

PROYECTO FINAL DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE GRAFICADOR DE TABLATURAS DE BAJO ELÉCTRICO ESTANDAR EN TIEMPO REAL

(design and implementation of a software plotter of tabs for standard electric bass in real time)

**Titulación:**

Ingeniería en sistemas y computación

**Autor:**

Alejandro Plaza Parra

**Directora:**

ING. Ivonne Castaño Osorio

**Universidad Tecnológica de Pereira**

Pereira, Risaralda, Colombia

Julio, 2019

This page intentionally left blank.

Abstract

This project explains the design and implementation of a software that makes the graph of the tablature of notes entered by means of a standard electrical bass in real time. For this purpose, you must obtain the value of the frequency in the notes entered. Frequency estimation is performed using the Fast Fourier Transform, processing the sound of the notes in real time and comparing it to a fixed threshold to determine the position of the note on the electric bass mast. The data obtained within this process of decomposition and comparison are used as parameters so that the software can graph the respective simplified tablature, showing on screen the string and fret where that note was generated. These comparison frequencies were previously found by the same decomposition method, since these values should not vary if the instrument is tuned. The results show a good approximation in the graphical representation of entered melodies, being able to correctly generate 75% of the positions of these.

This page intentionally left blank.

Resumen

Este proyecto explica el diseño e implementación de un software que realice la gráfica de la tablatura de notas ingresadas mediante un bajo eléctrico estándar en tiempo real. Para este propósito, se debe obtener el valor de la frecuencia en las notas ingresadas. La estimación de la frecuencia se realiza mediante el uso de la transformada rápida de Fourier, procesando el sonido de las notas en tiempo real y comparándola con un umbral fijo para determinar la posición de la nota en el mástil del bajo eléctrico. Los datos obtenidos dentro de este proceso de descomposición y comparación son utilizados como parámetros para que el software pueda graficar la respectiva tablatura simplificada, mostrando en pantalla la cuerda y el traste en donde esa nota fue generada. Estas frecuencias de comparación fueron halladas previamente mediante el mismo método de descomposición, dado que estos valores no deben variar si el instrumento esta afinado. Los resultados muestran una buena aproximación en la representación gráfica de melodías ingresadas, pudiendo generar correctamente el 75% de las posiciones de estas.

This page intentionally left blank.

Agradecimientos

Agradezco a muchas personas que este trabajo haya podido ser realizado con éxito es probable que se me olvide mencionar a alguien, así que pido disculpas de antemano.

En primer lugar, quiero agradecerles a esos maestros que me impartieron clases durante estos años, generando en mi de manera correcta el interés por la carrera que necesitaba y me impulso a terminarla.

Le agradezco a Ivonne Castaño su sabiduría y sus consejos en cada momento de la evolución de este proyecto, la orientación, todas las aportaciones en mi crecimiento tanto personal como profesional y en la confianza depositada en mí.

Agradezco a Diego Fernando Hincapié por su colaboración continua en la implementación técnica del proyecto, por no abandonarme y estar pendiente de mis avances.

Agradezco a mis compañeros de carrera, especialmente a Natalia Franco, Fernán Cañas, Juan David López, Sebastián Sánchez, Edgar Martínez por compartir conmigo esta etapa de mi vida, por brindarme su amistad y apoyo, además de los momentos inolvidables que vivimos.

Agradezco a mis amigos, especialmente a Mateo Bedoya, Sergio Villaquiran y José Santa quienes con sus conocimientos sobre la música me instruyeron y ayudaron a generar un proyecto basado en este arte, por los buenos momentos y experiencias, y por aligerar mis malos momentos con su invaluable ayuda y consejos.

Agradezco a Nicolás Ruiz por sus conocimientos, ayuda y apoyo a través de la carrera, por orientarme en la dirección correcta al resolver problemas lógicos de programación.

También quiero agradecer a Carlos Augusto Meneses y a Marta Lucy Estrella por su disposición, orientación y ayuda en las etapas finales de la carrera

A mis padres Hernando Plaza y Lorena Parra porque sin ellos no estaría realizando mi sueño, por el esfuerzo, apoyo y confianza depositados en mí.

This page intentionally left blank.

Índice

**Abstract iii**

**Resumen v**

**Agradecimientos vii**

**Índice x**

**Índice de Tablas xii**

**Índice de figuras xiii**

**1 Introducción 1**

* 1. Objetivos 2
     1. Objetivo General 2
     2. Objetivos específicos 2
  2. Limitaciones del proyecto 2
     1. Timbre 2
  3. Estructura de la memoria 3

1. **Estado del arte 4**
   1. Sonido 4
   2. Herramientas actuales 6
2. **Análisis 7**
   1. Catálogo de requisitos 8
      1. Requisitos funcionales 8
         1. Uso del software 8
         2. Sistema 9
      2. Requisitos no funcionales 9
   2. Metodología de desarrollo 10
3. **Diseño 12**
   1. Arquitectura del software 12
      1. Diseño Modelo – Vista – Controlador 13
      2. Comunicación entre campos 13
   2. Diagramas UML 14
      1. Diagrama de casos de uso 15
      2. Diagrama de clases 17
      3. Diagrama Entidad-Relación 18
      4. Diagrama de despliegue 19
      5. Diagrama de actividad 20
   3. Tecnologías utilizadas 23
   4. Diseño interfaz de usuario 24
4. **Implementación 25**
   1. Desarrollo del proyecto 25
   2. Conclusiones del desarrollo 31
5. **Pruebas 33**
6. **Conclusiones 35**
7. **Trabajo futuro 36**

**Bibliografía 37**

**Anexo 38**

**Manual de usuario 39**

Índice de Tablas

4.1 Casos de uso 16-17

4.2 Librerías necesarias dentro de python 19

Índice de Figuras

2.1 Una señal genérica se trasforma por una sumatoria de señales sinusoidales 5

2.2 Transformada de Fourier 5

2.3 Transformada rápida de Fourier 5

3.1 Etapas del modelo en cascada 11

4.1 Modelo –Vista- Controlador 14

4.2 Diagrama de casos de uso 15

4.3 Diagrama de clases 18

4.4 Diagrama Entidad-Relación 18

4.5 Diagrama de despliegue 19

4.6 Diagrama de actividad, caso de uso PassiveListen 20

4.7 Diagrama de actividad, caso de uso Comparar 21

4.8 Diagrama de actividad saso de uso Graficar 22

4.9 Diagrama de actividad, caso de uso GraficarL 22

4.10 Mockup inicio 24

4.11 Mockups para el graficado 24

5.1 Clases 25

5.2 Método Rms 26

5.3 Método Passivelisten y Cargar 26-27

5.4 Método Comparar y rangos de frecuencia 28

5.5 Método Graficar 29

5.6 Método Grabar y Load 30

5.7 Método Llenar\_lista 30

5.8 Método GraficarL 31

6.1 Notas de un bajo eléctrico 33

A1 Pantalla de inicio 39

A2 Ventana al presionas “graficar” 39

A3 Resultado “graficar” 40

A4 Re visualizar vacía 41

A5 Re visualizar 41

1

**Introducción**

El objetivo del presente proyecto es crear un intérprete graficador que permita generar y guardar tablature de manera autónoma y digital. Este desarrollo permitirá analizar los sonidos de las notas emitidas por el bajo eléctrico para descomponerlos en frecuencias y poder determinar en qué posición o traste fue generada y en qué cuerda.

Para poder llevar a cabo el desarrollo del software se debe tener en cuenta que el mercado cuenta con múltiples tipos y marcas de bajos eléctricos que varían tanto en número de cuerdas como en la tonalidad que manejan, los bajos eléctricos que se usan más comúnmente son aquellos denominados estándar, ya que permiten el dinamismo que se necesita si se está empezando en el mundo musical. Estos serán el enfoque del proyecto ya que, al tener un mayor grado de aceptación, permitirá que el software tenga una mayor expansión de uso.

Además, integra una interfaz gráfica amigable con el usuario, permitiendo su uso intuitivo acertado. El bajo eléctrico estándar cuenta con cuatro cuerdas y afinación Sol, Re, La, Mi.

* 1. Objetivos
     1. Objetivo general

El objetivo general de este proyecto es el diseño e implementación de un software que realice la gráfica de la tablatura de notas generadas mediante un bajo eléctrico en tiempo real; el cuál se llevará a cabo mediante los diferentes objetivos específicos detallados a continuación.

