UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

SOFTWARE MUSICAL PARA EL APOYO A LA MATERIA "ARMONÍA"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA:
ULISES PÉREZ ARCEO

ASESOR:
MTRO. CARLOS MONTES DE OCA GUERRERO

Julio de 2016

Santa Cruz Acatlán, Naucalpan, Estado de México

AGRADECIMIENTOS

a mi Co-asesor Lic. Oscar Gabriel Caballero Martínez por su gran aporte para que pudiera terminar el sistema *Armonia M@Puma.*

Omar Guerrero por brindarme un espacio en videojuegos de la Fes Acatlán que me sirvió para desarrollar la tesis.

A la máxima casa de estudio la UNAM, llevare un Goya en mi corazón por siempre.

Índice

Software musical para el apoyo a la materia “Armonía”

Introducción

La producción actual de software como herramienta de ayuda para el desarrollo de todos los aspectos del conocimiento humano abarca prácticamente todos los campos. El entretenimiento, la educación, los idiomas, los negocios, la mercadotecnia, la arquitectura y la ingeniería, las ciencias sociales y naturales, las comunicaciones y las artes se han visto beneficiadas por el mundo de la computación. Pese a esto, algunos campos específicos del conocimiento no han tenido la suerte de contar con software que ayude a los interesados a facilitar o automatizar su trabajo. Una de estas áreas específicas es la armonía musical.

Desde hace veintiocho años el mundo de la música cuenta con diversos paquetes de software. Los más representativos son *Finale* (1988) y *Sibelius* (1993), creados para la escritura musical; *Ear Master* (1996), diseñado para el entrenamiento auditivo; Transcribe!, para la transcripción de música grabada. Sin embargo hasta la fecha la armonía musical no contaba con un software especializado. Se percibe lo complicada que puede ser para algunos alumnos. De esto último surgió la idea de elaborar un software educativo que genere corrección de ejercicios de armonía musical, y que por otra parte ayude a estimular, corregir y apoyar el aprendizaje en dicha materia en las escuelas de música de nivel medio superior y superior. Así la hipótesis planteada es: la aplicación de software *Armonía M@Puma* es una herramienta que detecta errores en ejercicios de armonía musical. Por lo tanto el objetivo de este trabajo es desarrollar una aplicación de software que verifique que los ejercicios realizados en la materia de armonía cumplan con las reglas de armonía musical, y demostrar la hipótesis.

Los beneficios del software desde un punto de vista práctico serán:

Contendrá ítems, para poder seleccionar diversas notas musicales: cuartos, mitades y unidades. Otros ítems están diseñados para escribir notas auxiliares y algunos signos musicales. Una diversidad de botones marcados para las primeras 21 reglas de armonía musical según la recopilación de algunos fragmentos de piezas musicales, usados como ejercicios de armonía musical, utilizados por Isabel Jeremías Lafuente y Enrique Cordero R. de diversos compositores, con las cuales se pretende que el alumno ponga al día sus conocimientos en la materia.

Por el momento el software es susceptible de mejoras. Es decir, se presenta aquí en una versión Alpha, para dejar el desarrollo de una versión Beta para un trabajo futuro.

Java es un lenguaje de programación de entorno de desarrollo libre, siendo un producto gratuito sin restricciones de uso, lo que lo convierte en una herramienta poderosa y necesaria en estos tiempos para el aprendizaje de la programación en diversas universidades.

El desarrollo de software flexible y gratuito proporciona pautas de utilización efectivas. Para apoyar la materia de armonía musical se pretende implementar el sistema *Armonía M@Puma* que se instituirá en las escuelas de música de arte ó interesadas que impartan armonía musical.

Esta tesis está dividida en cuatro capítulos; El primero familiariza al lector sobre que es la armonía musical, explicando su significado y uso en la música. Se incluye una breve historia de su concepción y desarrollo, para poder implementarlo en el software.

El segundo capítulo, explica detalladamente las reglas generales de la armonía musical para su uso y comprensión adecuada.

El capitulo tres, define un mapa conceptual que seguirá el proceso de construcción del software para la autoevaluación del estudio de la armonía musical.

Y finalmente, el último capítulo muestra un estudio de campo del software para poder definir los resultados de su utilización en la práctica.

El software resultante será sometido a evaluación por parte de los alumnos de armonía musical de las de escuelas de Bellas Artes de Naucalpan, del Conservatorio Nacional de Música y de la escuela de Mariachis Ollin Yoliztli de Garibaldi, a través de una encuesta cuyos resultados serán presentados en el capítulo 4.

La construcción de la aplicación fortalecerá los conceptos de armonía musical.

El software estará dirigida a jóvenes de 17 a 23 años. Los grupos están formados entre 10 y 20 alumnos en las escuelas de Bellas Artes de Naucalpan y de Mariachis Ollin Yoliztli de Garibaldi.

Capítulo 1

Introducción al software educativo

Desde la introducción de las computadoras en el campo de la educación, han sido numerosas las tentativas de elaborar marcos de referencia que permiten disponer, a los diseñadores y a los usuarios de software educativo, de formas sistemáticas de reflexionar sobre sus tareas. Estas tentativas no se han debido solo al deseo de ayudar al desarrollo del software sino también de contribuir a la descripción, discusión, comparación, selección y evaluación del software educativo y a comprender la función que la informática podría desempeñar en contextos educativos más amplios.

Marcos de referencia como los conceptualizados por Pérez Marqués o Begoña Gros Se han centrado en las funciones que se pretende desempeñe el software. Más han intentado relacionar el software con fundamentos educativos. Que gozan de aceptación general un enfoque muy simple, aunque muy utilizado, distingue dos tipos de software educativo: el carente de contenidos y el específico para cada asignatura. El primero (suele denominarse también “genéricos”). Se analiza de acuerdo con las tareas que pueden desarrollar; por ejemplo, los procesadores de texto pueden usarse para, manipular textos; los paquetes de gestión de información se utilizan para buscar datos, las hojas de cálculo para efectuar cálculos extensos y complicados.[1]

El software carente de contenido, no se diseña específicamente para un área o tema del plan curricular aunque los profesores de diferentes asignaturas lo pueden utilizar con distintos fines, así, el mismo paquete de gestión de información puede utilizarlo un profesor de historia para explorar los registros del censo local. En cambio, el software específico de una asignatura es todo aquel diseñado para ser utilizado en la enseñanza, y el aprendizaje de temas concretos en especial de áreas curriculares o asignaturas.

1.1 Diseño de software educativo

El medio didáctico es definido como un elemento curricular que, por su sistema simbólico y estrategia de utilización propicia el desarrollo de habilidades cognitivas en los estudiantes en un contexto determinado se facilita la intervención mediada sobre la realidad, el empleo de determinadas estrategias de aprendizaje y la captación y comprensión de la información.

A estas características se añaden, en el caso del software educativo, la capacidad de influir en los objetivos, los contenidos y estrategias de enseñanza.

[1] Dr. Fernández Aedo C., Raúl. y Lic. Delavaut Romero E., (2013: 90). Martín. Educación y tecnología un binomio excepcional. Madrid: Grupo Editor k.

Hay que destacar dos cualidades básicas en todo programa informático con finalidad educativa: debe permitir al profesor su adaptación a múltiples situaciones que faciliten ciertos aprendizajes a los alumnos y además, debe ser coherente con las teorías pedagógicas actuales y los requerimientos del sistema educativo, contenidos y procedimientos de uso.[2]

1.2 Conceptualización del software educativo

Se puede definir el término de software educativo como "programas de computadora para la educación". Hay muchas definiciones entre las que destacan la de:

- *Pérez Marqués (1996).* "Son los programas de computadoras creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje".
- *Begoña Gros (1997).* "Cualquier producto realizado con una finalidad educativa".
- *Ceja Mena (2000).* "Son aquellos programas creados con la finalidad específica de ser utilizados como medio didáctico; es decir, para facilitar los procesos de enseñanza y aprendizaje, tanto en su modalidad tradicional presencial, como en la flexible y a distancia".[3]

Estas definiciones engloban todos los programas que han sido elaborados con fines didácticos, desde los tradicionales programas basados en los modelos conductistas de la enseñanza, los programas de enseñanza asistida por computadora hasta los programas de enseñanza inteligente asistida por computadora, que utilizando técnicas propias del campo de los sistemas expertos y de la inteligencia artificial, pretendiendo imitar la labor tutorial personalizada que realizan los profesores y presentan modelos de presentación el conocimiento en consonancia con los procesos cognitivos que desarrollan los alumnos.[4]

El software educativo puede tratar las diferentes materias de formas muy diversas y ofrecer un entorno de trabajo más o menos sensible a las circunstancias de los alumnos y más o menos rico en posibilidades de interacción; pero todos comparten cinco características esenciales según *Ríos y Ruíz (1998):*

---

[2] Dr. Fernández Aedo C., Raúl. y Lic. Delavaut Romero E., Martín. (2013: 91). Educación y tecnología un binomio excepcional. Madrid: Grupo Editor k.
[3] Dr. Fernández Aedo C., Raúl. y Lic. Delavaut Romero E., Martín. (2013: 105). Educación y tecnología un binomio excepcional. Madrid: Grupo Editor k.
[4] Dr. Fernández Aedo C., Raúl. y Lic. Delavaut Romero E., Martín. (2013: 105). Educación y tecnología un binomio excepcional. Madrid: Grupo Editor k.

