edificar los fundamentos del proyecto.

Tiempos estimados:

- Obtención de información acerca de fundamentos teóricos: 35 horas.
- Recopilación y análisis de fundamentos teóricos: 20 horas

#### Iteración EA2:

Se lleva a cabo una búsqueda de aplicaciones y líneas de investigación que se han centrado en el procesamiento de señales y aplicar las T.I.C. en la enseñanza de la música. Se recopila la información de los casos más relevantes para su análisis.

Tiempos estimados:

- Obtención de información sobre aplicaciones y líneas de investigación previas: 25 horas
- Recopilación y análisis de los casos más relevantes: 17 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>ESTUDIO DEL ARTE</b>	<b>10 días</b>	<b>vie 01/02/19</b>	<b>jue 14/02/19</b>
<b>EA1</b>	<b>6,88 días</b>	<b>vie 01/02/19</b>	<b>lun 11/02/19</b>
Obtención de información	35 horas	vie 01/02/19	jue 07/02/19
Recopilación y análisis	20 horas	jue 07/02/19	lun 11/02/19
<b>EA2</b>	<b>5,13 días</b>	<b>lun 11/02/19</b>	<b>lun 18/02/19</b>
Obtención de información	25 horas	lun 11/02/19	jue 14/02/19
Recopilación y análisis	16 horas	vie 15/02/19	lun 18/02/19

Figura 3.3: Calendario Estudio del Arte

### 3.4.4. PLANIFICACIÓN LOCAL: APLICACIÓN

#### GRABACIÓN DEL AUDIO

##### *Reconocimiento de la señal de audio de entrada en Android*

##### **Iteración A1:**

En la primera iteración se lleva a cabo un proceso de investigación de las librerías Dart en general así como de las librerías necesarias para realizar la captura de audio en *Android*.

Tiempos estimados:

- Obtención de información sobre librerías Dart: 15 horas
- Implementación de pruebas básicas: 12 horas
- Realización de pruebas: 9 horas

##### **Iteración A2:**

En esta iteración se implementa un subservicio en *Android* para recuperar el audio obtenido mediante el micrófono y guardarlo en una Lista.

Tiempos estimados:

- Implementación del subservicio *Android*: 15 horas
- Realización de pruebas: 10 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>Reconocimiento de la señal de audio de entrada en Android</b>	<b>7,63 días</b>	<b>mar 19/02/19</b>	<b>jue 28/02/19</b>
<b>A1</b>	<b>4,5 días</b>	<b>mar 19/02/19</b>	<b>lun 25/02/19</b>
Obtención de información sobre librerías Dart	15 horas	mar 19/02/19	mié 20/02/19
Implementación de pruebas básicas	12 horas	mié 20/02/19	vie 22/02/19
Realización de pruebas	9 horas	vie 22/02/19	lun 25/02/19
<b>A2</b>	<b>3,13 días</b>	<b>lun 25/02/19</b>	<b>jue 28/02/19</b>
Implementación del subservicio Android	15 horas	lun 25/02/19	mié 27/02/19
Realización de pruebas	10 horas	mié 27/02/19	jue 28/02/19

Figura 3.4: Calendario del reconocimiento de la señal de audio de entrada en *Android*

## PITCH TRACKING

### *Procesamiento de la señal*

#### **Iteración B1:**

División de la señal anteriormente guardada en una Lista en ventanas para proceder a procesar cada señal.

Tiempos estimados:

- Implementación división en ventanas: 13 horas
- Realización de pruebas: 10 horas

#### **Iteración B2:**

Transformación de las ventanas aplicando la transformada rápida de Fourier. Para ello se implementa la clase complejos y F.F.T. que se llamaran en el método transformación.

Tiempos estimados:

- Implementación de la clase complejos: 15 horas
- Implementación de la clase F.F.T.: 13 horas
- Implementación método Transformación: 10 horas
- Realización de pruebas: 10 horas

#### **Iteración B3:**

Creación del método para hacer la autocorrelación de las ventanas, después de la previa aplicación de la transformación.

Tiempos estimados:

- Implementación del método autocorrelación: 14 horas
- Realización de pruebas: 11 horas

#### **Iteración B4:**

Método para hacer un tratamiento de los picos de la ventana, para obtener el pico más alto de la ventana.

Tiempos estimados:

- Implementación del método picos: 12 horas
- Realización de pruebas: 10 horas

#### **Iteración B5:**

Método para obtener la frecuencia de las ventanas, en la cual se creará un objeto en el que se almacenará el pico más alto, la ventana, la frecuencia para su posterior análisis.

Tiempos estimados:

- Implementación de la clase resultados: 13 horas
- Implementación método obtener frecuencias: 11 horas
- Realización de pruebas: 12 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>Procesamiento de la señal</b>	<b>28 días</b>	<b>jue 28/02/19</b>	<b>mar 09/04/19</b>
<b>B1</b>	<b>2,88 días</b>	<b>jue 28/02/19</b>	<b>mar 05/03/19</b>
Implementación división en ventanas	13 horas	jue 28/02/19	lun 04/03/19
Realización de pruebas	10 horas	lun 04/03/19	mar 05/03/19
<b>B2</b>	<b>14,75 días</b>	<b>mar 05/03/19</b>	<b>mar 26/03/19</b>
Implementación de la clase complejos	15 horas	mar 05/03/19	jue 07/03/19
Implementación del subservicio Android	15 horas	lun 25/02/19	mié 27/02/19
Realización de pruebas	10 horas	mié 27/02/19	jue 28/02/19
Implementación de la clase FFT	13 horas	jue 07/03/19	vie 08/03/19
Implementación método Transformación	10 horas	lun 11/03/19	mar 12/03/19
Realización de pruebas	10 días	mar 12/03/19	mar 26/03/19
<b>B3</b>	<b>3,13 días</b>	<b>mar 26/03/19</b>	<b>vie 29/03/19</b>
Implementación del método autocorrelación	14 horas	mar 26/03/19	mié 27/03/19
Realización de pruebas	11 horas	jue 28/03/19	vie 29/03/19
<b>B4</b>	<b>2,75 días</b>	<b>vie 29/03/19</b>	<b>mié 03/04/19</b>
Implementación del método picos	12 horas	vie 29/03/19	lun 01/04/19
Realización de pruebas	10 horas	lun 01/04/19	mié 03/04/19
<b>B5</b>	<b>4,5 días</b>	<b>mié 03/04/19</b>	<b>mar 09/04/19</b>
Implementación de la clase resultados	13 horas	mié 03/04/19	jue 04/04/19
Implementación método obtener frecuencias	11 horas	jue 04/04/19	lun 08/04/19
Realización de pruebas	12 horas	lun 08/04/19	mar 09/04/19

**Figura 3.5:** Calendario del procesamiento de la señal

## DISCRETIZACIÓN DE LAS FRECUENCIAS

### *Obtener la nota asignada a cada frecuencia*

#### Iteración C1:

Implementación de un algoritmo que nos calcule y divida cada nota musical según su frecuencia.

Tiempos estimados:

- Implementación del algoritmo: 12 horas
- Realización de pruebas: 9 horas

#### Iteración C2:

Método para asignar una nota a la frecuencia de cada ventana que anteriormente ha sido procesada.

Tiempos estimados:

- Implementación del método: 13 horas
- Realización de pruebas: 10 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>Obtener la nota asignada a cada frecuencia</b>	<b>5,5 días</b>	<b>mar 09/04/19</b>	<b>jue 25/04/19</b>
<b>C1</b>	<b>2,63 días</b>	<b>mar 09/04/19</b>	<b>vie 12/04/19</b>
Implementación del algoritmo	12 horas	mar 09/04/19	jue 11/04/19
Realización de pruebas	9 horas	jue 11/04/19	vie 12/04/19
<b>C2</b>	<b>2,88 días</b>	<b>vie 12/04/19</b>	<b>jue 25/04/19</b>
Implementación del algoritmo	13 horas	vie 12/04/19	mar 23/04/19
Realización de pruebas	10 horas	mar 23/04/19	jue 25/04/19

**Figura 3.6:** Calendario de la obtención la nota asignada a cada frecuencia

### *Duración de las notas e identificación de los silencios*

#### **Iteración D1:**

Método para identificar si una nota es igual a la siguiente.

Tiempos estimados:

- Implementación del algoritmo: 15 horas
- Realización de pruebas: 11 horas

#### **Iteración D2:**

Método para identificar si existe un silencio.

Tiempos estimados:

- Implementación del método: 12 horas
- Realización de pruebas: 10 horas

#### **Iteración D3:**

Cálculo de la duración de cada nota o silencio.

Tiempos estimados:

- Implementación del algoritmo: 11 horas
- Realización de pruebas: 10 horas



Nombre de tarea	Duración	Comienzo	Fin
<b>Duración de las notas e identificación de los silencios</b>	<b>8,63 días</b>	<b>jue 25/04/19</b>	<b>mié 08/05/19</b>
<b>D1</b>	<b>3,25 días</b>	<b>jue 25/04/19</b>	<b>mar 30/04/19</b>
Implementación del algoritmo	15 horas	jue 25/04/19	vie 26/04/19
Realización de pruebas	11 horas	lun 29/04/19	mar 30/04/19
<b>D2</b>	<b>2,75 días</b>	<b>mar 30/04/19</b>	<b>lun 06/05/19</b>
Implementación del algoritmo	12 horas	mar 30/04/19	mié 01/05/19
Realización de pruebas	10 horas	mié 01/05/19	lun 06/05/19
<b>D3</b>	<b>2,63 días</b>	<b>lun 06/05/19</b>	<b>mié 08/05/19</b>
Implementación del algoritmo	11 horas	lun 06/05/19	mar 07/05/19
Realización de pruebas	10 horas	mar 07/05/19	mié 08/05/19

**Figura 3.7:** Calendario del calculo de la duración de las notas e identificación de los silencios

## LEER ARCHIVO MUSICXML

### *Renderizar los resultados obtenidos*

#### **Iteración E1:**

Investigar sobre formatos de visualización de partituras

Tiempos estimados:

- Implementación del algoritmo: 15 horas
- Realización de pruebas: 11 horas

#### **Iteración E2:**

Conocer el formato de los archivos *MusicXML*.

Tiempos estimados:

- Implementación del método: 17 horas
- Realización de pruebas: 12 horas

#### **Iteración E3:**

Creación archivo *MusicXML* con los resultados obtenidos.

Tiempos estimados:

- Implementación del método: 16 horas
- Realización de pruebas: 12 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>Renderizar los resultados obtenidos.</b>	<b>10,38 días</b>	<b>mié 08/05/19</b>	<b>mar 04/06/19</b>
<b>E1</b>	<b>3,25 días</b>	<b>mié 08/05/19</b>	<b>lun 20/05/19</b>
Implementación del algoritmo	15 horas	mié 08/05/19	vie 10/05/19
Realización de pruebas	11 horas	vie 10/05/19	lun 20/05/19
<b>E2</b>	<b>3,63 días</b>	<b>mar 21/05/19</b>	<b>vie 24/05/19</b>
Implementación del algoritmo	17 horas	mar 21/05/19	jue 23/05/19
Realización de pruebas	12 horas	jue 23/05/19	vie 24/05/19
<b>E3</b>	<b>3,5 días</b>	<b>vie 24/05/19</b>	<b>mar 04/06/19</b>
Implementación del algoritmo	16 horas	vie 24/05/19	vie 31/05/19
Realización de pruebas	12 horas	vie 31/05/19	mar 04/06/19

**Figura 3.8:** Calendario de la renderización los resultados obtenidos

## COMPARAR ARCHIVO MUSICXML Y DAR FEEDBACK

### *Comparación de las partituras*

#### **Iteración F1:**

Método para cargar los datos de los archivos *MusicXML* en una estructura.

Tiempos estimados:

- Implementación del algoritmo: 20 horas
- Realización de pruebas: 11 horas

#### **Iteración F2:**

Método para comparar las dos partituras y señalar los puntos donde hay errores.