* + 1. Objetivos específicos

Los objetivos específicos para lograr el objetivo general son los siguientes:

* Elaborar el plan de proyecto
* Levantar requerimientos.
* Definir los rangos de frecuencia de las notas del bajo eléctrico.
* Realizar el análisis del sistema.
* Realizar el diseño del sistema.
* Implementar el proyecto.
* Realizar las pruebas.
  1. Limitaciones del proyecto
     1. Timbre

Los bajos eléctricos se componen por varios elementos, los cuales definen su diseño y su timbre al tocar, uno de los componentes que alteran el sonido es el calibre de las cuerdas que se usan, ya que estas de entrada modifican la nota que producen dependiendo de su entorchado y el material del que estén echas, los bajos de cuatro cuerdas de serie vienen con las afinaciones estándar (Sol, Re, La, Mi) y con cuerdas de entorchado semiliso, permitiendo versatilidad.

Las pastillas, que funcionan captando las vibraciones de las cuerdas y transmitiéndolas al amplificador, también alteran el timbre, ya que dependiendo de la distribución de las mismas captaran en mayor o menor medida estas vibraciones, los bajos estándar se componen con al menos 2 de estas pastillas cubriendo las cuatro cuerdas del mismo.

El alma del bajo eléctrico, es una varilla delgada de acero que se encuentra entre el diapasón del bajo y el mástil, se emplea para modificar la curvatura de este mástil, que es donde se encuentran las cuerdas, modificar la curvatura ocasionara que el instrumento cambie su timbre, existiendo un estándar de medición que indica la curvatura optima del instrumento.

Existen a su vez diferentes maneras de tocar un bajo eléctrico, que alteran el timbre de la nota siendo la más popular el “fingerstyle” que consiste en pulsar las cuerdas con los dedos índice y corazón de la mano derecha, pero no es la única su variación más cercana es pulsar las cuerdas con ayuda de un “pick” o “púa”, y la variación más agresiva usada es el “slap”, que consiste en golpear directamente la cuerda con el pulgar, cada técnica genera distorsión adicional en las notas, por lo que las pastillas no captan de manera muy clara las vibraciones, ocasionando un timbre distante entre cada técnica.

Al existir tantos factores que podrían alterar el funcionamiento del software se ha decidido delimitar bastante el alcance del proyecto, definiendo entonces que se centrara en un bajo con cuatro cuerdas (Sol, Re, La, Mi), con entorche semiliso, que contengan mínimo 2 pastillas completas, que su alma esté debidamente acomodada y que la técnica utilizada sea “figerStyle”.

* 1. Estructura de la memoria

En el capítulo 2 se mencionan las investigaciones realizadas sobre implementaciones de software y herramientas utilizadas en el ámbito musical. En el capítulo 3 se dan a conocer análisis inicial del desarrollo y la metodología usada para desarrollar el software. En el capítulo 4 se explica el diseño de la aplicación, conociendo las especificaciones y requisitos de la misma. En el capítulo 5 se expone el procedimiento de implementación paso a paso. En el capítulo 6 se exponen los resultados experimentales. Finalmente se representan las conclusiones de este proyecto y los posibles trabajos futuros.

2

**Estado del arte**

En este capítulo se dan a conocer los trabajos y publicaciones previos relacionados con los objetivos de este proyecto. En la sección 2.1 se mencionan trabajos relacionados al manejo de sonidos, y en la sección 2.2 se muestran investigaciones sobre herramientas actuales que usan este mismo principio.

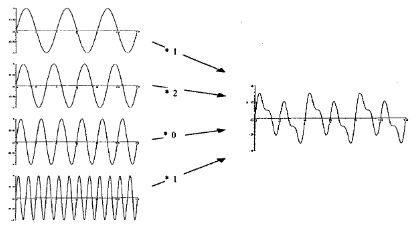
* 1. Sonido

El sonido audible consiste en ondas sonoras y ondas acústicas que se producen cuando las oscilaciones de la presión de aire, son convertidas en ondas mecánicas en el oído y percibidas por el cerebro, las cuales pueden viajar a través de cualquier medio sólido, liquido o gaseoso.

En la definición del sonido también debe considerarse tanto el fenómeno físico como el sicoacústico, ya que bajo la ausencia de un oyente puede existir un evento sonoro, pero no el evento auditivo.

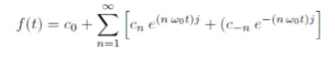
El sonido se compone de varios componentes como el tono, el cual es determinado por la frecuencia fundamental de las ondas sonoras medidas en ciclos, este es quien nos permite distinguir entre sonidos graves y agudos. La duración, que no es más que el tiempo de propagación de la onda. El timbre acompaña a la frecuencia y es el atributo que nos permite diferenciar entre dos sonidos con igual frecuencia

Los autores proponen la utilización de la transformada de Fourier para entender los sonidos y la música de una manera más matemática ya que esta expresa que toda función periódica puede expresarse como la suma infinita de funciones seno o coseno que son múltiplos enteros n de la frecuencia, hallando su frecuencia fundamental, y a su vez cada término de seno y coseno se le conocen como armónicos.

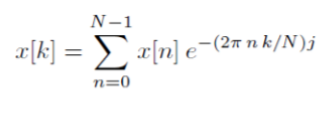


**Figura 2.1:** Una señal genérica se transforma por una sumatoria de señales sinusoidales

Logrando condensar su algoritmo se logra que realice los mismos cálculos, pero en un menor tiempo, dando lugar a la transformada rápida de Fourier. Estos infieren pues que la forma matemática más sencilla para la síntesis auditiva es utilizar las series de Fourier, pudiendo hallar la frecuencia del sonido basándose solo en su amplitud y su fase.



**Figura 2**.**2**: Transformada de Fourier



**Figura 2.3**: Transformada rápida de Fourier.

* 1. Herramientas actuales

En la actualidad el uso y tratamiento de señales tiene bastantes usos prácticos.

Algunos autores hacen mención de que el estudio de la síntesis de los sonidos de los instrumentos musicales es un tema motivador que interesa a los estudiantes de ciencias básicas, ingenierías y música ya que entrega una aproximación al tema desde diferentes perspectivas ya que el modelamiento de transmisión de ondas puede ser abordado desde la parte física y matemática a la vez, permitiendo versatilidad de aprendizaje sobre estos temas.

Algunos abordan este tema basándose en el funcionamiento de los afinadores actuales ya que el funcionamiento de estos dispositivos funciona de manera similar a lo planteado anteriormente, basados en que los sonidos de las cuerdas al aire de los instrumentos de cuerda tienen su propia frecuencia fundamental, la cual comparándola con registros anteriormente diseñados puede comunicarle al usuario cuando su instrumento esta afinado.

Usando este principio de tratamiento de señales y con algoritmos de recomposición y tratamiento de datos es como google implementa su búsqueda por voz implantado en nuestros teléfonos Smartphone, ya que utilizan la descomposición de los sonidos en ondas más manejables para compararlas con su inmensa base de datos, una red neuronal es la que se encarga de encontrar las similitudes para entender que es lo que se le está solicitando que busque, por lo que en ultimas somos los usuarios quienes no encargamos de entrenar a la red para que funcione más eficientemente.

En el campo de la robótica las utilizaciones de descomposición de sonidos en ondas manejables permiten que en la actualidad el androide llamado sophia pueda entender y dialogar con personas de su entorno, realizando tanto síntesis de lo que escucha como su proceso inverso, pudiendo así hacer uso de señales computacionales para comunicarse eficazmente.

3

**Análisis**

En este capítulo se presenta la fase de análisis, parte inicial de cualquier proyecto de software, donde se definen los requisitos y se enseña una vista global de la arquitectura que se desea implementar en el sistema. Ya que en el siguiente capítulo se tendrá en cuenta este apartado como base para el diseño de todos los aspectos de la aplicación. Por ello que la fase de análisis de se suma importancia para el futuro del software. Aquí es donde se sentarán las bases del proyecto.

El proceso a seguir se basa en primer lugar en una buena definición de requisitos y en elegir las metodologías más apropiadas para el desarrollo apropiado del proyecto.

La arquitectura del sistema tendrá el formato Modelo – Vista – Controlador constituyendo las funcionalidades de cada uno de estos componentes.

A continuación, se desglosan las funcionalidades y las características a modo de catálogo de requisitos, teniendo en cuenta tanto los funcionales como no funcionales

* 1. Catálogo de requisitos

El catálogo de requisitos es la especificación del comportamiento que se espera de cualquier proyecto de software, estudiando las necesidades de los usuarios se ha predefinido una serie de requisitos que se consideran indispensables para el proyecto. A continuación, se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

* + 1. Requisitos funcionales

Los requisitos funcionales describen todas las interacciones que tendrán los usuarios con el software.