- Son materiales elaborados con una finalidad didáctica.
- Utilizan la computadora como soporte en el que los alumnos realizan las actividades que ellos proponen.
- Son interactivos, contestan inmediatamente las acciones de los estudiantes y permiten un dialogo y un intercambio de informaciones entre la computadora y los estudiantes.
- Individualizan el trabajo de los estudiantes, ya que se adaptan al ritmo de trabajo de cada uno y pueden adaptar sus actividades según las actuaciones de los alumnos.
- Son fáciles de usar. Los conocimientos informáticos necesarios para utilizar la mayoría de estos programas son mínimos aunque cada programa tiene unas reglas de funcionamiento que es necesario conocer.

1.3 Armonía Musical

La armonía es un arte que trata de conjugar dos factores que muchas veces se encuentran en situación conflictiva o contradictoria: el factor horizontal melódico y el factor vertical armónico. Cuando alrededor del año 1000 de nuestra era los compositores adquirieron conciencia de la posibilidad de ejecutar varios sonidos simultáneamente y de tocar diferentes melodías al mismo tiempo, la música adquirió dos dimensiones, pues durante todos los siglos anteriores la música fue predominante monódica. Para entender esos conceptos mezclados en la música que dan origen a la armonía musical se explica a continuación algunas definiciones.

1.3.1 Música, melodía, ritmo y armonía musical.

Los principales elementos de la música son: Melodía, armonía musical y ritmo.

Melodía: es la sucesión de sonidos de diferente altura que animados por el ritmo, expresan una idea musical.

Armonía musical: es la parte de la música que estudia la formación y combinación de los acordes.

Ritmo: es el orden y proporción en que se agrupan los sonidos en el tiempo.[5]

La armonía funciona como acompañamiento de las melodías o como una base sobre la que se desarrollan varias melodías simultáneas. Con esto, podemos decir que melodía y armonía son términos muy relacionados entre sí, con lo cual se puede considerar la melodía como un conjunto de sonidos armónicos que se suceden en el tiempo y están en relación con los acordes en los que se basa esa melodía.

[5] Moncada García, Francisco. (1997: 19). La más sencilla, útil y práctica teoría de la música. México D.F. Framong.

Capítulo 2

Reglas de armonía musical

A continuación se describen las reglas de armonía del sistema *Armonía M @Puma.*

Las reglas de la armonía musical[6] se utilizan para mantener en equilibrio los acordes, en cada bloque correspondiente como se menciona a continuación.

2.1 Reglas de armonía musical para acordes de quinta

Al escribir una triada para un conjunto de cuatro voces nuestro primer objetivo será conseguir una sonoridad equilibrada. Las reglas siguientes tienen este propósito.

Regla 1: Teniendo cuatro voces y sólo tres notas será necesario duplicar una de ellas, se duplicará preferentemente la fundamental y en segundo término la 5ª del acorde. Sin duplicar la 3ª.

Regla 2: Se podrá disponer las notas del acorde en cualquier orden en las voces superiores pero el bajo cantará la fundamental del acorde.

Regla 3: No se debe escribir ninguna nota fuera de la tesitura7 de la voz correspondiente.

Regla 4: Se prescinde de evitar el "cruzamiento" de las voces, (por ejemplo que los tenores canten por encima de los contraltos).

Regla 5: Entre el bajo y el tenor puede haber el intervalo que se quiera pero entre dos de las voces superiores no puede haber un intervalo mayor a una octava.

Tener un acorde desequilibrado en su sonoridad tendría como consecuencia una pobreza estética o una sonoridad arcaica. Por lo tanto la distancia de las cuatro voces, debe ser la correcta. El software se encargará de mandar mensajes de error al existir algún rompimiento de alguna de las reglas establecidas. El software hará un chequeo de cada voz en cada rango en el tiempo determinado de los compases musicales.

2.2 Reglas de armonía musical para el enlace de acordes de quinta

El segundo objetivo de este trabajo es que el alumno de armonía musical pueda revisar si sus enlaces de acordes son correctos. Además de la sensación de continuidad. Para esto, existen en armonía musical tres tipos de movimientos de las voces según su interacción:

---

[6] Jeremías Lafuente, Isabel y Cordero R., Enrique. (2015). "Estas son reglas de armonía musical recopilada de algunos fragmentos de piezas musicales usada como ejercicios de armonía musical usados por de diversos compositores".

[7] "La tesitura en música, es un término usado para describir la parte vocal de algunos compas musicales la cual se encuentra en una pieza musical". Fuente propia.

Movimiento directo: las voces marchan en la misma dirección.
Movimiento contrario: las voces van en dirección opuesta.

Movimiento oblicuo: una o más voces permanecen inmóviles, la otra u otras van en alguna dirección.

Para alcanzar este objetivo se deben utilizar las siguientes reglas

Regla 6: Se aplicarán solo acordes en posición fundamental y con la fundamental duplicada.

Regla 7: Si el bajo hace un movimiento de 3ª (o 6ª) deben de mantenerse tenidas las dos notas comunes a los dos acordes y llevar la nota restante a la nota más próxima del acorde siguiente.

Regla 8: Si el bajo hace un movimiento de 4ª (o 5ª) se deja tenida la nota común a ambas y las otra voces se mueven a las notas más próximas del acorde siguiente.

Regla 9: Si el bajo hace un movimiento de 2ª las tres voces superiores hacen un movimiento contrario al del bajo que debe partir a las notas más cercanas del acorde siguiente.

2.3 Reglas de armonía musical para acordes de sexta (primera inversión)

El tercer objetivo será cuando el usuario desee armonizar los acordes de sexta; cuando el bajo canta la 3ª del acorde se forma un acorde de sexta o también conocido como, acorde en primera inversión.

Para alcanzar este objetivo utilizaremos las siguientes reglas:

Regla 10: Duplicar la fundamental o la 5ª según resulte más conveniente en cada caso.

Regla 11: No se debe llegar a un unísono o salir de el por movimiento directo de ambas voces.

Regla 12: Con el objetivo de producir una sensación polifónica completa se prohíben los unísonos, 8avas o 5as consecutivas en las mismas voces. Se permiten solo si permanecen tenidas. La regla se aplica tanto por movimiento directo como contrario.

Regla 13: Por causa semejante están prohibidas las 8vas y 5as por movimiento directo de las voces exteriores. Se permiten solo si una de las voces hace movimiento de

semitono. Se permiten también entre las voces internas o entre una exterior y otra interior.

Regla 14: Las voces deben de marchar a la nota más cercana del acorde siguiente, a menos que sea necesario evitar una falta.

Las reglas de enlace de los acordes de 5ª no se aplican cuando interviene un acorde 6ª.

2.4 Reglas de armonía musical para acordes en modo menor

El cuarto objetivo es enlazar los acordes en modo menor, este modo se construye a partir del sexto grado de la escala mayor. El modo menor no tiene sensible y por la influencia sonora fue modificándose hasta que terminó por adaptar una sensible.

La alteración al sexto grado no se aplica, sin embargo, si la nota que sigue a este grado no es la sensible ni tampoco si el movimiento desciende porque en tal caso no hay la necesidad de escuchar la sensible y el séptimo grado puede emplearse sin alteración alguna.

Para poder hacer frente a estas dificultades armónicas se usan las siguientes reglas:

Regla 15: Se podrá alterar el 7º grado de la escala para formar *la sensible* pero solamente en los acordes V y VII.

Regla 16: Se alterará el 6º grado de la escala cuando sea necesario evitar el intervalo melódico de segunda aumentada que este grado puede formar con la sensible.