Tiempos estimados:

- Implementación del método: 17 horas
- Realización de pruebas: 12 horas

Nombre de tarea	Duración	Comienzo	Fin
<b>Comparación de las partituras</b>	<b>7,5 días</b>	<b>mar 04/06/19</b>	<b>jue 13/06/19</b>
<b>F1</b>	<b>3,88 días</b>	<b>mar 04/06/19</b>	<b>vie 07/06/19</b>
Implementación del algoritmo	20 horas	mar 04/06/19	jue 06/06/19
Realización de pruebas	11 horas	jue 06/06/19	vie 07/06/19
<b>F2</b>	<b>3,63 días</b>	<b>lun 10/06/19</b>	<b>jue 13/06/19</b>
Implementación del algoritmo	17 horas	lun 10/06/19	mié 12/06/19
Realización de pruebas	12 horas	mié 12/06/19	jue 13/06/19

**Figura 3.9:** Calendario del la comparación de las partituras

### 3.4.5. PLANIFICACIÓN LOCAL: EXPERIMENTOS Y PRUEBAS

#### Iteración EP1:

Se realizan pruebas a nivel de funcionalidad, comprobando el nivel de fiabilidad del sistema. Tiempos estimados:

- Preparación del sistema: 15 horas.
- Realización de las pruebas: 35 horas
- Recopilación y análisis de los resultados: 25 horas.

#### Iteración EP2:

Se pone a prueba la interfaz haciendo un análisis crítico.

Tiempos estimados:

- Preparación del sistema: 15 horas.
- Realización de las pruebas: 25 horas
- Recopilación y análisis de los resultados: 20 horas.

Nombre de tarea	Duración	Comienzo	Fin
<b>Experimentos y pruebas</b>	<b>13,13 días</b>	<b>jue 13/06/19</b>	<b>mar 02/07/19</b>
<b>EP1</b>	<b>7,5 días</b>	<b>jue 13/06/19</b>	<b>mar 25/06/19</b>
Preparación del sistema	10 horas	jue 13/06/19	vie 14/06/19
Realización de las prueba	30 horas	vie 14/06/19	jue 20/06/19
Recopilación y análisis de los resultados	20 horas	jue 20/06/19	mar 25/06/19
<b>EP2</b>	<b>5,63 días</b>	<b>mar 25/06/19</b>	<b>mar 02/07/19</b>
Preparación del sistema	10 horas	mar 25/06/19	mié 26/06/19
Realización de las prueba	20 horas	mié 26/06/19	vie 28/06/19
Recopilación y análisis de los resultados	15 horas	vie 28/06/19	mar 02/07/19

**Figura 3.10:** Calendario experimentos y pruebas



---

## CAPÍTULO 4

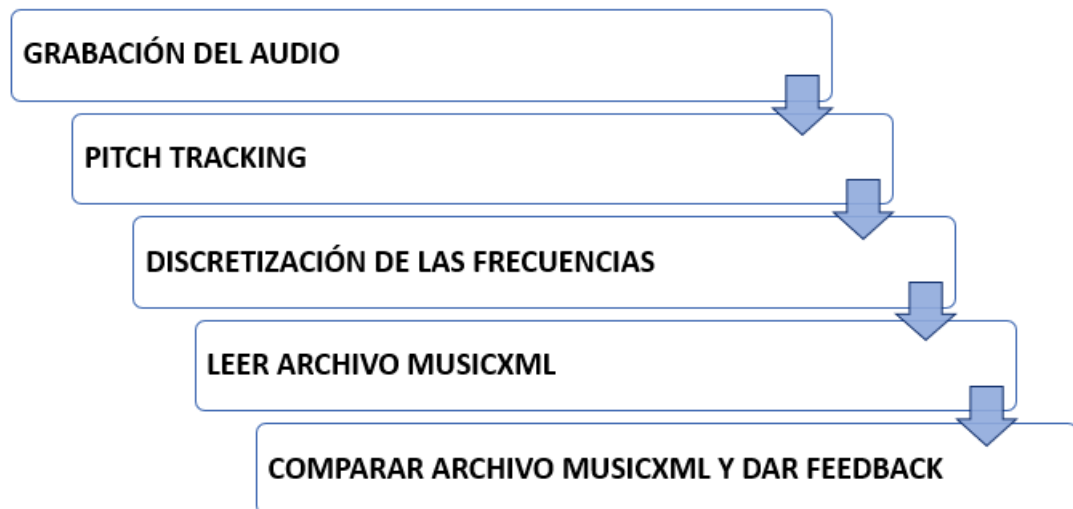
# RESULTADOS

---

En este capítulo se manejan los resultados obtenidos durante la ejecución de las distintas fases del desarrollo. Se estudian los resultados de las actividades previamente planificadas en el desarrollo de la aplicación, los experimentos y las pruebas.

### 4.1. VISIÓN GENERAL

Las fases por las que pasa el sistema son tal y como podemos ver en el siguiente diagrama:



**Figura 4.1:** Fases del sistema

Para facilitar la comprensión de todas las fases, en el Anexo C se muestra el diagrama comunicación entre las etapas con sus funciones internas.

### 4.2. GRABACIÓN DEL AUDIO

Como desde Flutter no podemos obtener la señal recogida del micrófono, se ha tenido que crear un subservicio *Android* para recoger la señal deseada.

#### 4.2.1. RECONOCIMIENTO DE LA SEÑAL DE AUDIO DE ENTRADA EN ANDROID

El módulo SW referente al reconocimiento de la señal de audio de entrada en *Android* es un conjunto de métodos programados en *Java* encargados de grabar audio de forma continua a través del micrófono y guardarlo en una estructura tipo lista para su posterior análisis.

Para poder obtener lo escuchado por el micrófono, se ha implementado un subservicio *Android* tal y como se muestra en el siguiente código.

```
1 public void handleMessage(Message msg) {  
2  
3     int bufferSize = AudioRecord.getMinBufferSize(16384,  
4         AudioFormat.CHANNEL_IN_MONO,  
5         AudioFormat.ENCODING_PCM_8BIT);  
6  
7     if (bufferSize == AudioRecord.ERROR || bufferSize ↵  
8         ↵ == AudioRecord.ERROR_BAD_VALUE) {  
9         bufferSize = 2048;  
10    }  
11  
12    AudioRecord record = new ↵  
13        ↵ AudioRecord(MediaRecorder.AudioSource.MIC,  
14        16384,  
15        AudioFormat.CHANNEL_IN_MONO,  
16        AudioFormat.ENCODING_PCM_8BIT,  
17        bufferSize);  
18  
19    byte[] audioBuffer = new byte[bufferSize];  
20  
21    record.startRecording();  
22    record.stop();  
23    record.release();  
24    stopSelfResult(msg.arg1);  
25 }
```

En este servicio *Android* se está usando la clase *AudioRecord* que como se puede observar tiene los siguientes parámetros:

- La frecuencia de muestreo es de 16384.
- El formato del archivo es mono.
- La codificación es de 8 bits.

La función principal del algoritmo es digitalizar el sonido obtenido por el micrófono transformado la señal en una lista de *bytes*.

### 4.3. PITCH TRACKING

El módulo SW referente al procesamiento de la señal está formado por una serie de clases relacionadas entre sí, tal y como se puede observar en el siguiente diagrama:

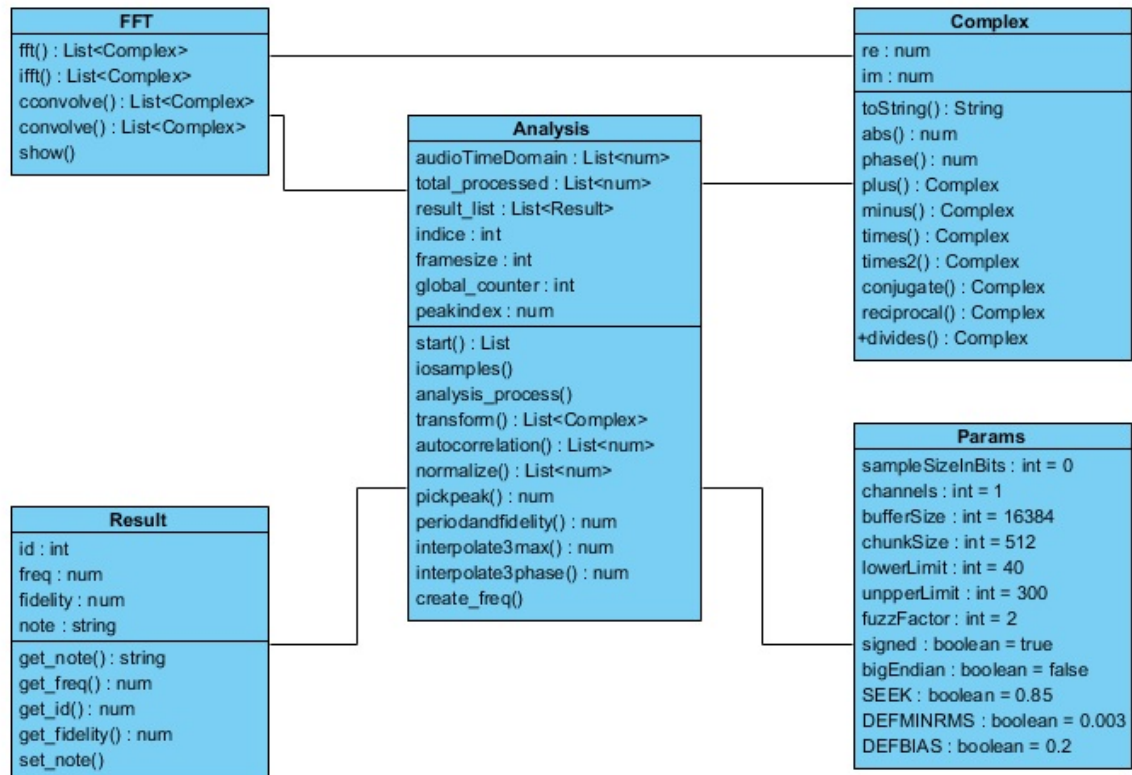


Figura 4.2: Diagrama de clases del porcesamiento de la señal

La función de cada una de las clases es:

- **Params:** Clase en las que se encuentran todas las constantes utilizadas en los distintos pasos del procesamiento de la señal.
- **Complex:** Implementación del tipo de datos de números complejos. En esta clase realizamos operaciones sobre este tipo de números para su posterior uso en la clase F.F.T..
- **FFT:** Clase que realiza la Transformada rápida de Fourier, por eso recibe el nombre de FFT. En esta clase forma parte es la implementación de los fundamentos físicos vistos en apartados anteriores. Esta clase también hace uso de la clase **Complex** y de sus funcionalidades.
- **Result:** Clase dedicada a guardar la información más relevante en el análisis de cada ventana.
- **Analysis:** Clase central del procesamiento de la señal, en la cual se definen las distintas etapas del procesamiento.

Este módulo está destinado al procesamiento de la señal. Este procesamiento se desarrolla en diferentes etapas.

### 4.3.1. DIVISIÓN DE LA SEÑAL EN VENTANAS

Para facilitar el análisis de toda la señal será necesario dividirla en ventanas que serán analizadas una a una. El método *iosamples()* se encarga de la división de la ventana y de enviarla a analizar.

```

1 void iosamples(){
2     // Declaration and initialization of parameters.
3     framesize=1024;
4     int mask = framesize - 1;
5     int counter;
6     List <num> analysis = new List(framesize);
7
8     do{
9         //control of the counters of the windows
10        counter = 0;
11        num resta=audioTimeDomain.length-global_counter;
12        if(mask > resta){
13
14            num aux= framesize-resta;
15
16            for(int i=0;i<resta;i++){
17                analysis[counter] = audioTimeDomain[global_counter];
18                global_counter++;
19                counter++;
20            }
21            for(int j=0;j<aux-1;j++){
22                analysis[counter] = 0;
23                counter++;
24            }
25        }else{
26
27            // Filling the window to analyze
28            while (counter < mask && global_counter < ↵
                ↵ audioTimeDomain.length) {
29
30                analysis[counter] = audioTimeDomain[global_counter];
31                global_counter++;
32                counter++;
33            }
34        }
35
36        // Start of the analysis
37        analysis_process(analysis);
38
39    }while(global_counter<audioTimeDomain.length); //end of all windows

```

Como se observa en el método, la variable *framesize* indica el tamaño que queremos que tenga la ventana.