* + - 1. Uso del software

**RF1: Graficar**

1. El software deberá iniciar la operación únicamente cuando el usuario lo indique.
2. Una vez iniciado las operaciones el sistema empezara a escuchar las notas que el usuario genere.
3. A medida que va escuchando el sistema va guardado el registro en un archivo ‘Audio.wav’
4. El sistema carga el archivo ‘Audio.wav’ y aplica la transformada rápida de Fourier para hallar un magnitud y fase.
5. El sistema encuentra la frecuencia fundamental a partir de la magnitud y fase.
6. El sistema compara la frecuencia con los rangos definidos y guardara su posición en un vector de posiciones.
7. El sistema guarda las posiciones en un archivo ‘save.bin’
8. El sistema grafica en la ventana las notas una vez se acabe el tiempo.

**RF2: Re visualizar** El sistema deberá volver a graficar la última tablatura generada

1. El sistema cargara el archivo ‘save.bin’
2. El sistema graficara en la ventana las notas una vez que haya cargado las posiciones.
   * + 1. Sistema

**RF3: Pantalla principal**

1. La pantalla principal de la aplicación constara de tres botones:
   1. **Graficar:** El sistema mostrará la ventana donde se graficará la tablatura.
   2. **Re visualizar:** El sistema mostrará la ventana con las últimas posiciones guardadas
   3. **Salir:** El sistema terminara las operaciones y se cerrara.
      1. Requisitos no funcionales

Requisitos complementarios que especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento.

**RNF1: Portabilidad**

1. Disponibilidad de operación sobre cualquier portátil con micrófono y python.

**RNF2: Interfaz y usabilidad**

1. El software debe constar de una interfaz sencilla, atractiva e intuitiva.
2. La tablatura generada debe ser de fácil entendimiento.

**RNF3: Rendimiento**

1. Se esperan tiempos de respuesta no superiores a 2 segundos en la petición de graficación una vez se acabe el tiempo de escucha.
2. Se esperan tiempos de respuesta no superiores a 2 segundos de carga para re visualizar la tablatura.
   1. Metodología del desarrollo

Se trata de un marco de trabajo usado para estructurar, planificar y controlar el proceso del desarrollo del software. Existe una gran cantidad de métodos diferenciados por sus fortalezas y debilidades. Estas se basan principalmente en:

-Herramientas, modelos y métodos para ejecutar dicho proceso de desarrollo

Cada metodología de desarrollo tiene su propio enfoque, entre ellos el más utilizado en el modelamiento de diseño de software es el lenguaje de modelado UML el cual es un lenguaje visual común, semántico y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas. En general los diagramas UML describen los limites, la estructura y el comportamiento de los objetos que contiene el sistema. Entre los más conocidos (y que se implementaron en el proyecto) están:

-Diagrama de clases

-Diagrama de casos de uso

-Diagrama de actividades

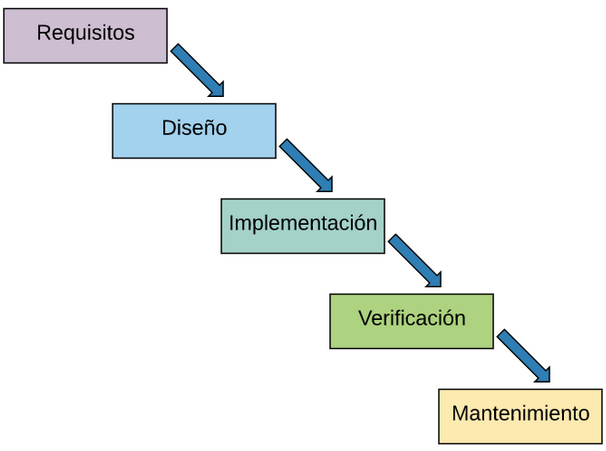
-Diagrama de clases de diseño

-Diagrama de despliegue

-Diagrama entidad relación

Otro modelado a tener en cuenta en el modelo en cascada, el cual nos indica que el proceso de desarrollo es secuencial, describiendo este último como un conjunto de etapas que se ejecutan

una tras otra. Se representan una encima de la otra y siguiendo el flujo de ejecución de arriba hacia abajo, como una cascada.



**Figura 3.1:** Etapas del modelo en cascada.

4

**Diseño**

En el presente capítulo se describe todo el proceso de diseño del software. Se Ha realizado un diseño que abarca todos los requisitos descritos en el apartado **3.1 Catálogo de requisitos.**  El diseño proporciona una idea completa del software desarrollado en el proyecto. Además, se justifican las decisiones tomadas para el desarrollo. Por ello, el capítulo se divide en 4.1 Arquitectura de la aplicación, 4.2 Diagramas UML, 4.3 Tecnologías utilizadas y 4.4 Diseño de la interfaz de usuario.

El diseño del software se ha ido transformando a medida que se avanzaba el proyecto, pero se hará presente solo el diseño final en este documento.

4.1 Arquitectura del software

Como se menciona en el capítulo 3- Análisis, la arquitectura del software se basa en el Modelo- Vista- Controlador, a continuación, se explica mejor el funcionamiento de esta arquitectura y como se implementó.

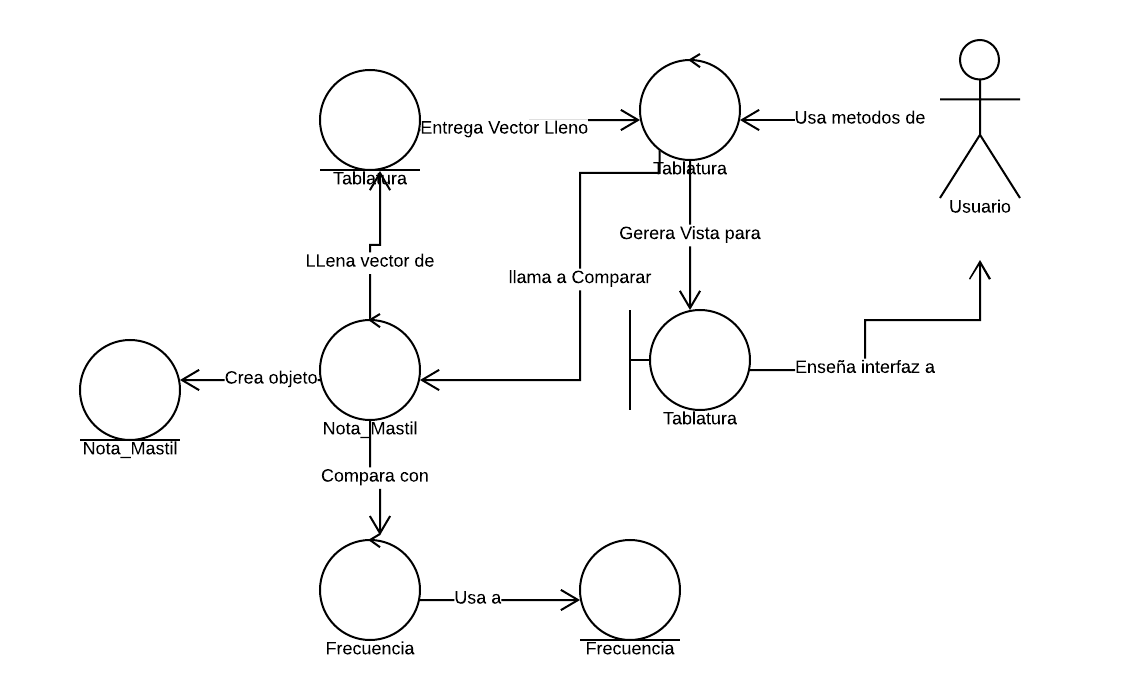
* + 1. Diseño Modelo- Vista- Controlador

El diseño Modelo – Vista – Controlador, o también llamado diseño MVC, está orientado a objetos dentro de la programación del software. Este permite separar la programación del sistema en tres campos. El Modelo contiene información del sistema, como pueden ser: las variables, las especificaciones, etc. El Controlador se encarga de como presentar lo que se encuentra en el modelo al usuario en una interfaz amigable e intuitiva. Finalmente, la Vista es la encargada de enseñar lo que le ordene el controlador.

* + 1. Comunicación entre campos

La comunicación entre los tres campos no es absoluta. El Controlador puede comunicarse con el Método sin ningún problema, y puede obtener toda la información que necesite cuando lo requiera, ya que el Controlador es el encargado de poner en pantalla el Modelo. También el Controlador puede comunicarse con la Vista para poder gestionar la interfaz de usuario. Mientras que la comunicación entre el Modelo y la Vista, idealmente no se debería contemplar, debido a que el Modelo es independiente de la interfaz de usuario. También se podría realizar un Modelo que se adaptara a la vista, pero esto afectaría a la reusabilidad del código y haría más complicada la búsqueda de errores durante la depuración.