2.5 Reglas de armonía musical para acordes de cuarta y sexta

El quinto objetivo es enlazar los acordes de cuarta sexta, estos se forman cuando el bajo canta la quinta del acorde se dice que la triada se encuentra en posición de cuarta y sexta o en segunda inversión. Se la llama de cuarta y sexta porque estos son precisamente los intervalos que forman la fundamental y la quinta en relación con el bajo.

Las reglas de empleo de los acordes de cuarta y sexta son las siguientes:

Regla 17: la armonía musical consideraba a los acordes de cuarta y sexta como inestables por lo que deben de emplear en tiempo débil. En esta regla se buscará que los acordes de cuarta y de sexta pertenezcan al tiempo débil ya mencionado.

Regla 18: para llegar y salir de un acorde en segunda inversión es necesario que el bajo haga un movimiento conjunto o se mantenga inmóvil.

Todas las otras reglas de disposición y enlace se mantienen vigentes.

2.6 Reglas de armonía musical para la resolución de acordes de séptima en dominante

El sexto objetivo es la resolución del acorde de séptima de dominante[8], estos acordes son inestables y requieren resolución, esto es, su enlace a otro acorde que resuelva la disonancia experimentada en el acorde de séptima de dominante a la consonancia de un acorde mayor o menor.

En su posición fundamental el acorde de séptima dominante puede tener tres soluciones según las siguientes reglas:

Caso 1.- Cuando el acorde de séptima dominante completo resuelve a I[9] sin quinta

Regla 19:

La fundamental va a la fundamental.

- La sensible va a la tónica.
- La 7ª desciende por grado.
- La 5ª desciende por grado. El acorde de I queda sin quinta y con la fundamental triplicada.

Caso 2.- Cuando el acorde de séptima dominante sin 5ª resuelve a I completo

Regla 20.

- La fundamental va a la fundamental.
- La sensible va a la tónica.
- La 7ª desciende un grado.
- La duplicación de la fundamental queda "tenida[10]".

Caso 3.-Cuando el acorde de séptima de dominante resuelve a I

[8] "Los acordes de séptima dominante se construyen superponiendo una tercera a la triada del quinto grado". Fuente propia

[9]""I" es conocida como el primer grado de un acorde, esta notación se usa en armonía musical, no confundir con numero 1, son diferentes se hace uso de la i latina mayúscula". Fuente propia

[10]"Una nota tenida es aquella que se mantiene extendida hasta los siguientes compases musicales, unida por una ligadura de prolongación". Fuente propia

Regla 21: Solo puede ocurrir cuando en las voces intermedias una voz canta la tónica a la que debió resolver la sensible:

- Fundamental a fundamental.
- Sensible a la 5ª de I.
- 7ª y 5ª se descienden un grado.

Capítulo 3

Construcción del software

Como es sabida la música se escribe en un rayado especial llamado pentagrama, tal como se muestra en la figura1:

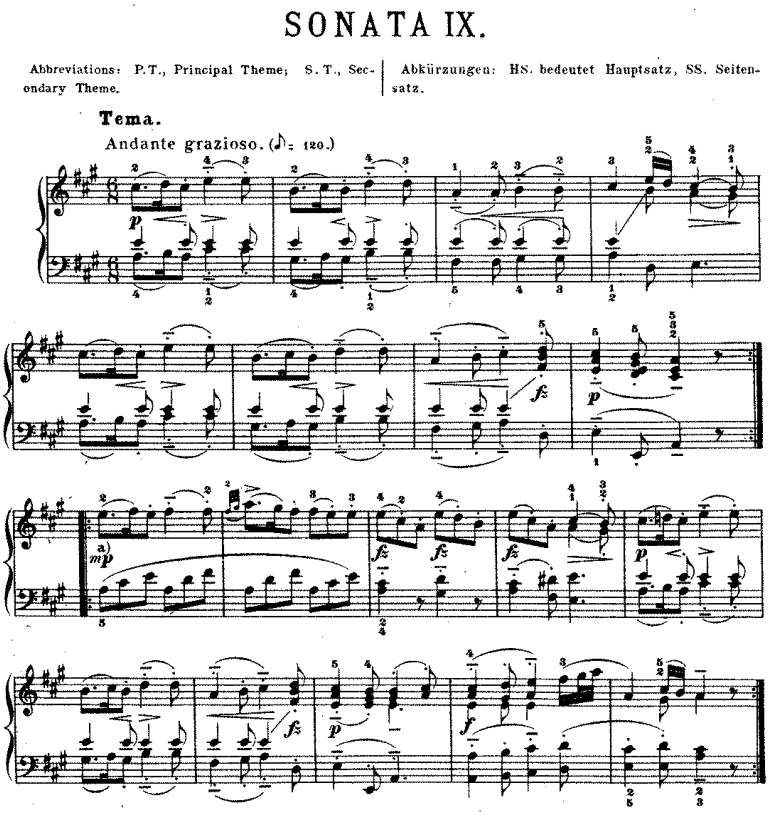

Figura 1. Sonata para piano n° 11 Kv 331 en La mayor, cc. 1-18

Se usan signos musicales, llamados notas musicales.

Estas notas se escriben sobre las líneas y espacios del pentagrama.

Las notas que por su altura ya no pueden escribirse dentro del pentagrama se escriben sobre pequeñas líneas escritas arriba o debajo del pentagrama y a la misma distancia que las líneas del pentagrama llamadas líneas adicionales.

Ahora explicaremos como se creó el software, que pretende analizar ejercicios básicos de armonía musical. Según la recopilación de Isabel Jeremías Lafuente y Enrique Cordero R.

3.1 Construcción del *frame*

Para empezar en el software se pueden visualizar, el pentagrama superior, el pentagrama inferior y en la parte inferior los botones que controlan cada una de las reglas.

Primero se construye un *frame;* se abre una aplicación en java, que tiene la función principal la cual mandará traer el *frame* como se muestra en la figura2.

```
package notas;
public class Notas {
 public static void main(String[] args) {
 ventana v = new ventana();
   }
}
```

Figura 2.- código de la clase *ventana*. Fuente propia

Donde *ventana* es el nombre de la clase donde se va a crear, el *frame "v"* es un objeto para mandar llamar la clase.

A continuación en la figura3 se muestra la clase ventana perteneciente a la super clase *frame* se crea el *Jframe.*

```
package frame;
import javax.swing.*;
public class ventana extends JFrame{
   public ventana() {
      setTitle( "Tesis" );
      this.setSize(1200, 700);
      setVisible(true);
   }
}
```

Figura 3.- código para generar el *frame.* Fuente propia

setTitle: nos permite ponerle un título al *frame.*

setSize: dará el tamaño en "x" y en "y" del *frame,* con respecto a los valores que demos como parámetros. En este caso 1200 en "x" y 700 en "y".

setVisible(true): nos permite hacer visible el *frame,* poniendo como parámetro *true*.

3.2 Construcción del panel

Ahora se construirá un contenedor sobre el *frame* un *panel*, el cual se encargará de mantener los objetos en él como los botones, pentagramas, *label,* menú y demás.

Dentro de la clase panel se construirá un llamado a otra clase, donde se creará el panel como se muestra en la figura4.

```
public ventana() {
        setTitle( "Tesis" );
        this.setSize(1200, 700);
        setVisible(true);
        panelclass p = new panelclass();
       add(p);
    }
```

Figura 4.- código de la clase *paneclass.* Fuente propia.

En la clase llamada *panelclass* que pertenece a nuestra primera clase, a la super clase creamos un objeto llamado *"p"*, y lo adicionamos a la clase poseedora del *frame*, con *add(p).*

Dentro de la clase *panelclass,* se puede realizar el panel como se muestra en la figura5.

```
package panel;
import javax.swing.*;
import java.awt.*;
public class panelclass extends JPanel{
    public void paintComponent(Graphics g){
    super.paintComponent(g);
    g.setColor(Color.WHITE);
    g.fillRect(0, 0, 1200,700);
    }
}
```

Figura 5.- código de las dimensiones del *panel.* Fuente propia.

Dentro de la clase usamos *extends JPanel* para, decirle al programa que, vamos a trabajar con un panel.

Luego en una función cuyo parámetro es, *(Graphics g)* daremos las proporciones del panel. Las funciones que trabajan con gráficos necesitan el parámetro antes mencionado.

g.setColor(Color.WHITE): le da al panel un color blanco.

g.fillRect(0, 0, 1200,700): rellena el panel en forma rectangular con las dimensiones establecidas en el parámetro del color antes especificado en *x* e *y*, en este caso 1200 en *x* y 700 en *y*.