Después de saber el tamaño de la ventana, se comprueba la cantidad de señal que queda por analizar, en el caso de que no sea lo suficiente para llenar una ventana esta se completa llenándose de ceros. Si por el contrario se dispone de señal para llenar una ventana, se añade el trozo correspondiente.

Cabe destacar que la variable *global counter*, tiene como principal función, la de ser un contador encargado de decir, en cada momento, en qué parte de la señal se encuentra.



### 4.3.2. ANÁLISIS DE LA VENTANA

Una vez obtenida la ventana, debe ser analizada. El método **analysis process** recibe como parámetro de entrada la ventana encargándose de preparar la ventana para someterla a la F.F.T. y al proceso de autocorrelación.

```

1 void analysis_process(List <num> analysis ) {
2     // variables for the analysis
3     List<Complex> complex_aux;
4     List<num> processed;
5     List<num> processbuf = new List(analysis.length);
6     List<Complex> complex_3;
7     num peak;
8     FFT f = new FFT();
9     num norm = (1.0 / sqrt((analysis.length * 2)));
10    // copy input to processing buffer
11    for (int n = 0; n < analysis.length-1; n++)
12        processbuf[n] = analysis[n] * norm;
13    // Analysis process
14    complex_aux = transform(processbuf);
15    complex_3 = f.ifft(complex_aux);
16    processed = autocorrelation(complex_aux, complex_3);
17    peak = pickpeak(processed);
18
19    for (int i = 0; i < processed.length; i++) {
20        total_processed[index]=processed[i];
21        index++;
22    }
23    create_freq(peak);
24 }

```

Se somete la ventana al proceso de transformación y de autocorrelación para después sacar su pico de frecuencia más alto.

Para tener un historial del todo el análisis, la ventana ya analizada se guarda en una lista y se procede a sacar la frecuencia de la ventana con el método **create freq** al cual se le pasa el pico más alto.

### 4.3.3. TRANSFORMACIÓN DE LA SEÑAL

Para transformar la señal se hace uso de la clase **FFT**, ya que el método **transform** recibe la ventana, y la somete a la transformada de Fourier y devuelve el resultado obtenido.

```

1 List<Complex> transform(List <num>processbuf){
2
3     // variables for the transform
4     List<Complex> complex_1;
5     List<Complex> complex_2;
6     FFT f = new FFT();
7     num aux=processbuf.length;
8     complex_1= new List(aux) ;
9     // Complex array creation
10    for (int i = 0; i < processbuf.length-1; i++)
11        complex_1[i] = new Complex(processbuf[i], 0.0);
12    // Calculation of complex numbers
13    complex_2 = f.fft(complex_1);
14    return complex_2;
15 }

```

Destacar que, al no poseer la parte imaginaria del número, se le asigna el valor 0. Como podemos observar en la línea 6 creamos la instancia de la clase **FFT** y llamamos al método encargado de hacer la transformada en la línea 15.

#### 4.3.4. AUTOCORRELACIÓN DE LA SEÑAL

Una vez obtenida la transformación de la señal, se lleva a cabo la autocorrelación tal y como se ha descrito en los fundamentos teóricos, y se aplica en el método de *autocorrelation*.

```

1 List<num> autocorrelation(List<Complex> complex, List<Complex> ↵
   ↵ complex_3) {
2     // variables for the autocorrelation
3     int n;
4     num value, value_2, val_aux;
5     List<num> procesados;
6     // compute power spectrum
7     value = complex[0].getRe() * complex[0].getRe();
8     complex[0].setRe(value);
9     value_2 = complex[complex.length - 1].getRe() * ↵
   ↵ complex[complex.length - 1].getRe();
10    complex[complex.length - 1].setRe(value_2);
11
12    for (n = 1; n < complex.length - 1; n++) {
13        val_aux = (complex[n].getRe() * complex[n].getRe()) + ↵
   ↵ (complex[n].getIm() * complex[n].getIm());
14        complex[n].setRe(val_aux);
15        complex[n].setIm(0.0);
16    }
17
18    procesados = List(complex_3.length);
19
20    for (n = 0; n < complex_3.length - 1; n++) {
21        procesados[n] = complex_3[n].getRe();
22    }
23    return procesados;
24 }

```

Este método hace el cómputo del espectro de la ventana y lo devuelve en una lista como procesado.

#### 4.3.5. DETENCIÓN DE LOS PICOS

Finalizado todo el procesamiento de la ventana, se inicia la detección de los picos, para ello se localizan todos los máximos (reales y relativos) y cuando se finaliza el recorrido por toda la señal, se recoge el máximo. Esta es la función del método *pickpeak*.

Para poder comprender el funcionamiento de este algoritmo, podemos seguir el diagrama que se encuentra a continuación. En él, podemos ver que inicialmente el resultado es inicializado a cero, y que a partir de ahí se empieza a comprar el punto actual con el punto siguiente y con el anterior, para identificar si es un máximo local, luego comparamos si el máximo encontrado es mayor que el que teníamos ya guardado o no.

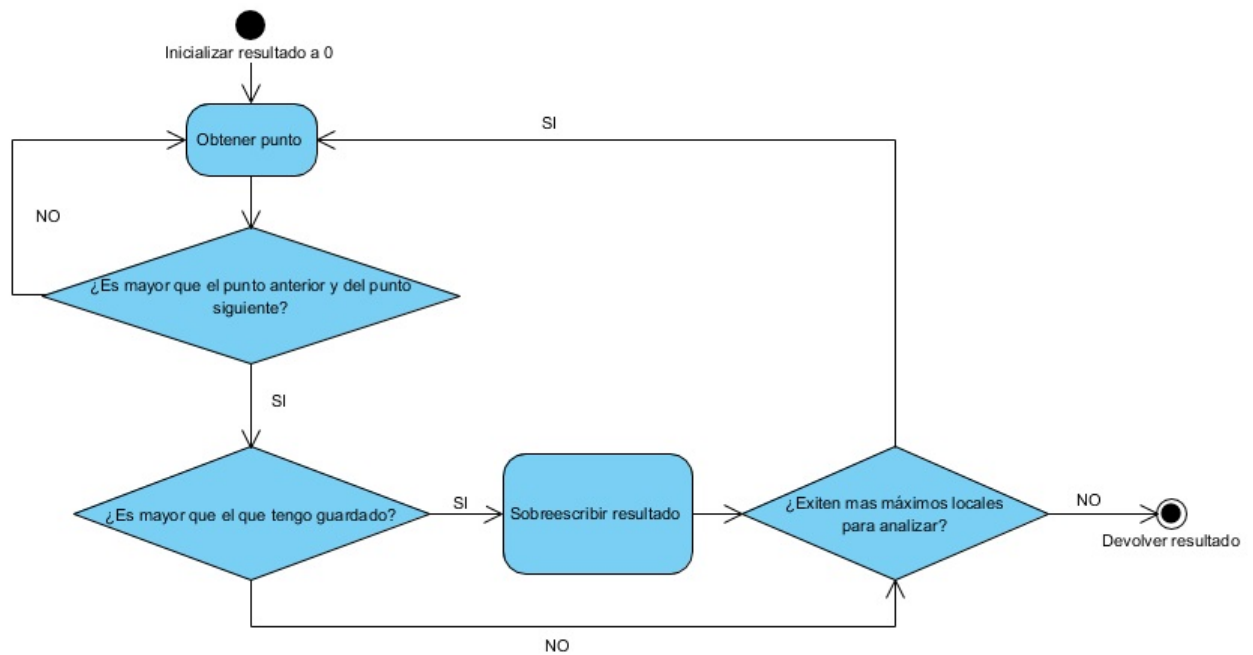


Figura 4.3: Diagrama obtención pico máximo

#### 4.3.6. OBTENCIÓN DE LA FRECUENCIA DE LA VENTANA

Una vez buscado el pico máximo de la ventana, se podrá iniciar el cálculo de la frecuencia que posee. Para ello se utiliza el método *create\_freq*.

```

1 void create_freq(num peak){
2     num periodo=(1/ ps.bufferSize)*peak;
3     num freq=1/periodo;
4     Result r= new Result(global_counter, freq);
5     result_list.add(r);
6 }
  
```

Este método calcula la frecuencia de la ventana, y la guarda como un objeto de la clase *result*.

### 4.4. DISCRETIZACIÓN DE LAS FRECUENCIAS

Finalizado todo lo relacionado con el *Pitch Tracking* y teniendo asignada a cada ventana una frecuencia específica, se procede a realizar todos los procesamientos necesarios para poder obtener todas las notas asignadas a cada ventana, su duración y la figura musical a la que equivale.

#### 4.4.1. OBTENER LA NOTA ASIGNADA A CADA FRECUENCIA

El módulo SW referente a la obtención de la nota asignada a cada frecuencia está formado por una serie de clases relacionadas entre sí, tal y como se observa en el siguiente diagrama:

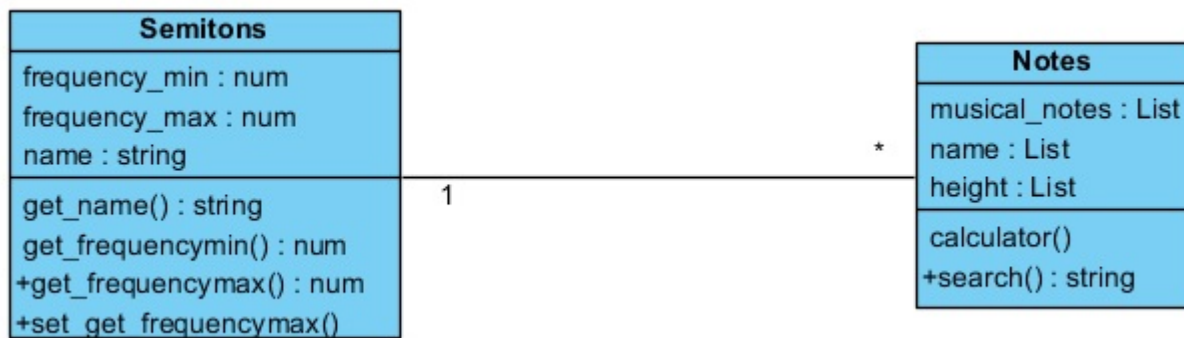


Figura 4.4: Diagrama de clases de las frecuencias de las notas

La función de cada una de las clases es:

- **Semitons:** Clase que almacena el nombre de la nota y el rango de frecuencia que abarca.
- **Notes:** Clase encargada de dar funcionalidad a todo lo relacionado de dar nombre y rango de frecuencias a las notas.

Durante este módulo se han calculado todas las frecuencias de las notas junto con su altura, además, se ha implementado un método de búsqueda, consistente en dada una frecuencia nos devuelva la nota y altura correspondiente.

#### 4.4.2. CALCULO DE LAS FRECUENCIAS DE LAS NOTAS

Inicialmente se define el nombre de las notas y se le asigna un rango de frecuencia para distinguir unas de otras.

```

1 List <Semitons> musical_notes;
2 List<String> name = ↵
   ↵ ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'];
3 List<String> height = ['1', '2', '3', '4', '5', '6', '7'];
4
5 void calculator(){ //assignment to each note a frequency
6
7     num freq=32.7; // minimum frequency C1
8     int cont_1=0;
9     int cont_2=0;
10    int indi=0;
11    musical_notes= new List(83); // number of notes
12
13    for(int i=0; i<musical_notes.length;i++){
14        if(i!=0){
15            freq=freq * pow(2,1/12);
16            freq.toStringAsFixed(3);
17        }
18        musical_notes[i]=new Semitons(freq, ↵
           ↵ name[cont_1]+height[indi]);
19
20        if(cont_1==11){
21            cont_1=0;
22        }else cont_1++;
23
  
```

```

24         if(cont_2==11){
25             cont_2=0;
26             indi++;
27         }else    cont_2++;
28     }
29     for(int i=0; i<musical_notes.length;i++){
30         if(i<musical_notes.length-2){
31             musical_notes[i].set_get_frequecymax(musical_notes[i+1]
32             .frequen y_min-0.001);
33         }else
34             musical_notes[i].set_get_frequecymax(
35             musical_notes[musical_notes.length-1].get_frequecymmin()+1);
36     }
37 }

```

La frecuencia base, se corresponde con la primera nota Do del piano cuya frecuencia equivale a 32.7 (línea 8). Una vez definida la base, se administran los fundamentos teóricos sobre el sistema de afinación temperado y se aplica la fórmula para obtener el resto de notas.