Además, la comunicación entre Vista y Controlador se puede realizar de tres maneras. Una forma es que la Vista a través de acciones pueda comunicarle al Controlador lo que el usuario está realizando en la interfaz. La segunda forma es que el Controlador tenga el control de lo que pase en la vista través de delegados. Por último, análogo a la anterior, el Controlador gestiona la información que quiera mostrar la vista a través de una fuente de datos. La comunicación entre el Modelo y el Controlador noes directa. Cuando el Modelo necesita actualizar información debe utilizar notificaciones, para notificarle al Controlador los nuevos cambios.



**Figura 4.1:** Modelo – Vista - Controlador

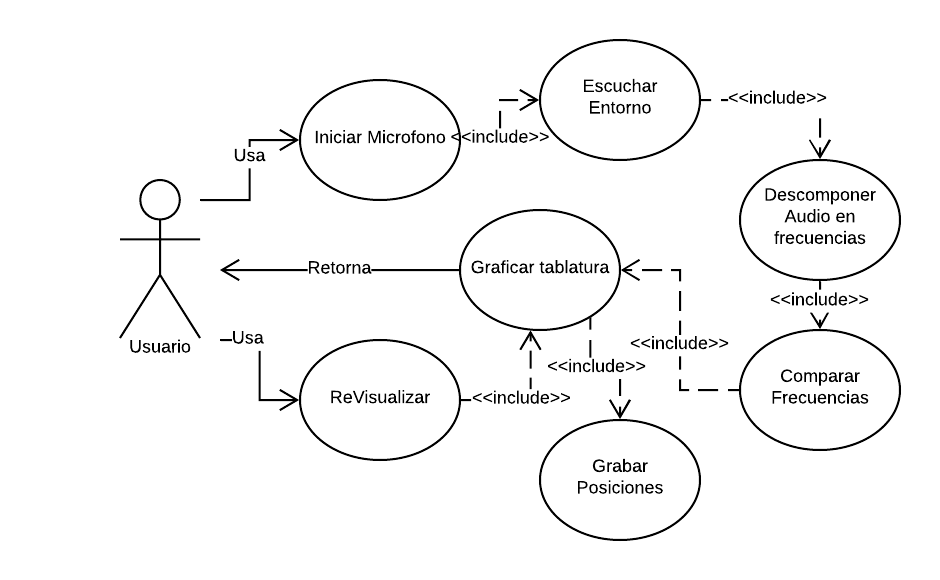
En el diagrama anterior apreciamos que el usuario se comunica con el sistema únicamente mediante el Control de tablatura, el cual a su vez se comunica con los demás modelos de manera ordenada, para obtener los datos necesarios, una vez que esta comunicación es satisfactoria y se acabe el tiempo de escucha, el Modelo tablatura le indica a la Vista tablatura que enseñe sus resultados al usuario mediante una interfaz gráfica.

* 1. Diagramas UML

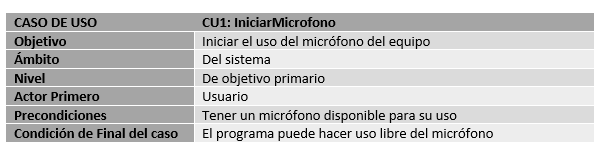
Como se mencionó en el capítulo 3 Análisis, se llevaron a cabo 2 modelamientos, que, aunque diferentes se complementan de la manera que se requería, a continuación, se mostraran los resultados de realizar el análisis evidenciado en los diagramas UML.

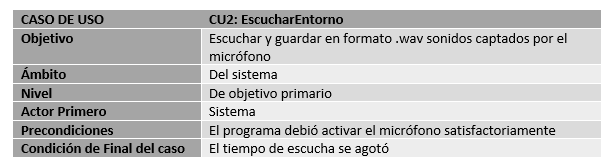
* + 1. Diagrama de casos de uso

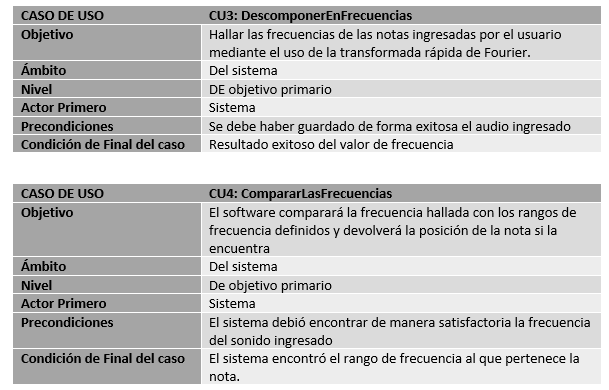
El diagrama de casos de uso representa la forma en como un usuario (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan para cumplir con las funcionalidades del sistema. A continuación, el diagrama para el proyecto propuesto.

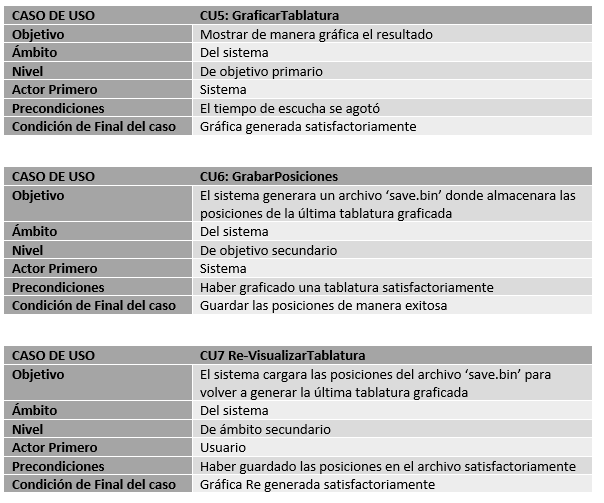


**Figura 4.2:** Diagrama de casos de uso.





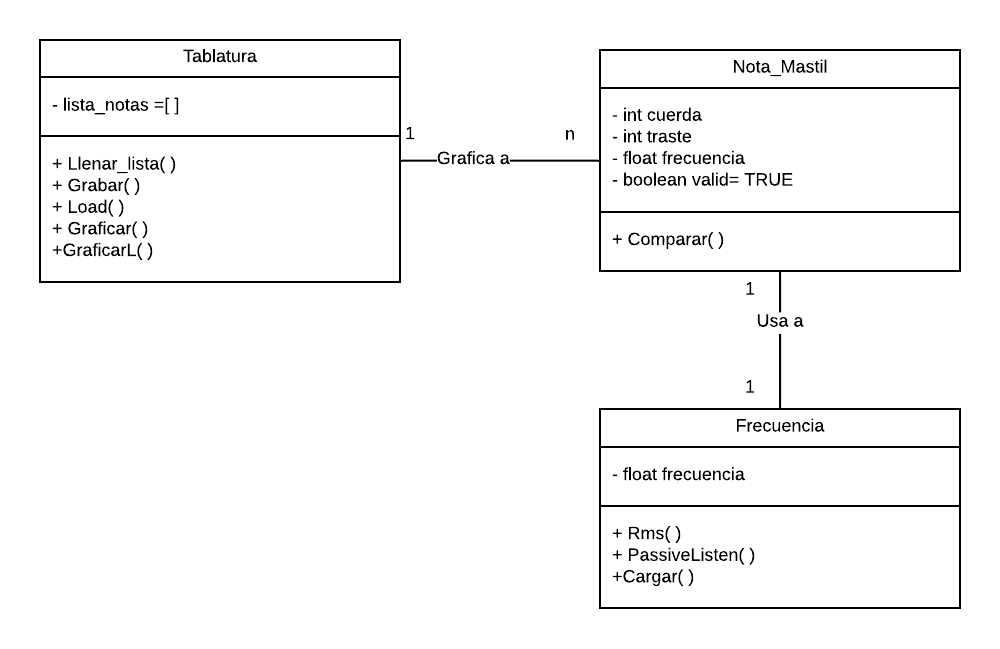




**Tabla 4.1:** Casos de uso.

4.2.2 Diagrama de clases

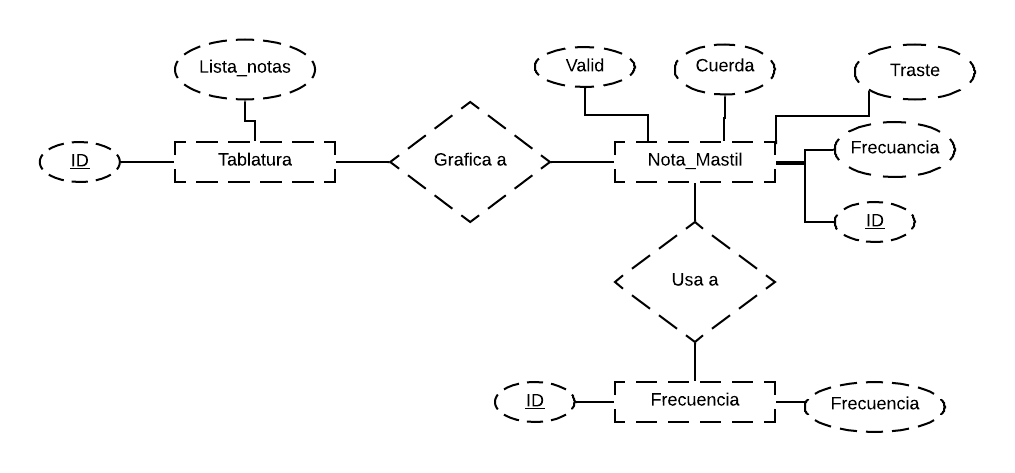
El diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema, mostrando las clases del sistema, sus atributos, operaciones o métodos, y las relaciones entre los objetos. De esta manera, a continuación, veremos el diagrama de clases del sistema.