3.3 Construcción de los pentagramas.

Teniendo el *frame* y el panel como contenedor, se puede empezar a dibujar los objetos en este. Para el pintado de los pentagramas se muestra en la siguiente figura:

```
public void paintComponent(Graphics g){
g.drawImage(imagensol, -5,72,this);
    g.drawLine(10,77,805,77);
    g.drawLine(10,85,805,85);
    g.drawLine(10,93,805,93);
    g.drawLine(10,101,805,101);
    g.drawLine(10,109,805,109);
    g.drawLine(10,157,805,157);
    g.drawLine(10,165,805,165);
    g.drawLine(10,173,805,173);
    g.drawLine(10,181,805,181);
    g.drawLine(10,189,805,189);
}
```

Figura 5.- código para pintar los pentagramas. Fuente propia.

Dentro de la función *public void paintComponent(Graphics g).* Se usan estas instrucciones para generar dos pentagramas.

drawLine: traza una línea recta en las coordenadas que proporcionadas como parámetro, generan así un pentagrama superior e inferior, en los parámetros se encuentran las coordenadas en *x* e *y* del punto donde inicia el trazado de la recta y en donde finaliza esta.

drawImage: se encarga de colocar una imagen que haya sido cargada dentro del paquete del proyecto java, en las coordenadas que tenga como parámetro en *x* y *y*, en este caso -5 para *x* y 72 para *y*, el otro parámetro *this* es un puntero hacia la imagen.

Ya solo falta el trazado de las líneas divisoras de compas para tener generado los pentagramas como se muestra en la figura7:

```
public void dibujarlinea(Graphics g){
    int i;
    for(i=0;i<l.length;i++){
    g.drawLine(l[i],77,l[i],189);
   }
 }
```

Figura 7.- código para trazar las líneas divisoras de compas. Fuente propia.

3.4 Pintado de notas

Para poder pintar cualquier nota en alguna parte de, alguno de los dos pentagramas, además de cargar la imagen de la nota que queremos, daremos clic en la posición deseada, así la nota quedará pintada. La posición de las notas en un *ArrayList* el cual maneja las posiciones de las notas que se usan, como se muestra en la figura8:

```
soprano.add(new Point( getX,getY));
```

Figura 8.- código del *ArrayList*. Fuente propia.

Se tiene un *ArrayList* llamado soprano este va ir almacenando la posición *x* y *y* que vaya marcándose al dar clic, en los lugares establecido en los parámetros *getX,getY* en la figura9 se muestra el ejemplo:

```
if(getY>=70 && getY<=74){ab[cont]=26;}
        if(getY>=75 && getY<=79){ab[cont]=22;}
        if(getY>=80 && getY<=82){ab[cont]=18;}
        if(getY>=83 && getY<=87){ab[cont]=13;}
        if(getY>=88 && getY<=90){ab[cont]=9;}
        if(getY>=91 && getY<=95){ab[cont]=5;}
        if(getY>=96 && getY<=98){ab[cont]=1;}
        if(getY>=99 && getY<=103){ab[cont]=-2;}
        if(getY>=104 && getY<=106){ab[cont]=-6;}
        if(getY>=107 && getY<=111){ab[cont]=-10;}
        if(getY>=112 && getY<=116){ab[cont]=-14;}
```

Figura 9.- código para marcar la posición de las notas en el pentagrama. Fuente propia.

De esta manera se indica el almacenamiento en la posición de “y” en el pentagrama de alguna voz activa, para así se almacene en algún espacio o línea de algún pentagrama de la voz soprano.

Al hacer clic donde indica *getY*>= y *getY*<=

Por lo cual queda almacenada la posición de la imagen en el arreglo tipo entero *ab[cont]*

Donde *cont* va incrementando conforme se va dando clic para almacenar más imágenes ver figura11.

```
cont++
```

Figura 11.- código del contador incrementado. Fuente propia.

Este incremento se da al final de la función de la voz activa.

Para ir almacenando el pintado de las diversas notas a continuación se explica.

3.5 Pintado de notas cuarto, mitad y entero

Para poder pintar las notas cuarto en los pentagramas se requiere cargar la imagen de la nota cuarto como se muestra en la figura12:

```
Image imagennegra = new ImageIcon(getClass().getResource("negra.png")).getImage();
```

Figura 12.- código para cargar una imagen. Fuente propia.

Donde se indica el nombre de la imagen y el tipo que es. En este caso del tipo *png*, así se tiene la imagen con borde transparente.

Después hay que ir almacenando la posición en donde se requiere la imagen, junto con el almacenamiento y pintado de las mismas ya explicado anteriormente.

Las notas se irán pintando en las líneas o espacios del pentagrama según sea necesario.

Ahora teniendo almacenada la posición de las imágenes ahora se almacenará la imagen cuarto como se muestra en la figura13:

```
imagensoprano[cont] = imagennegra;
```

Figura 13.- código del *array* de imágenes. Fuente propia.

imagensoprano: es un array de tipo imágenes.

imagennegra: es el nombre del archivo de la imagen cuarto.

Para pintar una nota cuarto en la voz soprano el fragmento de código queda estipulado en la figura14:

```
if(ventana.boton_negra && seleccion ==1){
        imagensoprano[cont] = imagennegra;
        if (cont==0){
            posicionX[cont] = a;
            compasSoprano++;
           }
        else if(compasSoprano==0){
            posicionX[cont] = l[lineasSoprano++] + k/2;
            compasSoprano++;
            }
        else{
            posicionX[cont] = posicionX[cont-1] + k+4;
            compasSoprano++;
            }
         cont++;
      }
```

Figura 14.- código para pintar notas de cuarto. Fuente propia.

Para el pintado de las notas cuarto en las demás voces que faltan contralto, tenor y bajo es de una forma similar lo que cambia es la selección de la voz.

Para el pintado de notas mitad o entero es necesario solo seleccionar el ítem de la nota, a elegir de cualquiera de las tres opciones de nota y el proceso idéntico.

Como se muestra en la figura15:

```
if(ventana.boton_redonda && seleccion == 1
        && Do_5==false && La_6==false){
       imagensoprano[cont] = imagenredonda;
       if (cont==0){
           posicionX[cont]= a+30;
           compasSoprano=4;
        }else{
           posicionX[cont]=l[lineasSoprano++]+k+k/2;
           compasSoprano=4;
       }
       cont++;}
       if(compasSoprano==4){
              compasSoprano=0;
            }
```

Figura 15.- código para pintar notas de unidad. Fuente propia.

Pintado de notas en voz soprano ver figura16

```
if(ventana.boton_blanca && seleccion == 1
       && Do_5==false && La_6==false){
        imagensoprano[cont] = imagenblanca;
        reglas_l_8.xnotasSoprano[cont] = 2;
        if(cont==0){
           posicionX[cont] = a;
           compasSoprano+=2;
           }else if(compasSoprano==0){
           posicionX[cont] = l[lineasSoprano++]+k;
           compasSoprano+=2;
         }else{
           posicionX[cont] = posicionX[cont-1]+k+k;
           compasSoprano+=2;
          }
        cont++;}
```

Figura 16.- código para pintar notas en voz soprano. Fuente propia.

3.6 Construcción de los menús

Para poder explicar las notas con alteraciones ligadas o combinaciones ligadas alteradas, hay que explicar los menús.

En los menús se maneja como pintar notas alteradas o con líneas complementarias.

Así como la selección de armadura ó salir del programa.

Primero se crea una barra de menú como se muestra en la figura17:

```
MenuBar menub;
```

Figura 17.- código de la barra de menú. Fuente propia.

La barra contendrá los diversos menús, como se muestra en la figura18:

```
Menu menu,menu2,menu3,menu4,menu6,menu7;
```

Figura 18.- código de los menús. Fuente propia.

Dentro de cada menú habrá contenidos diversos ítems en cada uno en los diversos menús.

3.7 Construcción del menú archivo

Para la construcción del primer menú se realiza como se muestra en la figura19:

```
menu = new Menu( "Archivo" );
```

Figura 19.- código del menú *Archivo*. Fuente propia.

En este menú encontraremos el siguiente ítem

3.8 Construcción del ítem "Salir"

En este ítem, se ejecuta la acción de cerrar el programa, el código para hacerlo se muestra en la figura20.

```
    menu.add( new MenuItem( "Salir" ) );
    menub.add( menu );
```

Figura 20.- código del *ítem* salir. Fuente propia.

3.9 Construcción del menú notas

En este menú se encuentran, diversos ítems que ayudarán a seleccionar las otras notas, para pintarlas en algunos de los dos pentagramas.