A cada nota se le asigna un nombre, (en notación americana), y una altura para definir en qué octava del piano se encuentra.

#### 4.4.3. BÚSQUEDA DE LAS NOTAS A TRAVÉS DE UNA FRECUENCIA

Este método relaciona el análisis de la señal de entrada con la obtención de las notas musicales. Por eso este método recibe como entrada un número de frecuencia.

```

1 String search(num freq){
2
3     for(int i=0; i<musical_notes.length;i++){
4
5         if(freq>=musical_notes[i].get_frequencymmin() && ←
6         ↪ freq<=musical_notes[i].frequency_max){
7             return musical_notes[i].get_name();
8         }
9     }
10    return '';
11 }

```

#### 4.4.4. DURACIÓN DE LAS NOTAS E IDENTIFICACIÓN DE LOS SILENCIOS

Una vez identificadas todas las notas, debemos efectuar otra filtración para identificar dos notas seguidas con el mismo nombre agrupándolas y sumando su duración para finalmente asignarle una figura musical.

Si la nota tiene una frecuencia menor a la estipulada como más baja (C1), se considera que es un silencio, y se ha almacenado con el nombre de **Silent**.

El diagrama de clases de esta sección es el siguiente:

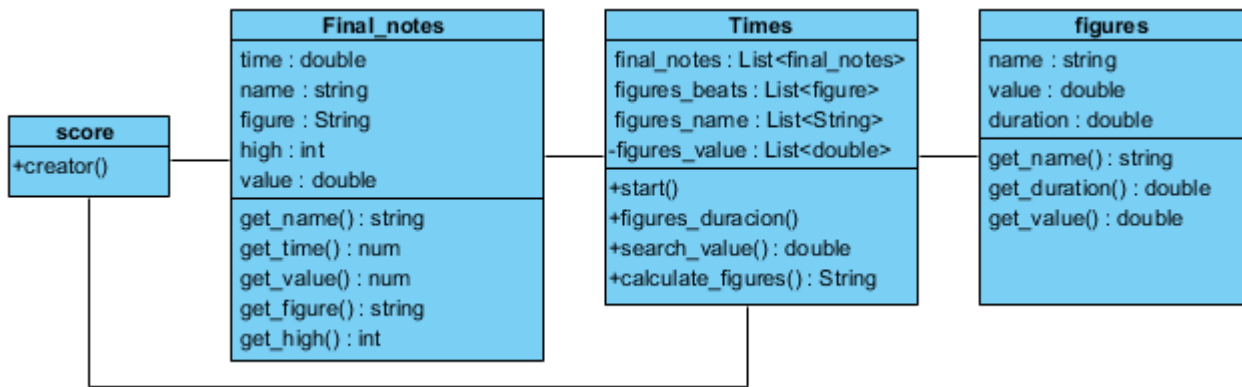


Figura 4.5: Diagrama de clases de las duración de las notas

La función de cada una de las clases es:

- **Figure:** Clase que almacena el nombre de las figuras musicales junto con su duración en el tiempo.
- **Final Notes:** Clase que almacena el nombre de las notas, con su duración en el tiempo y la figura asignada.
- **Times:** Clase que hace el filtrado de las notas, agrupandolas y asignandole una figura musical anteriormente calculada.
- **Score:** Clase encargada de generar una partitura en *MusicXML* con las notas guardadas.

Durante este modulo se han calculado los tipos de figuras musicales básicas existentes, asignándoles su duración en el tiempo según la velocidad de metrónomo.

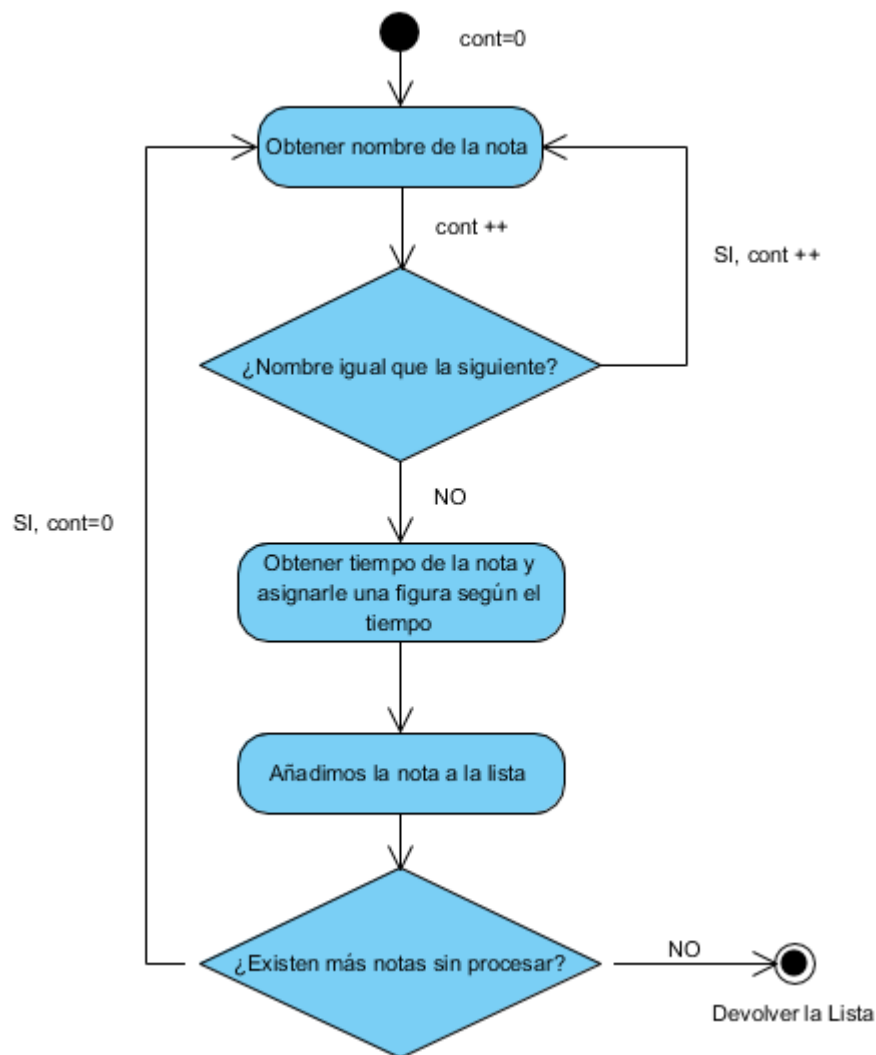
```

1 void figures_duracion(){
2     double quarter_note=60/beats; // Calculation of the time of each quarter
3     int divide=4;
4     int mult=2;
5     double value;
6     int cont=0;
7
8     for(int i=0;i<figures_beats.length;i++){
9         if(divide>0){
10             value=quarter_note/divide;
11             divide--;
12             divide--;
13         }else if(divide==0){
14             value=quarter_note;
15             divide--;
16         }else{
17             value=quarter_note*mult;
18             mult++;
19             mult++;
20         }
21         figures_beats[i]=new figures(figures_name[cont], ←
22             ↪ value,figures_value[cont]);
23         cont++;
24     }
25 }
  
```

Obtenida la duración de cada figura musical, se procede a realizar un segundo filtrado de las notas identificadas. El algoritmo hará las siguientes tareas:

- Agrupar las notas seguidas con el mismo nombre.
- Sumar los tiempos de todas las notas seguidas iguales.
- Obtener una figura según el tiempo que dure.
- Añadir la nota con todos sus valores a una lista

El funcionamiento sería como se observa en el siguiente diagrama:



**Figura 4.6:** Diagrama obtención Lista de notas definitivas

Una vez obtenida toda la información captada por el micrófono, se tiene la opción de crear una partitura o compararla con otra.

## 4.5. LEER ARCHIVO MUSICXML

En esta sección se puede observar como leer el archivo en formato *MusicXML* para después compararlo con lo obtenido por el micrófono.

El formato de un archivo *MusicXML* tiene las siguientes partes:

- **Cabecera:** Establece el tipo de version y codificación del archivo.

```

1 <?xml version="1.0" encoding='UTF-8' standalone='no' ?>
2 <!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD
   ↪ MusicXML 2.0 Partwise//EN"
   ↪ "http://www.musicxml.org/dtds/partwise.dtd">

```

- **Formato:** Aquí se indican las reglas de estilo de la partitura, el título, el autor y el resto de información relevante de la misma.

```

1 <score-partwise version="2.0">
2   <work>
3     <work-title />
4   </work>
5   <identification>
6     <rights>Copyright    </rights>
7     <encoding>
8       <encoding-date>2019-07-04</encoding-date>
9       <encoder>Eduardo Mora</encoder>
10      <software>Sibelius 7.0.2</software>
11      <software>Direct export, not from Dolet</software>
12      <encoding-description>Sibelius / MusicXML
13    </identification>
14    <part-list>
15      <score-part id="P1">
16        <part-name>P1</part-name>
17        <score-instrument id="P1-I1">
18          <instrument-name> </instrument-name>
19        </score-instrument>
20      </score-part>
21    </part-list>
22    <part id="P1">
23      <measure number="1" width="190">
24        <print new-page="yes">
25          <system-layout>
26            <system-margins>
27              <left-margin>22</left-margin>
28              <right-margin>0</right-margin>
29            </system-margins>
30            <top-system-distance>218</top-system-distance>
31          </system-layout>
32        </print>

```

- **Clave, armadura y compás:** En esta parte es donde se especifica la tonalidad de la obra, la clave en la que está y el compás.

```

1 <attributes>
2   <divisions>256</divisions>
3   <key color="#000000">
4   <fifths>0</fifths>

```



```

5      <mode>major</mode>
6      </key>
7      <time color="#000000">
8          <beats>4</beats>
9          <beat-type>4</beat-type>
10     </time>
11     <staves>1</staves>
12     <clef number="1" color="#000000">
13         <sign>G</sign>
14         <line>2</line>
15     </clef>
16     <staff-details number="1" print-object="yes" />
17 </attributes>

```

- **Definición de las notas:** Una vez se haya definido todo, solo queda señalar cada una de las notas una detrás de otra, asignando su nombre, altura y figura musical correspondiente. En el caso de los silencios, sólo hay que indicar la figura musical.

```

1      <note color="#000000" default-x="162">
2          <pitch>
3              <step>A</step>
4              <octave>4</octave>
5          </pitch>
6          <duration>128</duration>
7          <instrument id="P1-I1" />
8          <voice>1</voice>
9          <type>eighth</type>
10         <stem>up</stem>
11         <staff>1</staff>
12         <beam number="1">begin</beam>
13     </note>

```

- **Cierre del archivo:** Con estas líneas se procede al cierre del archivo:

```

1     </part>
2 </score-partwise>

```

Definido y conocido el formato del archivo *MusicXML*, estará preparado para ser analizado, ya que se deben encontrar todas las posibles notas musicales, y añadirlas junto con su nombre, altura y figura musical a una lista.

El algoritmo encargado de crear la lista divide el texto cada vez que encuentra la palabra que identifica una nota, para dividir el texto se usa la función *fileText.split("note")* que devuelve una lista de String.

Un vez dividido el texto por notas, procederemos a buscar la información necesaria de cada nota para añadirla a la lista, para ello también se hará uso de la función *fileText.split()*.