**Grafica 4.3:** Diagrama de clases

4.2.3 Diagrama Entidad-Relación

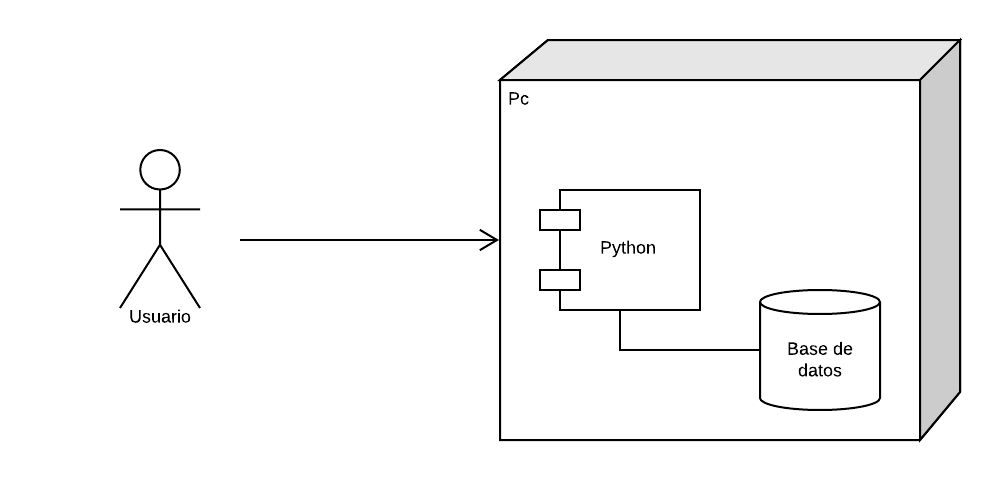
El diagrama entidad relación nos enseña cómo están relacionadas las entidades dentro del sistema, son usados para diseñar o depurar bases de datos relacionales si así se requiere. A continuación, veremos el diagrama Entidad-Relación del sistema.



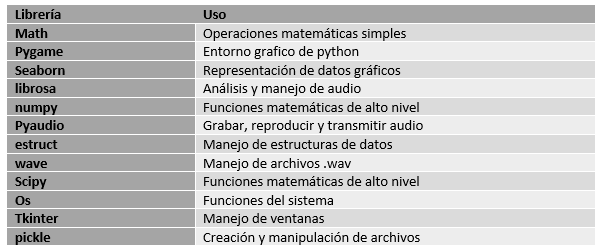
**Grafica 4.4:** Diagrama Entidad-Relación

4.2.4 Diagrama de despliegue

Este es quien modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan sobre estos nodos. A continuación, veremos el diagrama de despliegue del proyecto.



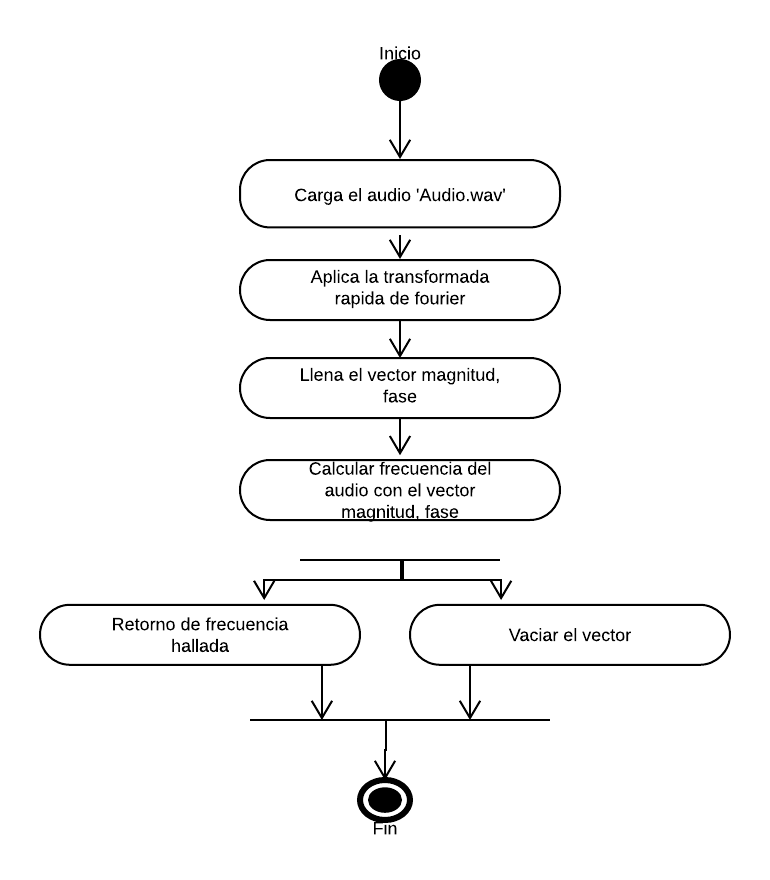
**Figura 4.5:** Diagrama de despliegue.



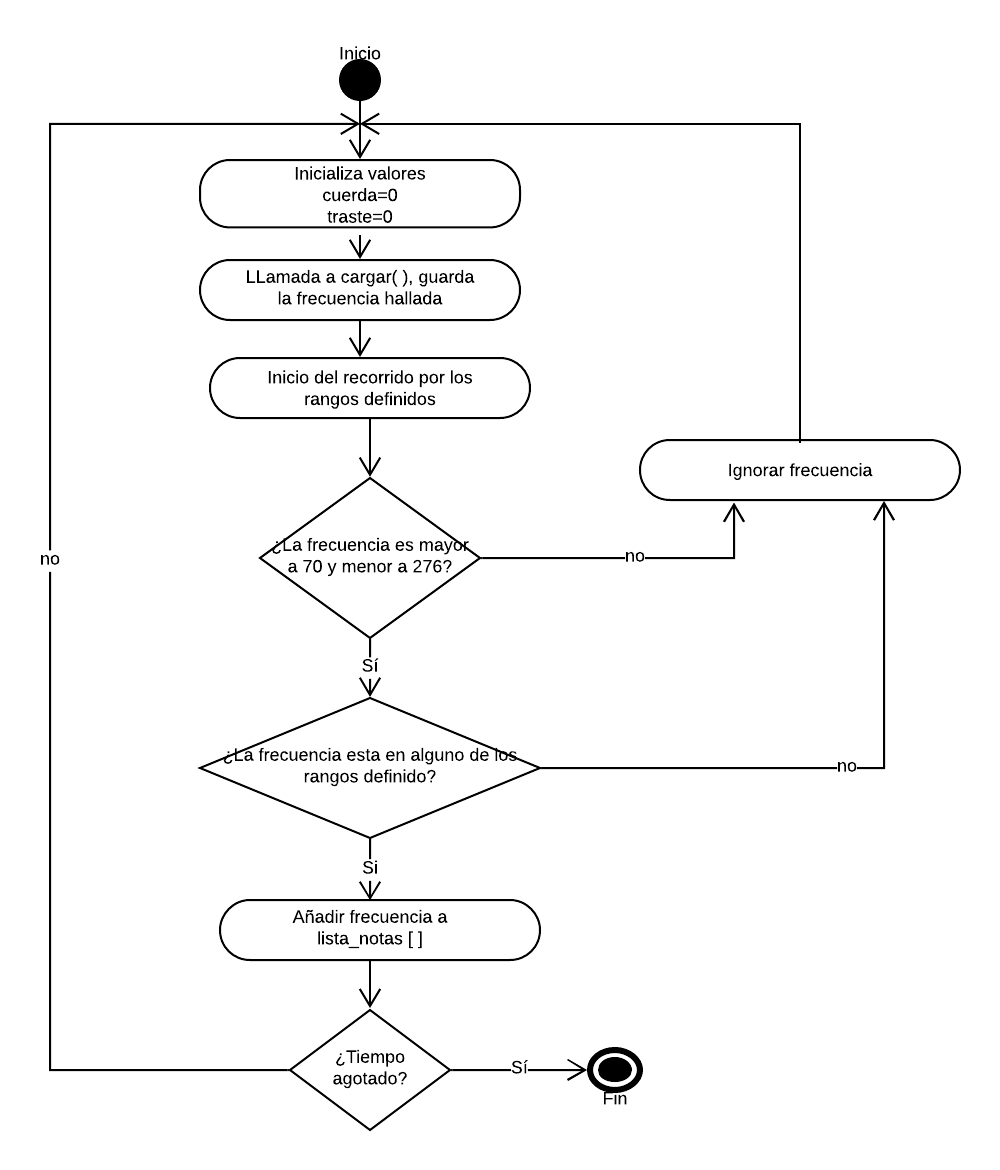
**Tabla 4.2:** Librerías necesarias dentro de python.