El código para el menú archivo se muestra en la figura21.

```
    menu2 = new Menu( "Notas" );
```

Figura 21.- código del menú *notas.* Fuente propia.

3.10 Construcción de los ítems cuarto, mitad y entero

En estos diversos ítems se encuentran las tres figura de nota que se pueden pintar, la nota cuarto, mitad y entero.

Al seleccionar alguna de estas notas y después de seleccionar alguna voz que se encuentra en otro menú, se puede empezar a pintar.

El código para los *ítems* de las notas se muestra en la figura22.

```
menu2.add(cuarto);
menu2.add(mitad);
menu2.add(entero);
```

Figura 22.- código de los *ítems menu2.* Fuente propia.

3.11 Construcción del menú voces

En el menú voces vamos a encontrar los ítems, que nos permitirán seleccionar las cuatro voces, contenidas en la reglas de armonía musical.

Cuando se selecciona una voz y una nota ya se puede empezar a pintar las diversas notas. Dependiendo la selección de la voz se crea un rectángulo de color que abarca el rengo de la voz donde se podrá pintar la nota correspondiente; en la voz soprano se genera un rectángulo de color verde, en la voz contralto, tenor y bajo un rectángulo amarillo, rosa y azul respectivamente. El código para el menú voces se muestra en la figura23.

```
menu3 = new Menu("Voces");
```

Figura 23.- código del menú *voces*. Fuente propia.

3.12 Construcción de los ítems Soprano, Contralto, Tenor y Bajo

En estos ítems seleccionamos las voces en las cuales, se pintarán las diversas notas, y las reglas de armonía evaluaran.

Cada voz es independiente en cuanto al almacenamiento de notas, ya que cada voz almacena sus propias notas y cada nota posee su propio valor numérico que sirve para evaluar las diversas reglas .

En la voz soprano las notas serán de color negro.

En la voz contralto las notas serán de color azul.

En la voz tenor las notas serán de color rojo.

Y por ultimo en la voz bajo las notas serán de color amarillo.

El código para los ítems de las diversas voces se muestra en la figura24.

```
menu3.add(new MenuItem("Soprano"));
menu3.add(new MenuItem("Contralto"));
menu3.add(new MenuItem("Tenor"));
menu3.add(new MenuItem("Bajo"));
```

Figura 24.- código de los *ítems* de las voces. Fuente propia.

3.13 Construcción del menú armadura

En este menú se seleccionan dos armaduras musicales una en tonalidad de ***do*** y otra en tonalidad de la menor, para poder evaluar la regla 15 y la regla 16, en el capítulo de las reglas mencionadas se habló de estas.

El código para el menú armadura se muestra en la figura25.

```
menu4 = new Menu("Armadura");
```

Figura 25.- código del menú *armadura*. Fuente propia.

3.14 Construcción de los ítems do y mi bemol

Cuando se selecciona el ítem ***mi bemol***, se pintarán al inicio del primer pentagrama después de la imagen de la clave de sol y la imagen de cuatro cuartos, se pintará tres signos de bemol, el primero se posicionará en la tercera línea el segundo en el último espacio y el tercero en el segundo espacio del pentagrama respectivamente.

Pero si se selecciona el ítem de do desaparecerán los signos de bemol. El código para los ítems do y la menor se muestra en la figura26.

```
menu4.add(new MenuItem("do"));
menu4.add(new MenuItem("mi bemol"));
```

Figura 26.- código de los *ítems* **do** y **mi bemol**. Fuente propia.

3.15 Construcción del menú NA

En este menú se van a encontrar las diversas notas auxiliares, las cuales se pintan afuera de los pentagramas.

Para poder pintar notas auxiliares, usando el software se requiere seleccionar la nota que se va a pintar, la voz en donde quiere trabajar y el ítem correspondiente a la nota auxiliar a pintar.

El código para el menú *NA* se muestra en la figura27.

```
menu6 = new Menu("N A");
```

Figura 27.- código del menú **NA**. Fuente propia.

3.16 Construcción de los ítems Do5, RE5, Mi5, Fa5, Sol5, La5, La6, Si4, La4, Sol4 y Mi3

Estos son los ítems para seleccionar la nota auxiliar la cual se desea pintar en alguno de los pentagramas y en alguna de las cuatro voces.

El código para los ítems de las notas auxiliares se muestra en la figura28.

```
static MenuItem Do55 = new MenuItem("Do 5");
static MenuItem Re55 = new MenuItem("Re 5");
static MenuItem Mi55 = new MenuItem("Mi 5");
static MenuItem Fa55 = new MenuItem("Fa 5");
static MenuItem Sol55 = new MenuItem("Sol 5");
static MenuItem La55 = new MenuItem("La 5");
static MenuItem La66 = new MenuItem("La 6");
static MenuItem Si44 = new MenuItem("Si 4");
static MenuItem La44 = new MenuItem("La 4");
static MenuItem Sol44 = new MenuItem("Sol 4");
static MenuItem Mi33 = new MenuItem("Mi 3");
```

Figura 28.- código de los *ítems* de notas auxiliares. Fuente propia.

3.17 Construcción del menú herramientas

En este menú se van a encontrar los ítems, que ayudarán a seleccionar la alteración musical necesaria.

Así se podrá pintar alguna nota musical con una alteración. Se necesita seleccionar la nota, la voz y la alteración musical, que se desea se pinte a la izquierda de la nota. El código para el menú *Herramientas* se muestra en la figura29.

```
menu7 = new Menu("Herramientas");
```

Figura 29.- código del menú *Herramientas.* Fuente propia.

3.18 Construcción de los ítems Sostenido, Bemol y Becuadro

El software cuenta con tres alteraciones musicales sostenidas, bemoles y becuadros.

Así podremos seleccionar la alteración musical que se pintara en la nota.

El código para los ítems de alteraciones musicales se muestra en la figura30.

```
menu7.add( new MenuItem( "Sostenido" ) );
menu7.add( new MenuItem( "Becuadro" ) );
menu7.add( new MenuItem( "Bemol" ) );
menub.add( menu );
menub.add(menu2);
menub.add(menu3);
menub.add(menu4);
menub.add(menu6);
menub.add(menu7);
```

Figura 30.- código de los *ítems* de las alteraciones musicales. Fuente propia.

Después se agrega la barra del menú como se muestra en la figura31:

```
setMenuBar( menub );
```

Figura 31.- código de la barra de menú. Fuente propia.

Por último se debe usar la sentencia de la figura32:

```
setVisible(true);
```

Figura 32.- código de la sentencia *setVisible.* Fuente propia.

Para hacer visibles los menús con sus ítems correspondientes.

3.19 Construcción de las etiquetas.

Dentro del software se visualizan los botones para evaluar algunas reglas de armonía musical, situadas debajo de los pentagramas.

Los botones de las reglas están divididos en seis secciones. Arriba de cada sección se encuentran una etiqueta las cual nos dice el tipo de regla, que los botones correspondientes a su sección evaluará.

El código de la primera sección se muestra en la figura33.

```
etiqueta1 = new JLabel("Reglas de armonía para acordes de quinta");
    etiqueta1.setSize(400,20);
    etiqueta1.setLocation(200,250);
```

Figura 33.- código de la etiqueta 1. Fuente propia.

El código de la segunda sección se muestra en la figura34.

```
etiqueta2 = new JLabel("Reglas de armonía para el  enlace de los acordes de quinta");
    etiqueta2.setSize(400,20);
    etiqueta2.setLocation(200,300);
```

Figura 34.- código de la etiqueta 2. Fuente propia.

El código de la tercera sección se muestra en la figura35.

```
etiqueta3 = new JLabel("Reglas de armonía  para acordes de sexta");
    etiqueta3.setSize(400,20);
    etiqueta3.setLocation(200,355);
```

Figura 35.- código de la etiqueta 3. Fuente propia.

El código de la cuarta sección se muestra en la figura36.

```
etiqueta4 = new JLabel("Reglas de armonía para acordes  en modo menor");
    etiqueta4.setSize(400,20);
    etiqueta4.setLocation(200,405);
```

Figura 36.- código de la etiqueta 4. Fuente propia.

El código de la quinta sección se muestra en la figura37.

```
etiqueta5 = new JLabel("Reglas de armonía para acordes de cuarta y sexta");
    etiqueta5.setSize(400,20);
    etiqueta5.setLocation(200,455);
```

Figura 37.- código de la etiqueta 5. Fuente propia.