## 4.6. COMPARAR ARCHIVO MUSICXML Y DAR FEEDBACK

En este último módulo se compara la partitura original ya cargada y almacenada en una lista con la lista de notas generadas a través del micrófono. En el siguiente diagrama se muestra el funcionamiento del algoritmo:

Una vez que termina el algoritmo muestra por pantalla todos los errores cometidos. Además, se calcula un porcentaje de progreso con la siguiente fórmula:

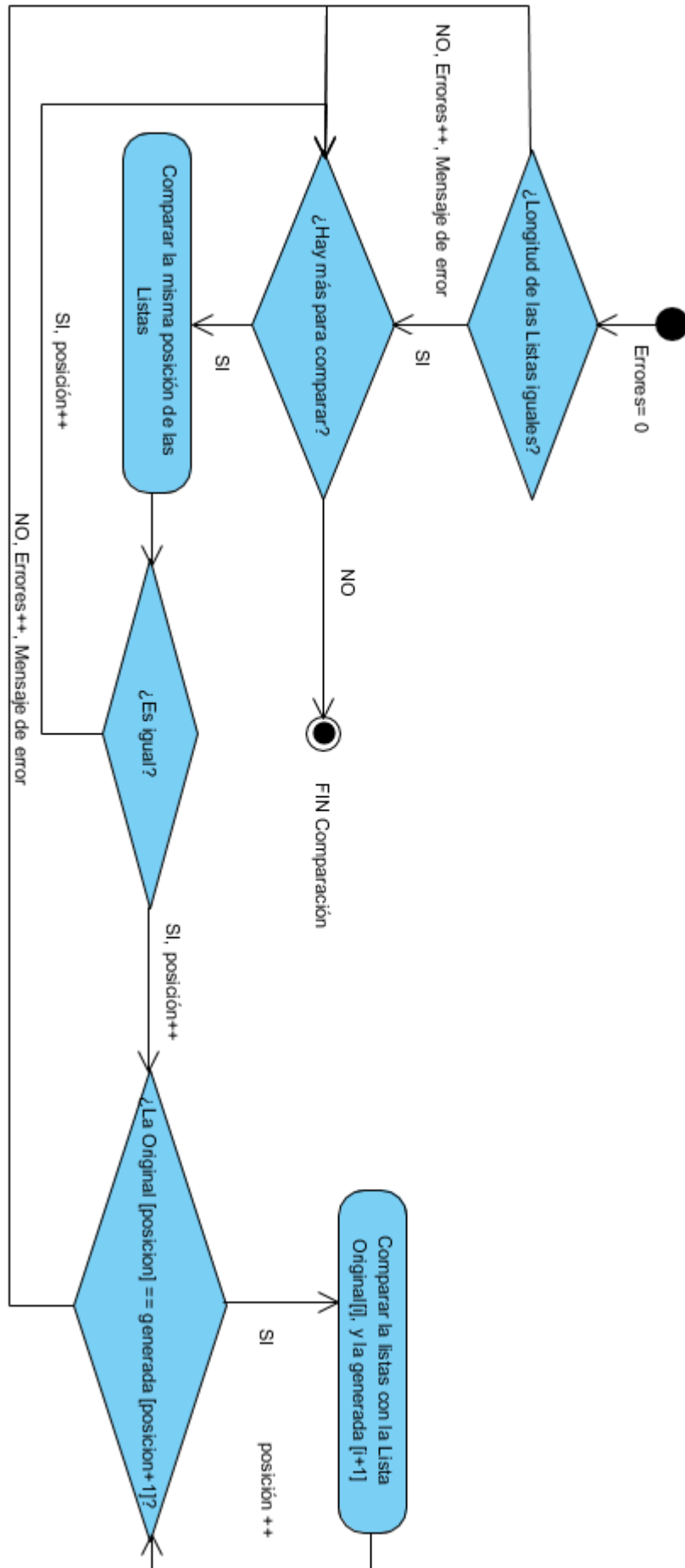


Figura 4.7: Diagrama de la comparación

$$\frac{(TotalNotas - Errores) * 100}{TotalNotas} \quad (4.1)$$

Así el usuario puede saber que errores ha cometido y que porcentaje tiene de la obra.



---

## CAPÍTULO 5

# EVALUACIÓN

---

En este capítulo se muestran una serie de evaluaciones sobre los algoritmos, que se han sometido a una serie de pruebas y observando el comportamiento obtenido en cada una de ellas.

### 5.1. EVALUACIÓN PITCH TRACKING

El algoritmo de *Pitch Tracking* está formado por distintas etapas. Para evaluar el algoritmo completo se han realizado una serie de pruebas en varios puntos de las distintas etapas de este.

#### 5.1.1. EVALUACIÓN DE LA DIVISIÓN EN VENTANAS

En primer lugar, se debe comprobar que la señal obtenida se divide bien en ventanas, ya que después, cada ventana será analizada una a una.

Para comprobar el correcto funcionamiento, se mostrará el tamaño de la señal de entrada y el número de ventanas creadas:

```
EL TAMAÑO DE LA ENTRADA ES:15398  
EL TAMAÑO DE LA VENTANA ES DE: 1024 Y EL NUMERO DE VENTANAS SON: 16
```

**Figura 5.1:** Comprobación número de ventanas

Para probar si la división en ventana funciona de forma correcta, se prueba con otro tamaño de ventana:

Tal y como se puede observar el resultado ha sido el esperado, ya que multiplicando el número de las ventanas por su tamaño, nos dará un número mayor o igual a la señal de entrada. El resto de posiciones sobrantes de la ventana después de tener toda la señal se rellena de ceros.

#### 5.1.2. EVALUACIÓN DE LA OBTENCIÓN DEL PICO MÁXIMO

El cálculo del pico máximo y la posición de la ventana en la que se encuentra son información importante, ya que con estos dos datos podemos calcular después la frecuencia de la ventana.

```
EL TAMAÑO DE LA ENTRADA ES:25894  
EL TAMAÑO DE LA VENTANA ES DE: 512 Y EL NUMERO DE VENTANAS SON: 51
```

**Figura 5.2:** Comprobación número de ventanas con otro tamaño

Para comprobar su funcionamiento, se muestra por la terminal el máximo valor y la posición en la que se encuentra:

```
El pico máximo obtenido es: 9.033419015835758 y su posición es: 313
El pico máximo obtenido es: 7.075806280385886 y su posición es: 42
El pico máximo obtenido es: 16.658821537376262 y su posición es: 104
El pico máximo obtenido es: 40.57519895779746 y su posición es: 20
El pico máximo obtenido es: 52.43505230530199 y su posición es: 25
El pico máximo obtenido es: 27.49799001213568 y su posición es: 377

EL TAMAÑO DE LA ENTRADA ES:11846
EL TAMAÑO DE LA VENTANA ES DE: 2048 Y EL NUMERO DE VENTANAS SON: 6
```

**Figura 5.3:** Comprobación del pico máximo y su posición

Pero para realizar una comprobación más exhaustiva, se mostrara por la terminal todos los máximos locales, y se comprueba si el elegido es el mayor de todos:

```
El pico obtenido es: 0.4606636453208574 y su posición es: 2
El pico obtenido es: 1.4582830584031106 y su posición es: 5
El pico obtenido es: 4.403735533722025 y su posición es: 416
El pico obtenido es: 7.414625005175026 y su posición es: 458
El pico obtenido es: 58.192418925606745 y su posición es: 660
-----
El pico máximo obtenido es: 58.192418925606745 y su posición es: 660
-----
El pico obtenido es: 0.1856116004422978 y su posición es: 2
El pico obtenido es: 0.21764567627184261 y su posición es: 4
El pico obtenido es: 0.25139357349037006 y su posición es: 6
El pico obtenido es: 0.7395486495423871 y su posición es: 18
El pico obtenido es: 1.2392762937965223 y su posición es: 21
El pico obtenido es: 3.90790982399428 y su posición es: 41
El pico obtenido es: 7.611524682237585 y su posición es: 105
-----
El pico máximo obtenido es: 7.611524682237585 y su posición es: 105
-----
El pico obtenido es: 0.29346490877392983 y su posición es: 2
El pico obtenido es: 2.2628244872864554 y su posición es: 6
El pico obtenido es: 13.12549573592269 y su posición es: 22
-----
El pico máximo obtenido es: 13.12549573592269 y su posición es: 22
-----
El pico obtenido es: 0.3533087930434144 y su posición es: 3
El pico obtenido es: 1.8081450035205422 y su posición es: 5
-----
El pico máximo obtenido es: 1.8081450035205422 y su posición es: 5
-----
```

**Figura 5.4:** Comprobación de los máximos locales y del pico máximo junto con su posición

### 5.1.3. EVALUACIÓN DEL CALCULO DE LA FRECUENCIA

Para la comprobación del algoritmo completo se le somete a una batería de pruebas, las cuales son un número de frecuencias correspondiente a notas musicales concretas y se comprueba si son reconocidas. Las frecuencias probadas son:

NOTA	FRECUENCIAS	RESULTADO
A4	440	Identificada
C3	130.81	Identificada
F5	698.46	Identificada
B3	246.94	Identificada
Bb4	466	Identificada
Bb5	932.33	Identificada
C1	32.7	Identificada
D2	73.42	Identificada
E3	164.81	Identificada
G7	3135.96	Identificada

**Tabla 5.1:** Frecuencias generadas

Con estas pruebas realizadas, se llega a la conclusión que el algoritmo funciona de manera correcta.

## 5.2. EVALUACIÓN DISCRETIZACIÓN DE LAS FRECUENCIAS

Sabiendo que el algoritmo de *Pitch Tracking* funciona de manera correcta, se va a proceder a la evaluación del algoritmo que discretiza las frecuencias en notas musicales y luego a cada nota musical se le asigna su figura correspondiente según el tiempo que dure.

### 5.2.1. EVALUACIÓN ASIGNACIÓN A CADA FRECUENCIA UNA NOTA MUSICAL

Para realizar esta evaluación se ha procedido de manera similar a la evaluación del cálculo de la frecuencia.

Primero se ha probado solo una frecuencia:

NOTA	FRECUENCIA
E3	164.81

**Tabla 5.2:** Frecuencias generada un sonido

Para comprobar su funcionamiento, se ha mostrado por la terminal el nombre de la nota, su altura y la frecuencia identificada.

```
La nota es: E3 que corresponde con la frecuencia: 164.56
uid=10287(com.example.learning_music_project) Thread-2 identical 4 lines
La nota es: E3 que corresponde con la frecuencia: 164.56
```

**Figura 5.5:** Salida de la discretización de una frecuencia

En la salida la frecuencia ha sido identificada de manera correcta, y ha aparecido más de una línea ya que se asigna a cada ventana una frecuencia y a cada frecuencia una nota.

Ahora se va a proceder a la comprobación de varias frecuencias alternas. Las frecuencias probadas son:

NOTA	FRECUENCIA
E3	164.81
A4	440
C3	130.81
E3	164.81

**Tabla 5.3:** Frecuencias generada varios sonidos

La salida ha sido:

```
La nota es: E3 que corresponde con la frecuencia: 164.56
uid=10287(com.example.learning_music_project) Thread-2 identical 6 lines
La nota es: E3 que corresponde con la frecuencia: 164.56
La nota es: A4 que corresponde con la frecuencia: 440
uid=10287(com.example.learning_music_project) Thread-2 identical 3 lines
La nota es: A4 que corresponde con la frecuencia: 440
La nota es: C3 que corresponde con la frecuencia: 131.02
uid=10287(com.example.learning_music_project) Thread-2 identical 4 lines
La nota es: C3 que corresponde con la frecuencia: 131.02
La nota es: E3 que corresponde con la frecuencia: 164.56
uid=10287(com.example.learning_music_project) Thread-2 identical 3 lines
La nota es: E3 que corresponde con la frecuencia: 164.56
```

**Figura 5.6:** Salida de la discretización de varias frecuencias

Se puede comprobar que el algoritmo funciona de manera correcta.