4.2.5 Diagramas de actividad

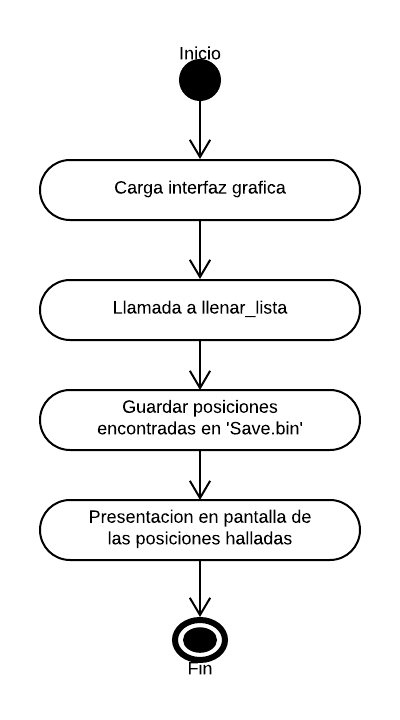
El diagrama de actividad es un diagrama de flujo de proceso multipropósito que se una para modelar el comportamiento del sistema, estos se pueden usar para modelar un caso de uso en particular, una clase, o un método complicado. Por ello y debido a que el sistema cuenta con diferentes casos críticos se han realizado los siguientes modelos.



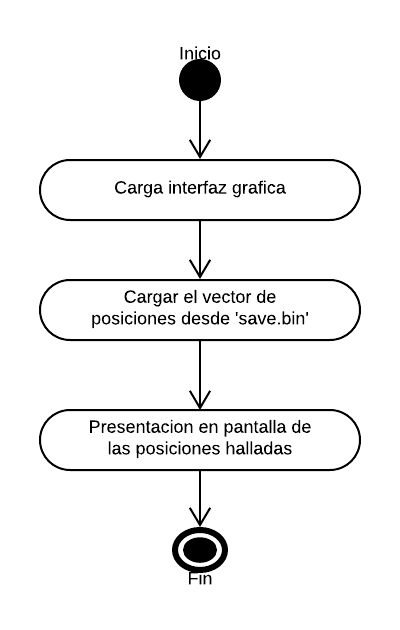
**Figura 4.6:** Diagrama de actividad, caso de uso PassiveListen



**Figura 4.7** Diagrama de actividad, caso de uso Comparar



**Figura 4.8:** Diagrama de actividad. Caso de uso Graficar



**Figura 4.9:** Diagrama de actividad, caso de uso GraficarL

* 1. Tecnologías utilizadas

Como se afirma en el numeral 4.2.4 Diagrama de despliegue la tecnología principal y bajo el cual se construyó la totalidad del proyecto es el lenguaje de programación Python. Ya que su manejo de audio y de las señales que lo acompañan es considerablemente más preciso que en otros lenguajes de programación conocidos por el desarrollador, ya que python cuenta con librerías especialmente diseñadas para el manejo de audio dando la posibilidad de grabar en nuestro ordenador un archivo con el audio que ha escuchado, además de conceder la posibilidad de modelar los objetos a nuestro gusto e inicializarlos de manera sencilla bajo los parámetros que establezcamos, pudiendo llenar después estructuras de datos como vectores con estos objetos, algo bastante útil para la implementación del sistema. Para este proyecto se ha utilizado específicamente la versión 2.7 de python, obtenido desde portal oficial de python.

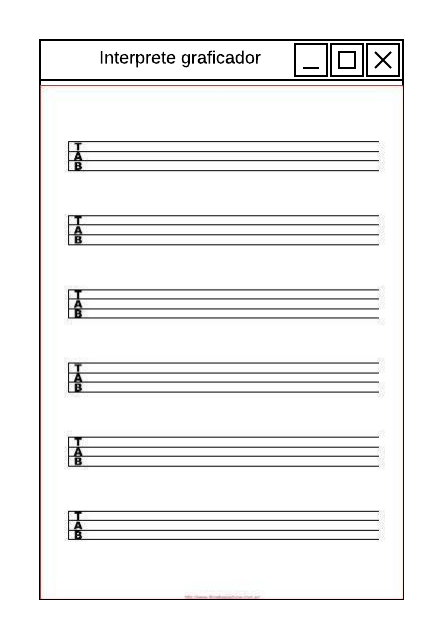
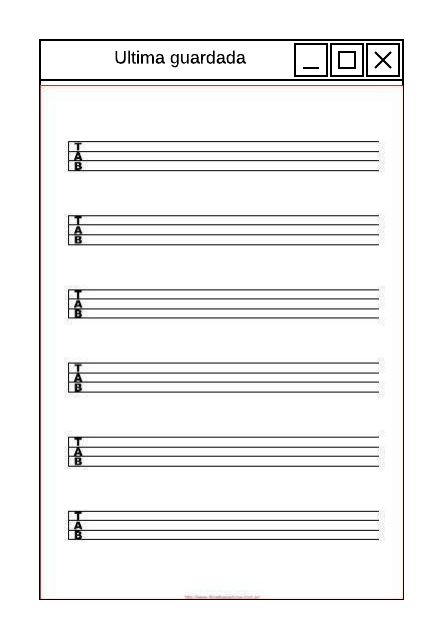
Además, para este desarrollo hemos tenido presente el uso de un bajo eléctrico marca Fander de 5 cuerdas, con alma estándar y pastillas completas como dedicadas. Aunque este ejemplar posea una cuerda demás para cumplir las especificaciones no tendrá inconvenientes para ajustarse a lo requerido, ya que la cuerda numero 5 (afinación Si) será ignorada. Como acompañante se ha utilizado un amplificador Boston de 115 watts y un parlante Bose de fácil transporte.

Para la realización de los diagramas UML los mockups se utilizó una herramienta gratis online que cuenta con todas las características necesarias para modelar cualquier sistema de manera sencilla e intuitiva, además de tener un estilo bastante agradable a la vista. Este es “lucidchar”.

* 1. Diseño interfaz de usuario

Para la implementación gráfica del sistema se realizaron primero algunos diseños en papel, para asegurarse de que la interfaz era amigable y era lo que el usuario necesitaba, después se plasmaron los ajustes necesarios en mockups. Estos mockups son una herramienta de diseño cuya intención es mostrar de manera sencilla y sin funcionalidades (aun) de como lucirá el sistema una vez finalizado su desarrollo. A continuación, veremos los mockups diseñados para el sistema.



**Figura 4.10:** Mockup inicio.

**Figura 4.11:** Mockups para el graficado

5

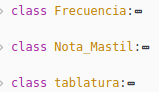
**Implementación**

En este capítulo se abordan las fases de implementación del proyecto. Se explicarán algunos detalles relevantes, así como las diferentes dificultades que se encontraron en el proceso

En esta fase se implementa el diseño realizado en el capítulo anterior de diseño, basado en el catálogo de requisitos del Capítulo 3 – Análisis, Como se ha especificado en el capítulo anterior se utilizará el lenguaje de programación python, junto a todas las librerías detalladas anteriormente, basando el diseño de la implementación en los diagramas UML del capítulo anterior y los mockups.

* 1. Desarrollo del proyecto

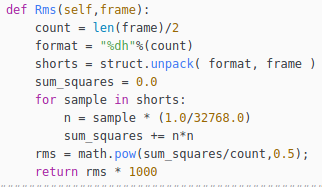
Al tener ya claro lo que se requería para el sistema, cuantas partes tendría, y como se comunicarían gracias al análisis y al diseño previo, la implementación del sistema no resulto ser tan complicada como se esperaba, primero definimos entonces las clases a usar.



**Figura 5.1:**  Clases

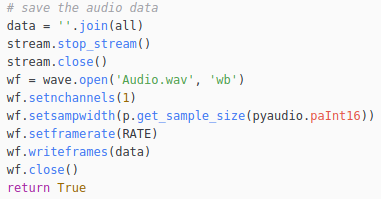
Se necesitaba empezar a implementar el apartado de la clase **Frecuencia** ya que esta quien escucha, graba, y descompone.