El código de la sexta sección se muestra en la figura38.

```
etiqueta6 = new JLabel("Reglas de armonía para la resolución del acorde de séptima de dominante");
    etiqueta6.setSize(430,20);
    etiqueta6.setLocation(200,505);
    El código para los números de las voces
    vozuno = new JLabel("1");
     vozuno.setSize(20,20);
     vozuno.setLocation(750,15);
     add(vozuno);
     vozdos = new JLabel("2");
     vozdos.setSize(20,20);
     vozdos.setLocation(770,15);
     add(vozdos);
     voztres = new JLabel("3");
     voztres.setSize(20,20);
     voztres.setLocation(790,15);
     add(voztres);
     vozcuatro = new JLabel("4");
     vozcuatro.setSize(20,20);
     vozcuatro.setLocation(810,15);
     add(vozcuatro);
```

Figura 38.- código de la etiqueta 6. Fuente propia.

new JLabel sirve para crear la etiqueta correspondiente.

setSize(x,y) con esta instrucción se indica el tamaño de la etiqueta.

setLocation(x,y) se puede posicionar la etiqueta en la coordenada “x” y “y” que se ingresan en los paréntesis como parámetros.

3.20 Construcción de los botones

El software contiene diversos botones; situados arriba en la parte derecha se encuentran los botones para borrar las notas, debajo de los pentagramas están los botones para escuchar las notas escritas ya sea las contenidas en las cuatro voces o cada voz por separado. Más abajo están los botones para evaluar las reglas de armonía.

3.21 Construcción de los botones para audio

Los botones para el audio son aquellos que vamos a usar para escuchar las notas, que se escriben en los pentagramas; hay cinco botones para el audio; el primer botón *reproducir*

todo sirve para escuchar simultáneamente las notas de las cuatro voces; el segundo *reproducir soprano*, como dice su nombre permite escuchar las notas escritas en la voz soprano, así como *reproducir contralto*, *reproducir tenor* y *reproducir bajo* y ayudarán a oír las notas que pintemos en voz esas voces.

El código para el botón *reproducir todo* se muestra en la figura39.

```
corre = new JButton("►Reproducir Todo");
        corre.setSize(150,15);
        corre.setLocation(70, 230);
        corre.addActionListener(this);
        add(corre);
```

Figura 39.- código del botón ►*Reproducir Todo*. Fuente propia.

El código para el botón *reproducir soprano* se muestra en la figura40.

```
corre1 = new JButton("►Reproducir Soprano");
        corre1.setSize(160,15);
        corre1.setLocation(227, 230);
        corre1.addActionListener(this);
        add(corre1);
```

Figura 40.- código del botón ►*Reproducir Soprano*. Fuente propia.

El código para el botón *reproducir contralto* se muestra en la figura41.

```
corre2 = new JButton("►Reproducir Contralto");
        corre2.setSize(168,15);
        corre2.setLocation(397, 230);
        corre2.addActionListener(this);
        add(corre2);
```

Figura 41.- código del botón ►*Reproducir Contralto*. Fuente propia.

El código para el botón *reproducir tenor* se muestra en la figura42.

```
corre3 = new JButton("►Reproducir Tenor");
        corre3.setSize(160,15);
        corre3.setLocation(583, 230);
        corre3.addActionListener(this);
        add(corre3);
```

Figura 42.- código del botón ►*Reproducir Tenor*. Fuente propia.

El código para el botón *reproducir tenor* se muestra en la figura43.

```
corre4 = new JButton("►Reproducir Bajo");
        corre4.setSize(160,15);
        corre4.setLocation(750, 230);
        corre4.addActionListener(this);
        add(corre4);
```

Figura 43.- código del botón ►*Reproducir Bajo*. Fuente propia.

new JButton(" ") se usa para crear un botón, el nombre del botón se pone adentro del paréntesis entre comillas.

Corre1, corre2, corrre3 y *corre4* son variables del tipo *JButton*

setSize sirve para darle tamaño a los botones

setLocation le dará la posición a los botones

addActionListener activará el evento del botón presionado.

Add presentaáa el botón en pantalla.

3.22 Construcción de los botones para borrar notas

Los botones para borrar las notas están posicionados en la parte superior derecha del software.

Estos sirven para el borrado de la o las notas pintadas en alguno de los pentagramas, el borrado de nota se hace en forma de pila, sea la última nota que se pinta primera nota que se borra.

Se hizo de esta forma ya que en los ejercicios de armonía alguno de los compases que esté mal perjudica a los siguientes compases que se sigan escribiendo.

Hay cuatro botones para borrar las notas, uno por cada voz

Hay un botón para borrar notas en voz soprano, voz contralto, voz tenor y voz bajo.

El código para estos botones se muestra en la figura44, figura45, figura46 y figura47.

```
borrarUltimaNotaSoprano = new JButton("Borrar Soprano");
        borrarUltimaNotaSoprano.setSize(125,15);
        borrarUltimaNotaSoprano.setLocation(350+kiss,15);
        borrarUltimaNotaSoprano.addActionListener(this);
        add(borrarUltimaNotaSoprano);
        borrarUltimaNotaSoprano.setEnabled(false);
```

Figura 44.- código del botón *Borrar Soprano.* Fuente propia.

```
borrarUltimaNotaContralto = new JButton("Borrar Contralto");
    borrarUltimaNotaContralto.setSize(130,15);
    borrarUltimaNotaContralto.setLocation(480+kiss,15);
    borrarUltimaNotaContralto.addActionListener(this);
    add(borrarUltimaNotaContralto);
    borrarUltimaNotaContralto.setEnabled(false);
```

Figura 45.- código del botón *Borrar Contralto.* Fuente propia.

```
    borrarUltimaNotaTenor = new JButton("Borrar Tenor");
    borrarUltimaNotaTenor.setSize(125,15);
    borrarUltimaNotaTenor.setLocation(350+kiss,35);
    borrarUltimaNotaTenor.addActionListener(this);
    add(borrarUltimaNotaTenor);
    borrarUltimaNotaTenor.setEnabled(false);
```

Figura 46.- código del botón *Borrar Tenor.* Fuente propia.

```
    borrarUltimaNotaBajo = new JButton("Borrar Bajo");
    borrarUltimaNotaBajo.setSize(125,15);
    borrarUltimaNotaBajo.setLocation(480+kiss,35);
    borrarUltimaNotaBajo.addActionListener(this);
    add(borrarUltimaNotaBajo);
    borrarUltimaNotaBajo.setEnabled(false);
```

Figura 47.- código del botón *Borrar Bajo.* Fuente propia.

borrarUltimaNotaBajo es el nombre de la variable tipo *JButton*

new JButton ayuda a crear un botón

setSize nos da el tamaño del botón

setLocation nos da a localización del botón en la pantalla

addActionListener controla la acción del botón

add(borrarUltimaNotaBajo) añade el botón al programa

borrarUltimaNotaBajo.setEnabled(false) hace visible el botón

3.23 Construcción de los *JCheck Box*

Los *JCheck Box* ayudarán para hacer combinaciones de notas por ejemplo

El *JCheck Box* de notas auxiliares nos ayuda para pintar, como su nombre las notas auxiliares

El *JCheck Box* alteraciones para pintar las notas con alguna alteración

El *JCheck Box* apoyar para combinar notas auxiliares ligadas y alteradas.

El *JCheck Box* ligadura nos ayuda a pintar notas musicales ligadas.

El código para los *JCheckBox* se muestra en la figura48.

```
notaaux = new JCheckBox("Notas Auxilliares");
      notaaux.setSize(121,20);
      notaaux.setLocation(10,20);
      alteraciones = new JCheckBox("Alteraciones");
      alteraciones.setSize(97,20);
      alteraciones.setLocation(150,20);
      apoyo = new JCheckBox("Apoyar");
      apoyo.setSize(65,20);
      apoyo.setLocation(270,20);
      ligadura = new JCheckBox("Ligadura");
      ligadura.setSize(75,20);
      ligadura.setLocation(360,20);
new JCheckBox(" ")= crea un JCheckBox, se pone el nombre entre paréntesis y comillas
 setSize(x,y)= especifica el tamaño del JCheckBox.
setLocation(x,y)= pone el JCheckBox en posición x y y.
```

Figura 48.- código del *JCheckBox.* Fuente propia.

3.24 Construcción de los botones para las reglas de armonía musical

En la parte inferior del software se encuentran los botones que se encargan de evaluar las reglas de armonía programadas en este trabajo.

Cada botón se encarga de evaluar la regla correspondiente en el capítulo sobre las reglas.

Para que el programa verifique alguna regla debe de cumplir la condición requerida en su respectiva regla.