### 5.2.2. EVALUACIÓN DE LA ASIGNACIÓN DE FIGURAS MUSICALES

Este algoritmo depende del *tempo* (número de negras que hay en un minuto), por esta razón el algoritmo será probado con distintos *tempos*.

Pero antes de nada hay que comprobar que se agrupa cada ventana idéntica y se suma su tiempo. Para saber el tiempo que tiene cada ventana se calcula de la siguiente manera:

$$\frac{1\text{segundo}}{\frac{\text{Muestras/s}}{\text{framesize}}} = \frac{1}{\frac{16384}{\text{framesize}}} = \text{TiempoVentana} \quad (5.1)$$

Sabiendo que en cada segundo se recogen 16384 muestras, y que 16384 se divide en X ventanas, ya que cada ventana depende del *framesize* que tenga, pues solo hay que dividir 1 segundo por el numero de ventanas que tiene.

Ahora vamos a ver como agrupa los sonidos en uno, y suma su tiempo, para calcular el tiempo hemos puesto el tamaño de la ventana en 1024. Las frecuencias usadas son:

NOTA	FRECUENCIA
A4	440
E3	164.81

**Tabla 5.4:** Frecuencias usadas para agrupar



Aplicando la formula anterior se supone que cada ventana tiene un tiempo de:

$$\frac{1\text{segundo}}{\frac{\text{Muestras/s}}{\text{framemsiz}}} = \frac{1}{\frac{16384}{1024}} = 0,0625 \quad (5.2)$$

La salida ha sido la siguiente:

```
La nota es: A4 que corresponde con la frecuencia: 442
uid=15849(com.example.learning_music_project) Thread-2 identical 15 lines
La nota es: A4 que corresponde con la frecuencia: 442
La nota es: E3 que corresponde con la frecuencia: 165.02
La nota es: E3 que corresponde con la frecuencia: 165.02
uid=15849(com.example.learning_music_project) Thread-2 identical 18 lines
La nota es: E3 que corresponde con la frecuencia: 165.02
=====
La nota es: A4 tiene una duración de: 1,0625 segundos
La nota es: E3 tiene una duración de: 1,3125 segundos
=====
```

Figura 5.7: Salida de unir varias notas

Comprobado el funcionamiento de la salida y del calculo del tiempo, vamos a ver si dado un *tempo* de 100 negras por minuto, se calcula de manera correcta el tiempo de cada figura:

```
El tempo es de 100 negras/minuto ----> 1 negra son 0,6 segundos
=====
La figura: 16th tiene una duración de: 0.15 segundos
La figura: eighth tiene una duración de: 0.3 segundos
La figura: quarter tiene una duración de: 0.6 segundos
La figura: half tiene una duración de: 1.2 segundos
La figura: whole tiene una duración de: 2.4 segundos
```

Figura 5.8: Salida de la duración de cada figura

Por último, ya solo queda comprobar que dependiendo del tiempo de cada nota se le asigne la figura musical correcta, para hacer la comprobación se va a cambiar de frecuencia a distintas velocidades. Las frecuencias usadas son:

NOTA	FRECUENCIA
F5	698.46
B3	246.94
Bb4	466
Bb5	932.33
C1	32.7

Tabla 5.5: Frecuencias usadas para asignar notas musicales

La salida ha sido la siguiente, en la cual se muestra el nombre de la nota, el numero de ventanas, la duración que tiene y la figura musical a la que corresponde. Para asignar la figura musical se establece un margen de tiempo para cada figura.





La salida del algoritmo ha sido la esperada, con una pequeña particularidad, ya que las enarmonias no pueden ser identificadas, es decir, en la quinta nota en la partitura es un A y el sistema lo ha reconocido como un Bb. Esto pasa porque aunque el nombre de la nota es distinto, la frecuencia y su sonido es el mismo; el resto ha funcionado correctamente tal y como se muestra en la siguiente imagen:

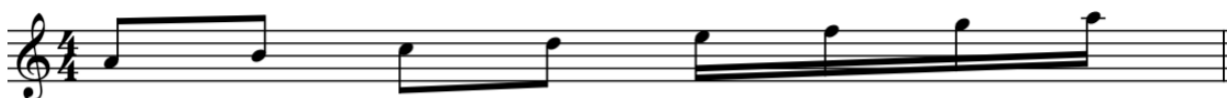
```
La nota leida es: A y la altura correspondiente es: 5  
La nota leida es: B y la altura correspondiente es: 3  
La nota leida es: E y la altura correspondiente es: 5  
La nota leida es: C y la altura correspondiente es: 5  
La nota leida es: Bb y la altura correspondiente es: 4  
La nota leida es: D# y la altura correspondiente es: 6  
La nota leida es: F y la altura correspondiente es: 5  
La nota leida es: G# y la altura correspondiente es: 3
```

**Figura 5.13:** Salida de la lectura de la alturas de las notas

### 5.3.2. EVALUACIÓN DE LA COMPROBACIÓN DE PARTITURAS

Una vez realizada la comprobación de que la lectura de los ficheros funciona correctamente, se da paso a la evaluación del algoritmo completo.

Para comprobar el algoritmo se han cargado dos archivos, en la primera prueba, los dos archivos eran iguales:



**Figura 5.14:** Partitura a comparar con si misma

La salida de comparar estos dos archivos iguales ha sido:

```
Ha tenido 0 errores, de 8 notas, su porcentaje de progreso es: 100.0%
```

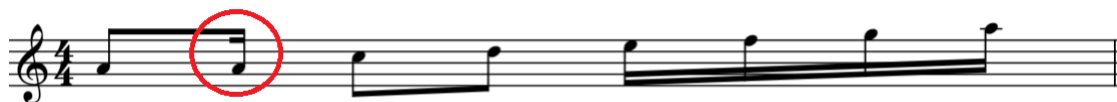
**Figura 5.15:** Salida de la comparación de dos partituras iguales

Como era de esperar, las dos partituras son iguales, entonces no deberías salir ningún error, además el nivel de progreso es el máximo.

Ahora bien, para comprobar si el algoritmo detecta fallos, se realizan una serie de pruebas añadiendo errores de forma incremental.

En estas pruebas se quiere comprobar en todo lo posible el funcionamiento correcto del algoritmo.

En la primera prueba solamente se realiza una pequeña modificación en la partitura a comparar con la original; en la segunda nota de la partitura se cambiará la nota y la figura:



**Figura 5.16:** Partitura modificada para ser comparada con la original

La salida que nos da el algoritmo es la siguiente:

```
En la partitura original la nota numero 2 era un: B y has tocado un: A  
En la partitura original la nota numero 2 era un/una: eighth y has tocado un/una: 16th  
Ha tenido 2 errores, de 8 notas, su porcentaje de progreso es: 75.0%
```

**Figura 5.17:** Salida de la comparación dos partituras con un pequeño cambio

Se puede comprobar, que se ha detectado el cambio en la partitura puesto que se muestra el error cometido y el porcentaje de progreso obtenido.

La siguiente prueba consistirá en cambiar más de una nota y más de una figura quedando la partitura modificada según se muestra:



**Figura 5.18:** Partitura modificada 2 para ser comparada con la original

En la salida del algoritmo, se reflejan todas las notas erróneas que hemos tenido, tanto en el nombre de la nota como en la altura de dicha nota. Además, como se ha observado anteriormente, se muestran también los errores de las figuras musicales que no coinciden (Figura 5.19):

```

En la partitura original la nota numero 1 era un/una: eighth y has tocado un/una: quarter
En la partitura original la nota numero 2 era un: B y has tocado un: A
En la partitura original la nota numero 2 era un/una: eighth y has tocado un/una: 16th
En la partitura original la nota numero 5 era un/una: 16th y has tocado un/una: quarter
En la partitura original la nota numero 6 era un: F5 y has tocado un/una: F3
Ha tenido 5 errores, de 8 notas, su porcentaje de progreso es: 37.5%

```

**Figura 5.19:** Salida de la comparación dos partituras con varios cambios

Ahora se comprueba que sucede cuando el número de notas no coincide, para ello se procede a quita una nota de la partitura original:



**Figura 5.20:** Partitura modificada 3 para ser comparada con la original

La salida del algoritmo ha sido la siguiente:

```

Ha tocado un numero distinto de notas que en la partitura original
En la partitura original la nota numero 5 era un: E y has tocado un: F
En la partitura original la nota numero 6 era un: F y has tocado un: G
Ha tenido 3 errores, de 7 notas, su porcentaje de progreso es: 57.142857142857146%

```

**Figura 5.21:** Salida de la comparación dos partituras y una de ellas con menos notas

# CONCLUSIONES

---

Este capítulo analizará los objetivos propuestos y los completados, así como las competencias definidas en el anteproyecto que se han adquirido durante el desarrollo de este T.F.G. Se tratará también hacia dónde podría dirigirse un trabajo futuro relacionado con dicho proyecto.

## 6.1. CONSECUCIÓN DE OBJETIVOS

EL objetivo principal del proyecto se ha cumplido con éxito: se ha desarrollado un sistema para apoyar el aprendizaje de un instrumento monódico mediante comparación de pasajes musicales.

Al haber alcanzado el objetivo principal, los objetivos parciales se han cumplido como se mostrará a continuación. Los objetivos parciales del proyecto son:

- **Implementación de un hilo de ejecución en segundo plano para la adquisición de la señal acústica proveniente del micrófono.**

Este objetivo parcial se ha conseguido con éxito, aunque se ha tenido que modificar la manera de obtener la señal acústica por un problema identificado durante el desarrollo del tercer bloque.

- **Construir el procedimiento de “enventanado” de la señal acústica de la interpretación grabada y su preprocesamiento.**

Este objetivo parcial se ha conseguido con éxito ya que se ha podido dividir la señal en pequeñas fragmentos para ser analizada después.

- **Diseño e implementación del algoritmo de *Pitch Tracking* y pruebas de validación.**

Este objetivo parcial se ha conseguido con éxito, aunque al realizar las pruebas de validación, se tuvo que reestructurar el primer objetivo parcial ya que la señal percibida inicialmente no era la correcta.

- **Diseño e implementación del algoritmo de demarcación de notas musicales concretas (discretización en semitonos, estimación de duración y segmentación de silencios, estimación de intensidad/volumen de cada nota). Pruebas de validación del algoritmo.**

Este objetivo parcial se ha conseguido con éxito, ya que, dada la frecuencia obtenida de cada ventana se obtiene la nota a la que corresponde y se le asigna una nota musical.

- **Diseño e implementación del algoritmo para mapear notas identificadas (y sus características) en notación *MusicXML*.**

Este objetivo parcial se ha conseguido con éxito, dado que podemos identificar todas las notas musicales de una partitura y almacenarlas en una lista para manipularlas. Además, se puede generar un archivo *MusicXML* dada una serie de notas.

- **Diseño e implementación del algoritmo para comparar las estructuras de árbol que albergan los archivos *MusicXML* (pasaje musical grabado Versus interpretación de referencia del pasaje musical).**

Este objetivo parcial se ha conseguido con éxito, puesto que podemos comparar el archivo original con la partitura generada por el sistema.

- **Proporcionar *feedback* al aprendiente de las diferencias encontradas tras la comparación de las interpretaciones de pasajes musicales.**

Este objetivo parcial se ha conseguido con éxito, ya que una vez que se ha hecho la comparación de las partituras, se le muestra al usuario los errores cometidos y el porcentaje de progreso de la obra.

## 6.2. COMPETENCIAS

Las competencias demostradas en este T.F.G. son las siguientes:

- **Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática.**
- **Capacidad para conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes.**
- **Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.**
- **Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.**
- **Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora**

## 6.3. DIFICULTADES

Durante el desarrollo de este T.F.G. se han planteado los siguientes problemas:

- Al principio el planteamiento de la aplicación disponía de un sistema codificado en java y se llevó a cabo la codificación en este lenguaje, pero al querer una aplicación multiplataforma se cambió el proyecto a *Flutter*, lo que llevó a la adaptación de lo codificado al nuevo lenguaje.
- Para el reconocimiento del audio, después de terminar la parte del *Pitch Tracking*, se observó un error en la frecuencia de muestreo, lo que llevó a usar la clase *AudioRecord* para subsanar el problema.