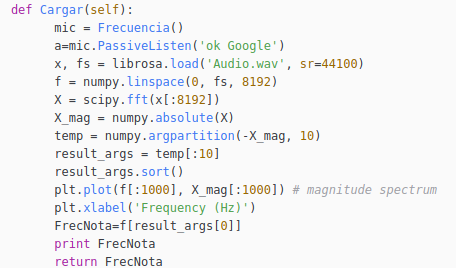
El diseño de esta clase nos permitirá empezar a conocer los rangos de frecuencia se mueven las notas generadas por el bajo eléctrico y se necesitaba esta información para continuar con los demás apartados, por eso fue la primera en implementarse, a continuación, veremos los métodos que la componen.



**Figura 5.2:**  Método Rms



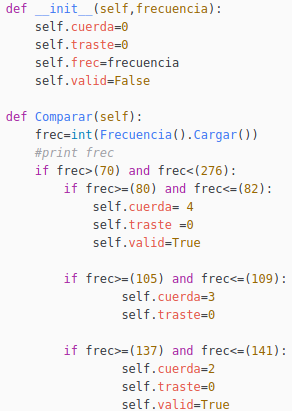




**Figura 5.3:**  Método Passivelisten y cargar

Como se apreció en la imagen anterior la implementación está acorde a lo que se definió en los diagramas de clases del capítulo 4- Diseño. Estos se encargan de escuchar guardar y aplicar la transformada de Fourier respectivamente, El diagrama de actividades del capítulo 4 nos enseña de una manera más amplia su funcionamiento.

Una vez implementada la clase que se encargara de escuchar y descomponer se hicieron múltiples pruebas con el bajo eléctrico, se anotaron las frecuencias que más resaltaban al generar una nota y se guardaron aparte, estas frecuencias fueron corroboradas en múltiples ocasiones para evitar usar datos erróneos, y a partir de estas se diseñaron los rangos bajo los cuales se moverán los resultados.



**Figura 5.4** Método Comparar y rangos de frecuencia

Al terminar este apartado, que hace parte de la clase **Nota\_Mastil** se implementó en totalidad la clase, al contar ya con los rangos de frecuencia solo fue necesario definir el método que iba a ser llamado por la clase **Tablatura**, este sería Comparar, quien utilizando el atributode la clase **Frecuencia** recorrería los rangos definidos, una vez encontrado el rango al que pertenece se crea un objeto **Nota\_Mastil** y se le asignan los atributos cuerda y traste dependiendo del rango que haya encontrado. Para evitar recorridos innecesarios se ha definido que el valor mínimo de entrada debe ser mayor a 70 Hz y máximo de 276 Hz, así se asegura que la frecuencia ingresada podría pertenecer a una nota generada por un bajo eléctrico. Además, el sistema se asegura que no haya datos vacíos, guardando dentro del vector lista\_notas [ ] únicamente los objetos que tienen asignado algún valor de posición.



**Figura 5.5:** Método Graficar

Ya teniendo la parte gruesa del sistema solo nos resta realizar la parte gráfica y el llamado a las demás funciones que hacen parte de la clase **Tablatura**, teniendo los mockups no se tuvieron muchos inconvenientes. Para la venta principal se utilizó la librería Tkinter de python, ya que el manejo de las ventanas es más sencillo con ella, diseñamos la ventana principal y el primer llamado a graficar. Para esta parte se decidió que creara una ventana con la librería Pygame, dado que se graficara sobre una imagen las posiciones almacenadas en el vector lista\_notas [ ] y es más sencillo con esta librería, a su vez el vector posición es guardado en un archivo ‘save.bin’ en nuestra máquina, permitiendo graficar nuevamente estas posiciones más adelante.

Para ver con más detalle el funcionamiento de estos métodos críticos en el capítulo 4 – Diseño se muestran los diagramas de diseño del sistema.



**Figura 5.6:** Método Grabar y Load

Una vez implementado todo por aparte, solo resta juntar y hacer los llamados a las funciones definidas. Lo primero que se le presentará al usuario será la ventana principal de Tkinter, en ella habrán 2 opciones de graficado, la primera genera el vector lista\_notas [ ] vacío y llama a su función comparar para llenarlo, esta a su vez hereda el valor de frecuencia de la clase **Frecuencia** para hacer las debidas comparaciones, entrando en un bucle de escucha, descomposición, comparación, guardado hasta que el tiempo se termine, una vez que esto suceda el método Graficar mostrara los valores almacenados dentro del vector lista\_notas [ ], y creando el archivo ´save.bin´. La opción Re visualizar, carga directamente el vector lista\_notas [ ] del archivo ‘save.bin’ y re grafica las notas de la misma manera que lo hace la opción 1.



**Figura 5.7:** Método Llenar\_lista



**Figura 5.8:** Método GraficarL

* 1. Conclusiones del desarrollo

Este apartado se redacta a modo de resumen final del capítulo con el fin de condensar los aspectos más importantes y algunas de las dificultades encontradas.

Se han desarrollado con éxito los módulos previstos en la fase de diseño, las interfaces y las tareas de cada clase que completan la funcionalidad del software

Dado el conocimiento adquirido durante los años en la carrera, tanto es asignaturas directamente ligadas al lenguaje como las que se centran más en desarrollo de software, no han surgido problemas que no hayan podido solucionar haciendo uso de internet y de documentación de las librerías usadas.

El problema más remarcable que se encontró en la implementación radicaba en cómo mostrar la tablatura generada, ya que al comienzo solo se planeaba usar Tkinter el cual no contiene un entorno grafico muy poderoso, limitando mucho lo que se podía hacer por lo que se estaban complicando las cosas sin necesidad, al final se optó por usar Pygame junto a Tkinter para que la interfaz fuera como se deseaba, dejando la ventana principal en Tkinter con su capacidad para manejar demás ventanas y el apartado gráfico solo a Pygame, usando este conjunto se logró lo que

se buscaba, una comunicación entre ambas que posibilitara el cumplimiento de los objetivos del proyecto.

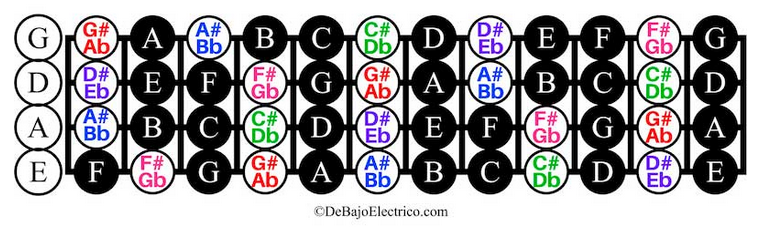
Algunos problemas recayeron en la llenada del vector lista\_notas [ ] ya que en diseño original cuando no encontraba un rango de frecuencias al cual perteneciera la nota ingresada (ruido), agregaba un objeto con valores =0, ocasionando que en la gráfica se visualizaran 0 que no deberían existir. Esto se solucionó agregando una variable boolean para que el sistema asegurara que si había entrado a un rango antes de añadirla al vector.

6

**Pruebas**

Para la realización de las pruebas se utilizó el bajo eléctrico descrito en el apartado 4.3 del capítulo 4- Diseño, las pruebas realizadas se basaron principalmente en tocar frente al software y ver qué datos arrojaba.

Se evidencio que la forma de tocarlo afecta a algunas de las frecuencias que el software puede reconocer dando posiciones de notas erradas.

Se evidencio que el software tiende a confundir posiciones, debido a que tienen una frecuencia similar, investigando más sobre esto se encontró que las notas del bajo sigue cierto patrón en su estructura

**Figura 6.1** Notas de un bajo eléctrico.

Debido a que después del traste 12 esta estructura se repite, el software retorna la primera que encuentra mientras realiza el recorrido de los rangos de frecuencia, esto genera un inconveniente para el objetivo del software, pero será tenido en cuenta en versiones futuras.

Sin embargo, al delimitar el software a trabajar sobre los 12 primeros trastes no se encontraron casi incoherencias en sus resultados, se decidido también darles prioridad a las cuerdas 4 (Mi) y 3 (La) ya que son las más utilizadas por los bajistas.

El sistema cuenta con una precisión del 65% de notas ingresadas, debido a que es muy sensible y los demás ruidos del entorno pueden alterar el resultado de la frecuencia que será buscada, en un entorno ideal el porcentaje sube considerablemente.

7

**Conclusiones**

Este capítulo se dedica a conclusiones finales y personales del proyecto, extraídas a lo largo de todo el proceso de desarrollo del mismo.