- El algoritmo del *Pitch Tracking* ha sido bastante más complejo de lo esperado por lo que su codificación se ha llevado más tiempo de lo planificado. Por otro lado, el problema en la frecuencia de muestreo ha llevado más tiempo del previsto identificarlo y subsanarlo.
- La codificación en el lenguaje *Dart* al ser un lenguaje todavía en desarrollo, ha supuesto problemas a la hora de implementar ciertas funcionalidades.

Estos han sido los problemas más influyentes durante el desarrollo del sistema.

## 6.4. TRABAJO FUTURO

Tras analizar los resultados obtenidos a partir del desarrollo de la herramienta y su rendimiento con usuarios reales, se plantean diversas mejoras y avances a implementar en un futuro para incrementar la eficacia y usabilidad del sistema:

- Creación de un sistema para renderizar la partitura para que el usuario pueda visualizar la partitura desde el sistema.
- Poder obtener más información de la comparación de las partituras, pudiendo mostrando de manera gráfica el lugar donde se han cometido los errores.
- Creación de un método de aprendizaje a partir de este sistema, para aprender un instrumento.

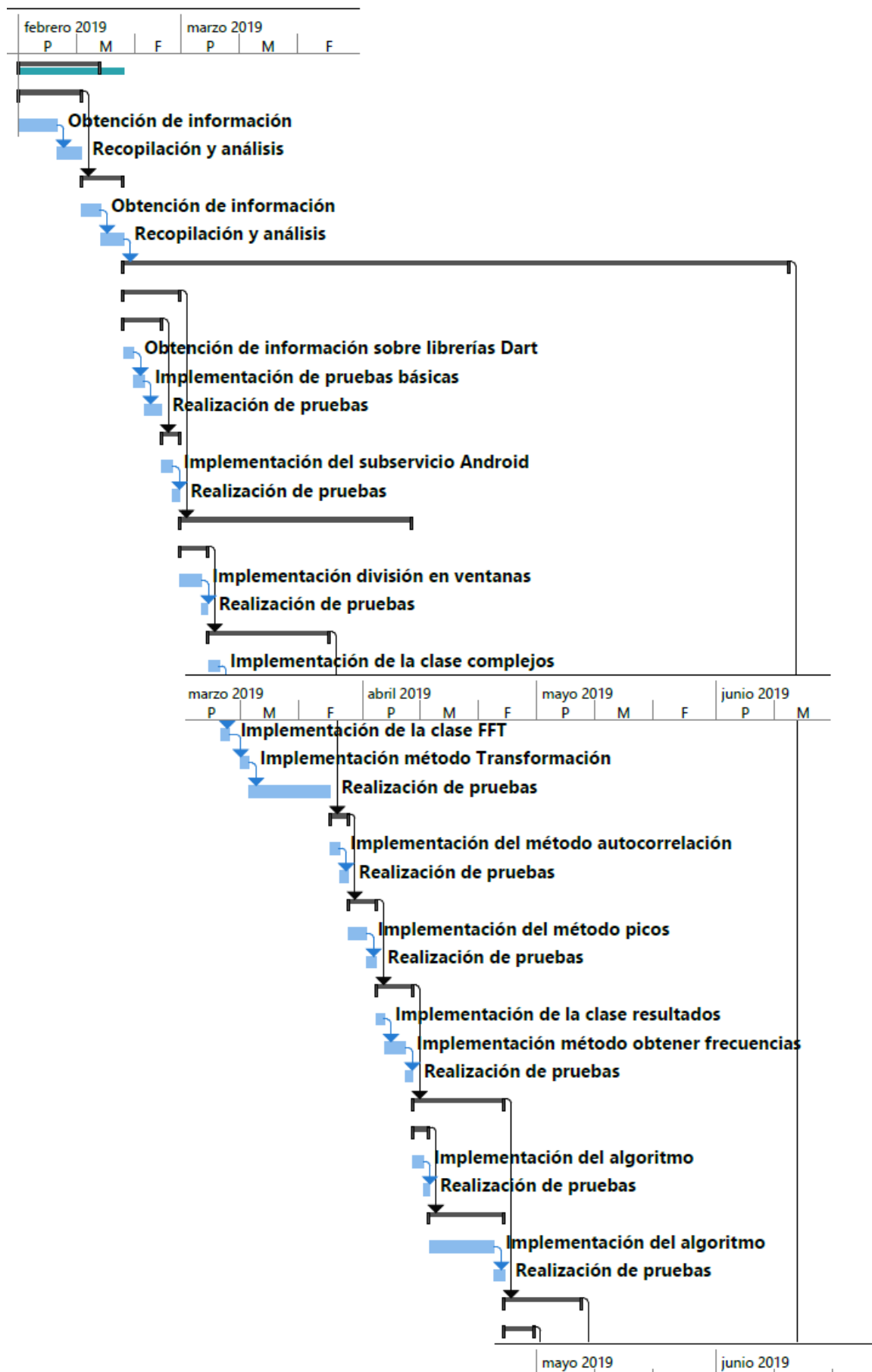


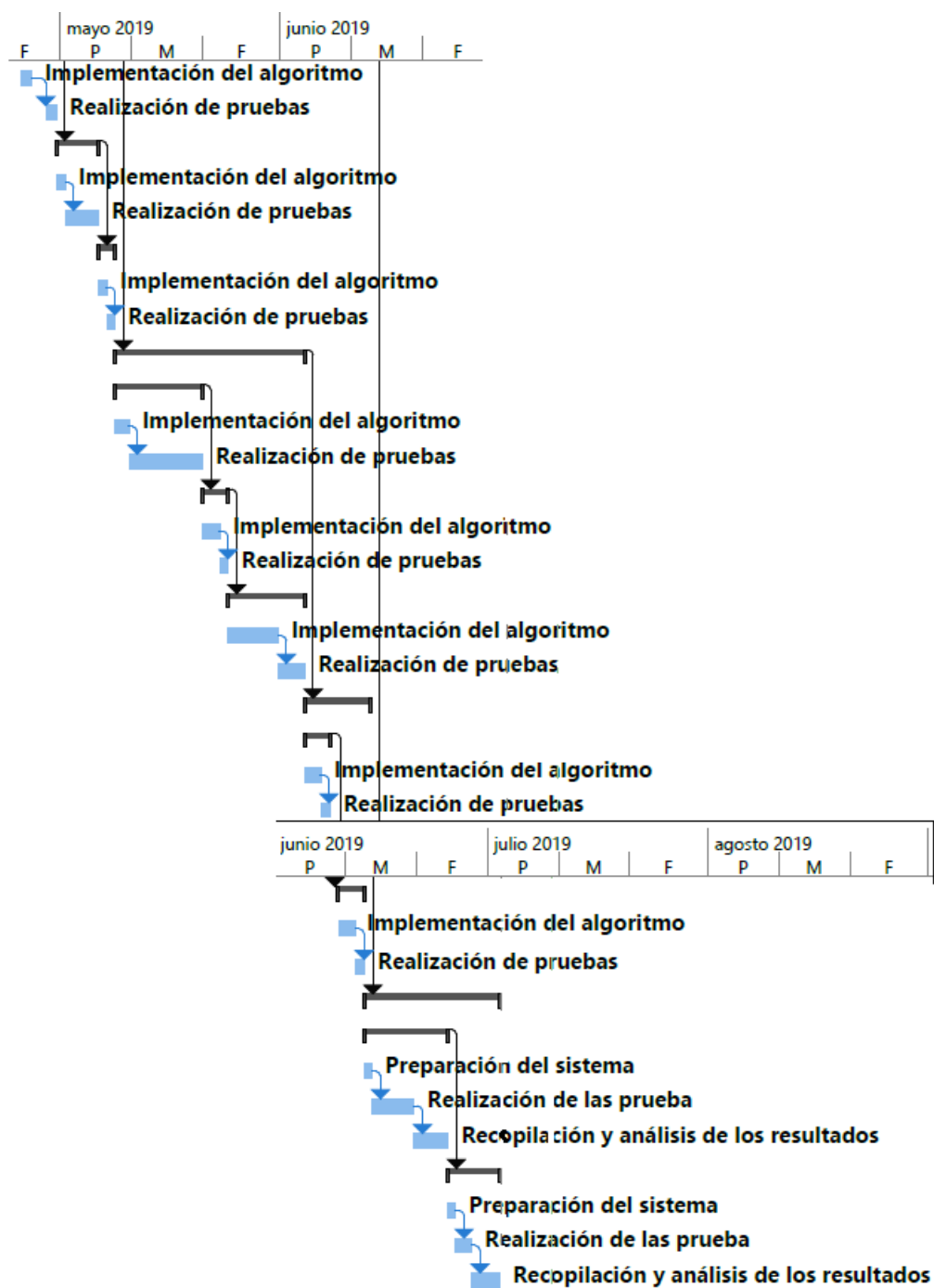
---

ANEXO A

# DIAGRAMA DE GANTT

---







---

ANEXO B

**ARCHIVO MUSICXML DE LA  
PARTITURA ORIGINAL DE LAS  
PRUEBAS**

---

```

1
2 <?xml version="1.0" encoding='UTF-8' standalone='no' ?>
3 <!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD_MusicXML_2.0_↵
   ↵ Partwise//EN" "http://www.musicxml.org/dtds/partwise.dtd">
4 <score-partwise version="2.0">
5   <work>
6     <work-title />
7   </work>
8   <identification>
9     <rights>Copyright    </rights>
10    <encoding>
11      <encoding-date>2019-07-04</encoding-date>
12      <encoder>Eduardo Mora</encoder>
13      <software>Sibelius 7.0.2</software>
14      <software>Direct export, not from Dolet</software>
15      <encoding-description>Sibelius / MusicXML ↵
        ↵ 2.0</encoding-description>
16      <supports element="print" type="yes" value="yes" ↵
        ↵ attribute="new-system" />
17      <supports element="print" type="yes" value="yes" ↵
        ↵ attribute="new-page" />
18      <supports element="accidental" type="yes" />
19      <supports element="beam" type="yes" />
20      <supports element="stem" type="yes" />
21    </encoding>
22  </identification>
23  <defaults>
24    <scaling>
25      <millimeters>210</millimeters>
26      <tenths>1200</tenths>
27    </scaling>
28    <page-layout>
29      <page-height>1697</page-height>
30      <page-width>1200</page-width>
31      <page-margins type="both">
32        <left-margin>72</left-margin>
33        <right-margin>72</right-margin>
34        <top-margin>72</top-margin>
35        <bottom-margin>72</bottom-margin>
36      </page-margins>
37    </page-layout>
38    <system-layout>
39      <system-margins>
40        <left-margin>22</left-margin>
41        <right-margin>0</right-margin>
42      </system-margins>
43      <system-distance>92</system-distance>
44    </system-layout>
45    <appearance>
46      <line-width type="stem">0.9375</line-width>
47      <line-width type="beam">5</line-width>
48      <line-width type="staff">0.9375</line-width>
49      <line-width type="light_barline">1.5625</line-width>
50      <line-width type="heavy_barline">5</line-width>
51      <line-width type="leger">1.5625</line-width>
52      <line-width type="ending">1.5625</line-width>
53      <line-width type="wedge">1.25</line-width>
54      <line-width type="enclosure">0.9375</line-width>
55      <line-width type="tuplet_bracket">1.25</line-width>
56      <line-width type="bracket">5</line-width>
57      <line-width type="dashes">1.5625</line-width>
58      <line-width type="extend">0.9375</line-width>
59      <line-width type="octave_shift">1.5625</line-width>

```



```

60 <line-width type="pedal">1.5625</line-width>
61 <line-width type="slur_middle">1.5625</line-width>
62 <line-width type="slur_tip">0.625</line-width>
63 <line-width type="tie_middle">1.5625</line-width>
64 <line-width type="tie_tip">0.625</line-width>
65 <note-size type="cue">75</note-size>
66 <note-size type="grace">60</note-size>
67 </appearance>
68 <music-font font-family="Opus Std" font-size="19.8425" />
69 <lyric-font font-family="Plantin MT Std" font-size="11.4715" />
70 <lyric-language xml:lang="es" />
71 </defaults>
72 <part-list>
73 <score-part id="P1">
74 <part-name>P1</part-name>
75 <score-instrument id="P1-I1">
76 <instrument-name> </instrument-name>
77 </score-instrument>
78 </score-part>
79 </part-list>
80 <part id="P1">
81 <measure number="1" width="190">
82 <print new-page="yes">
83 <system-layout>
84 <system-margins>
85 <left-margin>22</left-margin>
86 <right-margin>0</right-margin>
87 </system-margins>
88 <top-system-distance>218</top-system-distance>
89 </system-layout>
90 </print>
91 <attributes>
92 <divisions>256</divisions>
93 <key color="#000000">
94 <fifths>0</fifths>
95 <mode>major</mode>
96 </key>
97 <time color="#000000">
98 <beats>4</beats>
99 <beat-type>4</beat-type>
100 </time>
101 <staves>1</staves>
102 <clef number="1" color="#000000">
103 <sign>G</sign>
104 <line>2</line>
105 </clef>
106 <staff-details number="1" print-object="yes" />
107 </attributes>
108 <note color="#000000" default-x="162">
109 <pitch>
110 <step>A</step>
111 <octave>4</octave>
112 </pitch>
113 <duration>128</duration>
114 <instrument id="P1-I1" />
115 <voice>1</voice>
116 <type>eighth</type>
117 <stem>up</stem>
118 <staff>1</staff>
119 <beam number="1">begin</beam>
120 </note>
121 <note color="#000000" default-x="197">
122 <pitch>

```

```

123     <step>B</step>
124     <octave>4</octave>
125     </pitch>
126     <duration>128</duration>
127     <instrument id="P1-I1" />
128     <voice>1</voice>
129     <type>eighth</type>
130     <stem>up</stem>
131     <staff>1</staff>
132     <beam number="1">end</beam>
133 </note>
134
135 <note color="#000000" default-x="15">
136   <pitch>
137     <step>C</step>
138     <octave>5</octave>
139   </pitch>
140   <duration>128</duration>
141   <instrument id="P1-I1" />
142   <voice>1</voice>
143   <type>eighth</type>
144   <stem>down</stem>
145   <staff>1</staff>
146   <beam number="1">begin</beam>
147 </note>
148 <note color="#000000" default-x="50">
149   <pitch>
150     <step>D</step>
151     <octave>5</octave>
152   </pitch>
153   <duration>128</duration>
154   <instrument id="P1-I1" />
155   <voice>1</voice>
156   <type>eighth</type>
157   <stem>down</stem>
158   <staff>1</staff>
159   <beam number="1">end</beam>
160 </note>
161 <note color="#000000" default-x="85">
162   <pitch>
163     <step>E</step>
164     <octave>5</octave>
165   </pitch>
166   <duration>64</duration>
167   <instrument id="P1-I1" />
168   <voice>1</voice>
169   <type>16th</type>
170   <stem>down</stem>
171   <staff>1</staff>
172   <beam number="1">begin</beam>
173 </note>
174 <note color="#000000" default-x="113">
175   <pitch>
176     <step>F</step>
177     <octave>5</octave>
178   </pitch>
179   <duration>64</duration>
180   <instrument id="P1-I1" />
181   <voice>1</voice>
182   <type>16th</type>
183   <stem>down</stem>
184   <staff>1</staff>
185   <beam number="1">continue</beam>

```

```
186     </note>
187     <note color="#000000" default-x="140">
188         <pitch>
189             <step>G</step>
190             <octave>5</octave>
191         </pitch>
192         <duration>64</duration>
193         <instrument id="P1-I1" />
194         <voice>1</voice>
195         <type>16th</type>
196         <stem>down</stem>
197         <staff>1</staff>
198         <beam number="1">continue</beam>
199     </note>
200     <note color="#000000" default-x="167">
201         <pitch>
202             <step>A</step>
203             <octave>5</octave>
204         </pitch>
205         <duration>64</duration>
206         <instrument id="P1-I1" />
207         <voice>1</voice>
208         <type>16th</type>
209         <stem>down</stem>
210         <staff>1</staff>
211         <beam number="1">end</beam>
212     </note>
213 </measure>
214 </part>
215 </score-partwise>
```

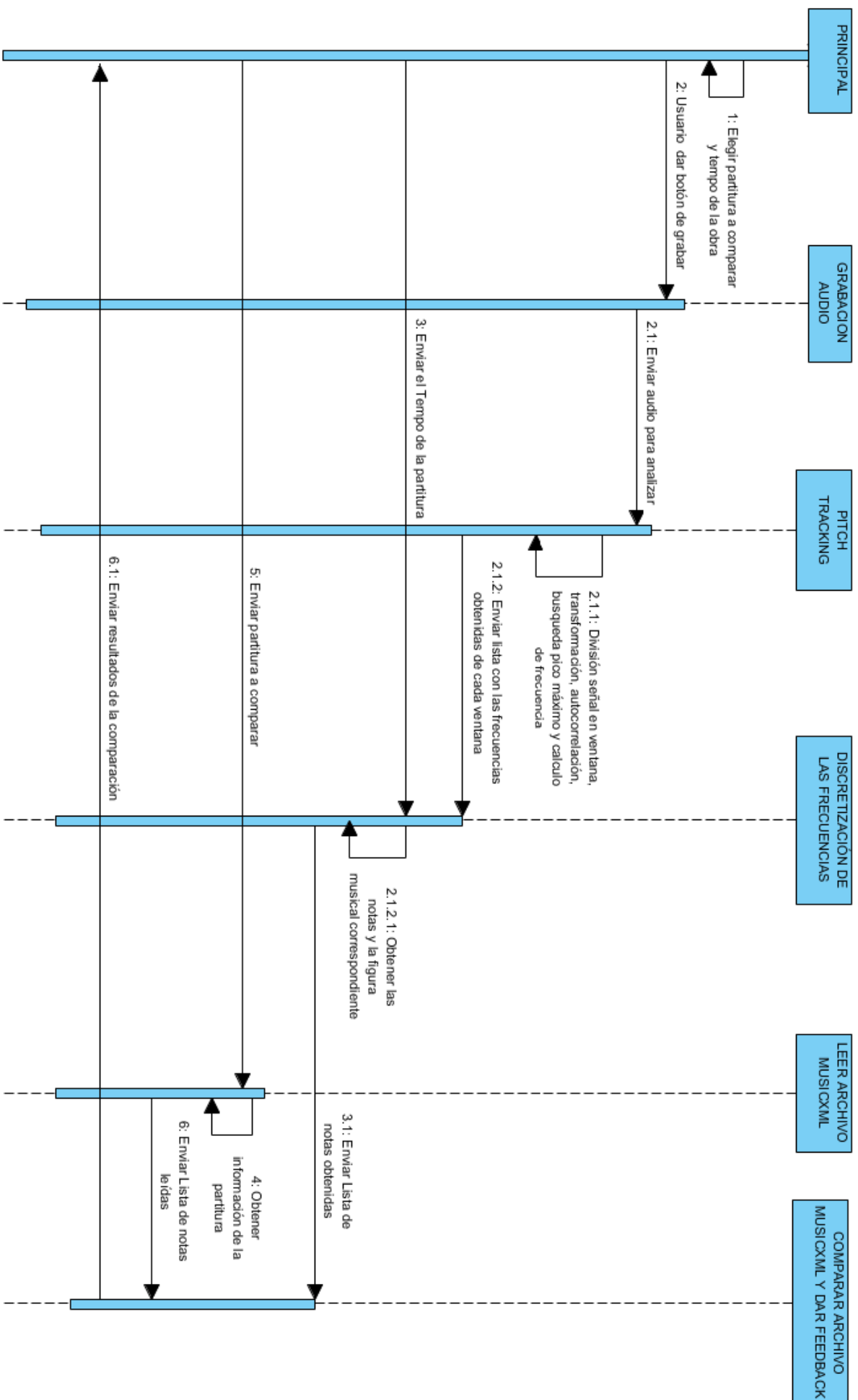


---

ANEXO C

# DIAGRAMA DE COMUNICACIÓN

---



# BIBLIOGRAFÍA

---

- [1] I.E.S. Mar de Alboran. «Lenguaje Musical: Figuras, Silencios, Compases, Puntillo Y Ligadura». En: (2009). URL: [http://maralboran.org/wikipedia/index.php/Lenguaje\\_musical:\\_figuras2C\\_silencios\\_2C\\_compases\\_2C\\_puntillo\\_y\\_ligadura](http://maralboran.org/wikipedia/index.php/Lenguaje_musical:_figuras2C_silencios_2C_compases_2C_puntillo_y_ligadura) (visitado 20-07-2019).
- [2] Anders Friberg Anders Elowsson. «Modeling Music Modality With a Key-Class Invariant Pitch Chroma». En: (2019). URL: <https://arxiv.org/abs/1906.07145> (visitado 20-07-2019).
- [3] Almudena M. Castro. «Música y Matemáticas. La Afinación Temperada». En: (2009). URL: <https://www.enchufa2.es/archives/musica-y-matematicas-la-afinacion-temperada.html> (visitado 20-07-2019).
- [4] Meyer Cooper. *OP.CIT.* 2000.
- [5] Ronny Demera. «Metodologías Tradicionales Vs Metodologías Ágiles.» En: (2018). URL: <https://tech.tribalyte.eu/blog-metodologias-tradicional-vs-agil> (visitado 20-07-2019).
- [6] Emir Demirel. «Automatic Chord-Scale Recognition Using Harmonic Pitch Class Profiles». En: (2019). URL: [https://repositori.upf.edu/bitstream/handle/10230/41706/demirel\\_smc\\_auto.pdf?sequence=1&isAllowed=y](https://repositori.upf.edu/bitstream/handle/10230/41706/demirel_smc_auto.pdf?sequence=1&isAllowed=y) (visitado 20-07-2019).
- [7] «ISO 16:1975: Standard tuning frequency». En: (1975). URL: <https://www.iso.org/standard/3601.html> (visitado 20-07-2019).
- [8] P. Gómez y J. Bemol J. Bobadilla. *La Transformada De Fourier. Una Visión Pedagógica*. Escuela Técnica Superior de Ingeniería de Sistemas Informáticos de la Universidad Politécnica de Madrid, 1999. URL: <https://www.raco.cat/index.php/EFE/article/download/144488/256868> (visitado 20-07-2019).
- [9] O’Keefe R. M. Linda P. J. *Experiences with an Expert Systems Prototyping Methodology*. Expert Systems, 1994.
- [10] Clemens Kühn. *Tratado de la Forma Musical*. Pg: 17 y 18. 1992.
- [11] Patricio Letelier. «Metodologías ágiles para el desarrollo de software: Extreme Programming». En: (2005). URL: <http://www.cyta.com.ar/ta0502/v5n2a1.htm> (visitado 20-07-2019).
- [12] «Manifiesto por El Desarrollo Ágil De Software». En: (2001). URL: <https://agilemanifesto.org> (visitado 20-07-2019).
- [13] Universidad de Manizares. «Prototipo Incremental». En: (2006). URL: [www.acofi.edu.co](http://www.acofi.edu.co) (visitado 20-07-2019).
- [14] Adolfo Nuñez. *Informática y Electrónica Musical*. Editorial PARANINFO, 1993.
- [15] Gustavo Santos. «Inteligencia Artificial y Matemática Aplicada». En: (2002).
- [16] Liming Shi. «Bayesian Pitch Tracking Based on the Harmonic Model». En: (2019). URL: [https://www.researchgate.net/publication/333238415\\_Bayesian\\_Pitch\\_Tracking\\_Based\\_on\\_the\\_Harmonic\\_Model](https://www.researchgate.net/publication/333238415_Bayesian_Pitch_Tracking_Based_on_the_Harmonic_Model) (visitado 20-07-2019).