Se han cumplido todos los objetivos y requisitos mencionados en los capítulos anteriores. El principal de ellos era desarrollar un software que permitiera la graficación de posiciones de las notas ingresada por el usuario en tiempo real en una tablatura simplificada.

El software desarrollado ofrece la comodidad de diseñar la tablatura por el usuario, además de soportar memoria para re visualizar la última tablatura que fue generada.

Atendiendo el desarrollo fue difícil sintetizar todas las ideas que había en un principio, pero una vez realizada la fase de análisis y diseño, las fases posteriores fueron abordadas rápidamente, por lo que las primeras fases de planeación resultaron ser de suma importancia.

Además de haber aprendido a desarrollar un proyecto de principio a fin, pasando por todas sus fases, he adquirido nuevos conocimientos, o ampliado muchos de ellos, eh aprendido a utilizar correctamente muchas librerías de las que dispone python, además de ampliar mi conocimiento sobre manejo de ventanas en el mismo, y otros conocimientos necesarios que se han requerido durante todas las fases, resaltando sobre todo el modelamiento UML donde tuve que trabajar un poco más ya que era muy abstracto para mí.

8

**Trabajo futuro**

Dentro del trabajo futuro de este proyecto se pueden destacar varias mejoras.

La primera de ellas es mejorar la interfaz gráfica, con ayuda de una persona especialista se puede lograr una aplicación con un acabado más profesional.

Puede ampliarse el trabajo realizado, haciendo ajustes para que la cantidad de bajos que pueden ser partícipes del software sea cada vez mayor, incluyendo bajos de todos los tipos de cuerdas existentes, ampliando el número de cuerdas, pudiendo reconocer las notas a pesar diferentes técnicas que se utilicen.

Se puede ampliar la gama de instrumentos que pueden hacer uso de este sistema, incluyendo otros instrumentos de cuerdas como guitarra, contrabajo, ukulele etc.

Con un poco más de trabajo sobre las señales se podrían introducir piezas musicales completas y que el sistema sea capaz de graficar la tablatura del instrumento que se le solicite, ignorando los demás.

La velocidad del reconocimiento de notas puede ser aumentada, pudiendo ingresar más notas por segundo.

La precisión puede ser aumentada considerablemente si se añaden más valores a tener en cuenta a la hora de calcular la frecuencia.

**Bibliografía**

[1] https://docs.python.org/3/

[2] www.homohominisacrares.net/php/articulos.php?num\_revista=3&cod\_articulo=34

[3] https://www.redalyc.org/html/849/84925149018/

[4] https://es.wikipedia.org/wiki/Bajo\_el%C3%A9ctrico

[5] https://es.wikipedia.org/wiki/Sonido

[6] https://latam.googleblog.com/2011/11/una-mirada-dentro-de-la-tecnologia-de\_23.html

[7]https://repositorio.uam.es/bitstream/handle/10486/662281/gomez\_matesanz\_alfonso\_tfg.pdf?sequence=1

[8] https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

[9]https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de- desarrollo/4538221-en-que-consiste-el-modelo-en-cascada

[10] https://es.wikipedia.org/wiki/Timbre\_(ac%C3%BAstica)

[11]https://www.lucidchart.com/document

**Anexos**

**Manual de usuario**

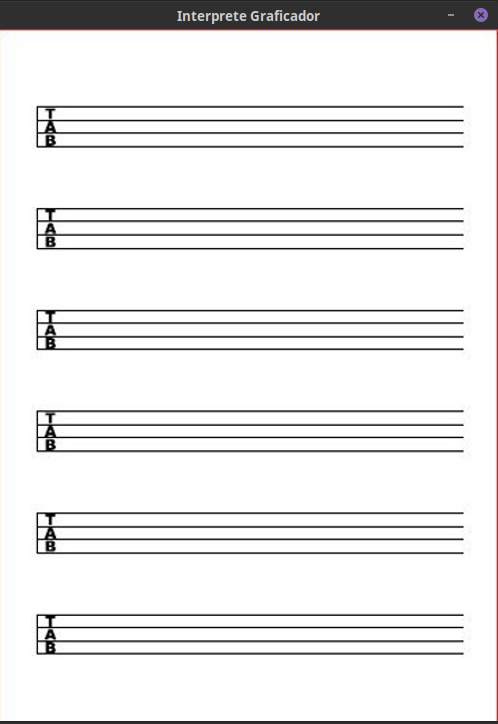
Al arrancar el software se mostrará una pantalla de inicio.



**Figura A1:** Pantalla de inicio

El software esperara en esta ventana hasta que el usuario presione alguno de los botones

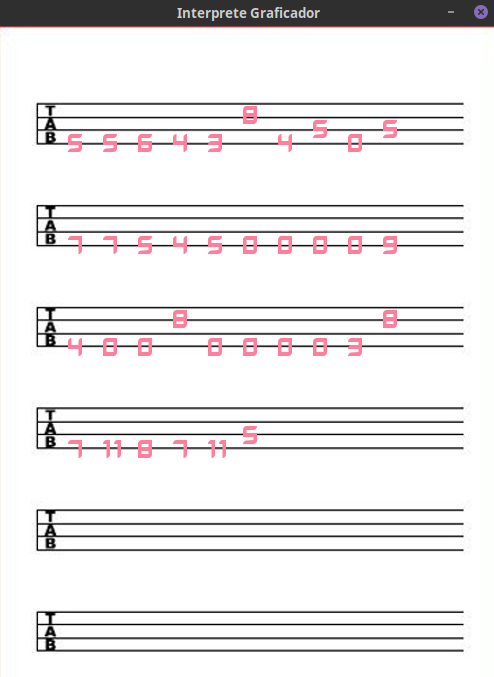
Botón “Graficar”



**Figura A2:** Ventana al presionar “graficar”

Manual de usuario

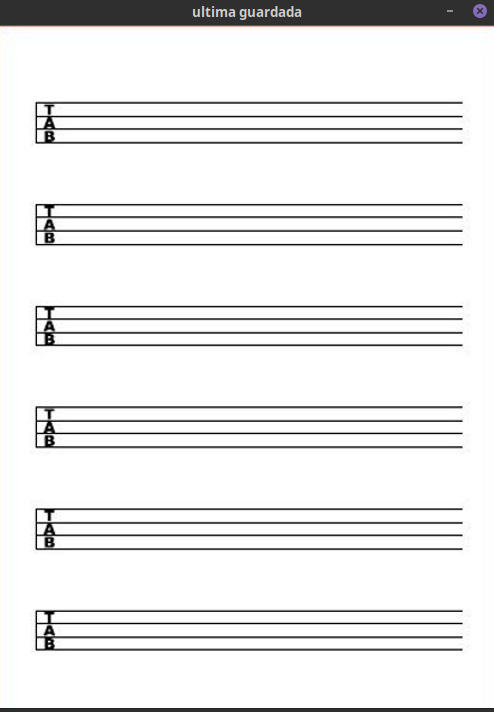
El software empezara a escuchar una vez el botón sea pulsado y mostrara la ventana en blanco hasta que el tiempo de escucha.



**Figura A3:** Resultado “graficar”

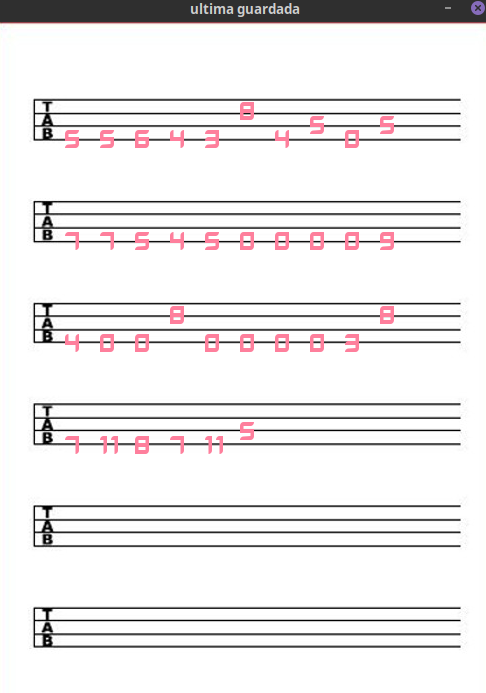
Una vez el tiempo se agote se mostrará en pantalla el resultado de las notas que el usuario ingreso.

Botón “Re visualizar”



**Figura A4:** Re visualizar vacía

Si el software no ha sido ejecutado anteriormente el software mostrara una tablatura vacía, ya que el archivo ‘save.bin’ no existe en ese momento.



**Figura A5:** Re visualizar

Cuando el programa sea ejecutado y se guarden las posiciones, el software mostrara con “Re visualizar” estas últimas posiciones guardadas en ‘save.bin’