Al presionar algún botón de alguna regla, aparecerá un *frame* que nos mostrara un encabezado, explicando de que consta la regla, los botones están enumerados del 1 al 21, que son el número de reglas programadas.

Sise colocan notas en las voces correspondientes a alguna regla y se da clic a su respectivo botón con su respectiva regla, aparecerá el encabezado de la regla mas la evaluación de la regla si esta correcta o incorrecta.

La evaluación de la regla se hace dependiendo el tiempo musical, contenido como un elemento del *array.* Si es el tiempo 1 es el elemento 1 del *array* y así sucesivamente hasta el enésimo elemento. En la figura 49 se muestra el código de los 21 botones para las reglas de armonía.

```
regla1 = new JButton("Regla 1");
        regla1.setSize(80,20);
        regla1.setLocation(10,250+kiss-75);
        regla2 = new JButton("Regla 2");
        regla2.setSize(80,20);
        regla2.setLocation(93,250+kiss-75);
        regla3 = new JButton("Regla 3");
        regla3.setSize(80,20);
        regla3.setLocation(176,250+kiss-75);
        regla4_a = new JButton("Regla 4-a");
        regla4_a.setSize(88,20);
        regla4_a.setLocation(259,250+kiss-75);
        regla4_b = new JButton("Regla 4-b");
        regla4_b.setSize(88,20);
        regla4_b.setLocation(349,250+kiss-75);
        regla4_c = new JButton("Regla 4-c");
        regla4_c.setSize(88,20);
        regla4_c.setLocation(440,250+kiss-75);
        regla5_a = new JButton("Regla 5-a");
        regla5_a.setSize(88,20);
        regla5_a.setLocation(530,250+kiss-75);
        regla5_b = new JButton("Regla 5-b");
        regla5_b.setSize(88,20);
        regla5_b.setLocation(620,250+kiss-75);
        regla6 = new JButton("Regla 6");
        regla6.setSize(80,20);
        regla6.setLocation(200,280+kiss-50);
        regla7 = new JButton("Regla 7");
        regla7.setSize(80,20);
        regla7.setLocation(283,280+kiss-50);
        regla8 = new JButton("Regla 8");
        regla8.setSize(80,20);
        regla8.setLocation(373,280+kiss-50);
        regla9 = new JButton("Regla 9");
        regla9.setSize(80,20);
        regla9.setLocation(460,280+kiss-50);
        regla_10 = new JButton("Regla 10");
        regla_10.setSize(83,20);
        regla_10.setLocation(10,310+kiss-30);
        regla11 = new JButton("Regla 11-a");
        regla11.setSize(95,20);
        regla11.setLocation(95,310+kiss-30);
        regla11_a = new JButton("Regla 11-b");
        regla11_a.setSize(95,20);
        regla11_a.setLocation(193,310+kiss-30);
        regla12 = new JButton("Regla 12");
        regla12.setSize(95,20);
        regla12.setLocation(290,310+kiss-30);
        regla12_a = new JButton("Regla 12-a");
        regla12_a.setSize(95,20);
        regla12_a.setLocation(389,310+kiss-30);
```

```
regla12_b = new JButton("Regla 12-b");
regla12_b.setSize(95,20);
regla12_b.setLocation(489,310+kiss-30);
regla12_c = new JButton("Regla 12-c");
regla12_c.setSize(95,20);
regla12_c.setLocation(589,310+kiss-30);
regla12_d = new JButton("Regla 12-d");
regla12_d.setSize(95,20);
regla12_d.setLocation(695,310+kiss-30);
regla12_e = new JButton("Regla 12-e");
regla12_e.setSize(95,20);
regla12_e.setLocation(800,310+kiss-30);
regla13 = new JButton("Regla 13");
regla13.setSize(83,20);
regla13.setLocation(900,310+kiss-30);
regla14 = new JButton("Regla 14");
regla14.setSize(83,20);
regla14.setLocation(990,310+kiss-30);
regla15 = new JButton("Regla 15");
regla15.setSize(83,20);
regla15.setLocation(200,340+kiss-10);
regla16 = new JButton("Regla 16");
regla16.setSize(83,20);
regla16.setLocation(290,340+kiss-10);
regla_17 = new JButton("Regla 17");
regla_17.setSize(83,20);
regla_17.setLocation(200,340+kiss+40);
regla18 = new JButton("Regla 18");
regla18.setSize(83,20);
regla18.setLocation(290,340+kiss+40);
regla19 = new JButton("Regla 19");
regla19.setSize(83,20);
regla19.setLocation(200,370+kiss+60);
regla20 = new JButton("Regla 20");
regla20.setSize(83,20);
regla20.setLocation(300,370+kiss+60);
regla21 = new JButton("Regla 21");
regla21.setSize(83,20);
regla21.setLocation(400,370+kiss+60);
add(regla1);
add(regla2);
add(regla3);
add(regla4_a);
add(regla4_b);
add(regla4_c);
add(regla5_a);
add(regla5_b);
add(regla6);
add(regla7);
add(regla8);
add(regla9);
```

```
        add(regla_10);
        add(regla11);
        add(regla11_a);
        add(regla12);
        add(regla12_a);
        add(regla12_b);
        add(regla12_c);
        add(regla12_d);
        add(regla12_e);
        add(regla13);
        add(regla14);
        add(regla15);
        add(regla16);
        add(regla_17);
        add(regla18);
        add(regla19);
        add(regla20);
        add(regla21);
       regla1.addActionListener(this);
        regla2.addActionListener(this);
        regla3.addActionListener(this);
        regla4_a.addActionListener(this);
        regla4_b.addActionListener(this);
        regla4_c.addActionListener(this);
        regla5_a.addActionListener(this);
        regla5_b.addActionListener(this);
        regla6.addActionListener(this);
        regla7.addActionListener(this);
        regla8.addActionListener(this);
        regla9.addActionListener(this);
        regla_10.addActionListener(this);
        regla11.addActionListener(this);
        regla11_a.addActionListener(this);
        regla12.addActionListener(this);
        regla12_a.addActionListener(this);
        regla12_b.addActionListener(this);
        regla12_c.addActionListener(this);
        regla12_d.addActionListener(this);
        regla12_e.addActionListener(this);
        regla13.addActionListener(this);
        regla14.addActionListener(this);
        regla15.addActionListener(this);
        regla16.addActionListener(this);
        regla_17.addActionListener(this);
        regla18.addActionListener(this);
        regla19.addActionListener(this);
        regla20.addActionListener(this);
        regla21.addActionListener(this);
```

Figura 49.- código de los 21 botones para las reglas de armonía musical. Fuente propia.

new JButton= crea un botón

setSize(x,y)= da tamaño al botón

setLocation(x,y)=especifica la posición del botón

add(regla n)= adiciona el botón n a la *frame*

regla n.addActionListener(this)=permite la acción al botón.

3.25 Mapa conceptual

Para facilitar la visión y comprensión de las ideas principales expuestas se presenta en la figura50 el mapa conceptual:

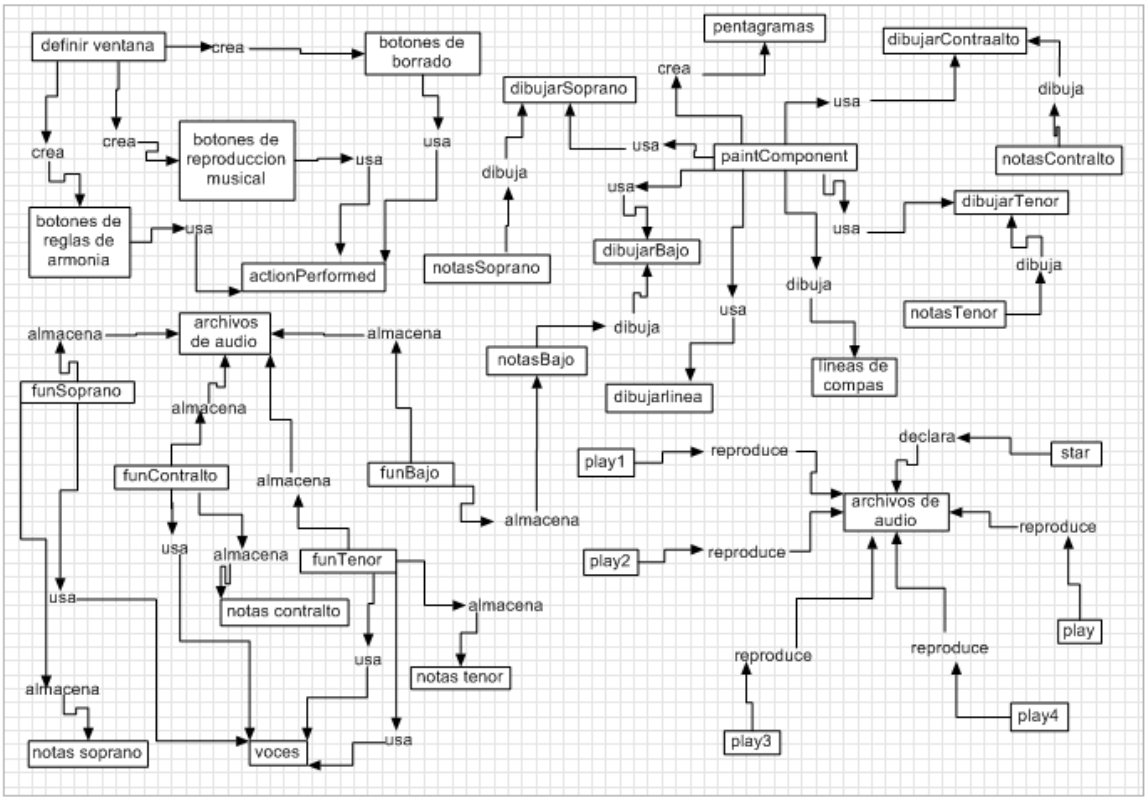

Figura 50.- Mapa conceptual: Uso del software *armonía M@Puma* Fuente propia

Capítulo 4

Estudio de campo

La licenciatura en Matemáticas Aplicadas y Computación proveyó un conocimiento general y solido para realizar el análisis, diseño, desarrollo e implantación de las herramientas propuestas en el presente trabajo, a través de sus asignaturas empleando creativa y racionalmente las matemáticas y técnicas computacionales.

Por la versatilidad de la carrera, se puede interactuar fácilmente con profesionistas de otras disciplinas y responder a las necesidades de crear y entender a posibles cambios tecnológicos en la rama del software libre, así como, la capacidad de analizar, innovar y tomar decisiones sobre el comportamiento de software educativo y sus procesos industriales.

4.1 Variables

- Inconformidades. (Puede haber disconformidad de los alumnos porque el software está en una sola tonalidad y solo un compas musical de cuatro cuartos).
- Complicaciones. (Puede resultarles muy complicado su uso).
- Consistencias. (Esperan una interfaz grafica más amigable y poderosa).

4.2 Marco conceptual

Los conceptos establecidos en el libro *Educación y tecnología un binomio excepcional* sobre lo concerniente al software educativo están bien definidos en base a estos se establece el apoyo de un software para armonía musical, las reglas dadas en el libro *Curso básico de armonía* sirvieron para la lógica de programación del sistema *Armonía M@Puma* aunque la forma en la que están establecidos requería de una interpretación a conciencia, pero con el apoyo de otra fuente como lo es *Trattato d'armonía* ayudo a esclarecer la visión de la armonía musical para poder conceptualizarla como un software, la demás biografía sirvió de apoyo para fortalecer algunas cuestiones de informática para terminar el presente trabajo, al igual que las referencias de internet para usar conceptos necesarios para la culminación del mismo.

Música es el arte y la ciencia de los sonidos.

Software el equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de componentes lógicos necesarios que hacen posible la realización de tareas especificas, en contraposición a los componentes físicos que son llamados hardware.

4.3 Marco metodológico

La necesidad de apoyar a las escuelas de música con un software para el ayudar a los ejercicios de armonía musical lleva a una metodología de una encuesta exploratoria para intentar satisfacer los requerimientos de la materia armonía musical, la opinión de los alumnos en curso a esta área es de gran ayuda para establecer una mejor guía a la construcción de futuros trabajos con el sistema *armonía M@Puma.*

4.3.1 Enfoque metodológico

Es exploratorio.

4.3.2 Técnica

Encuesta estructurada, el software fue prueba.

4.4 Muestreo

El muestreo adoptado al presente trabajo se basa en el alumnado de las escuelas importantes que imparten armonía musical en la zona metropolitana y la ciudad de México, dado que tienen un sistema de escolar muy demandado además de actualizarlo cada ciclo escolar apoyados de tecnología musical eficiente, con maestros de grado reconocidos a nivel nacional y mundial. Siendo estas instituciones accesibles y eficientes para el muestreo de este trabajo.

4.4.1 Universo

Los alumnos de las escuelas de Bellas Artes de Naucalpan, del Conservatorio Nacional de Música y de la Escuela de Mariachis Ollin Yoliztli de Garibaldi que cursan la materia de armonía musical.

4.4.2Tamaño total de la muestra

24 alumnos de la escuela de Bellas Artes de Naucalpan.

44 alumnos de la escuela del Conservatorio Nacional de Música.

13 alumnos de la escuela de mariachis Ollin Yoliztli de Garibaldi.

Total de 81 alumnos.

Nivel de confianza

95%

Margen de error

5%

Modalidad de muestreo

No estratificado los resultados son representativos.

Muestreo probabilístico cualquier elemento puede participar.

Instrumento

Cuestionario.

El tipo de preguntas son abiertas.

El tipo de respuestas son abiertas.

Aplicación 17 de julio del 2015, 28 de noviembre del 2015, 5 de febrero del 2016, 11 de febrero del 2016 y 1 de marzo del 2016.

¿Has notado diferencias al hacer los ejercicios de armonía usando el software y al hacerlos en tu cuaderno? Ver figura51.

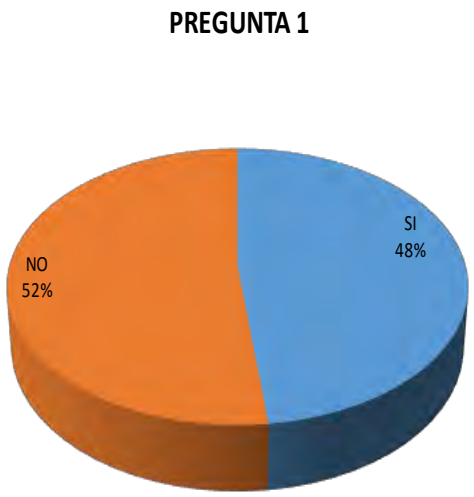

Figura51.- pregunta 1 para evaluar el software *armonía M@Puma.* Fuente propia

Los alumnos recalcan que, el software puede marcarles cuál es el error con respecto a la regla a tratar. Situación muy poco usual con respecto a otro software de música, aunque resulta un poco tardada la revisión.

Siendo estas las diferencias transcendentales, haciendo el comparativo de los ejercicios en el cuaderno y la computadora con el sistema.

¿Consideras muy complicado el uso del software? Ver figura52.

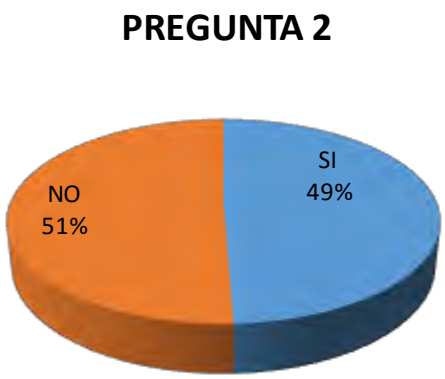

Figura52.- pregunta 2 para evaluar el software *armonía M@Puma*. Fuente propia

La mayoría expreso, el software no es difícil su uso, no tiene herramientas muy complejas. Esta todo lo suficientemente marcado para disponer del software en el manejo de este pero es algo tardado.

¿Te gustaría un examen de armonía usando el software? Ver figura53.

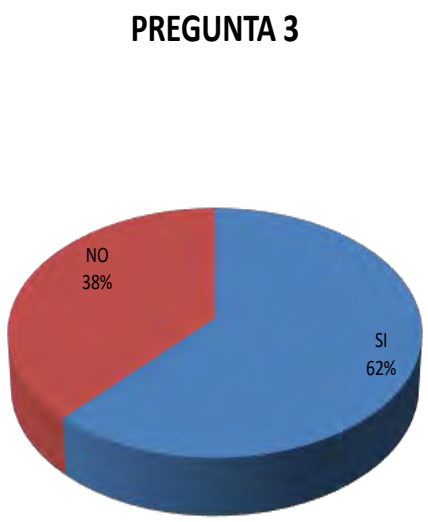

Figura53.- pregunta 3 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría considera bueno hacer un examen con el software para aprender a usarlo mejor y verificar ó corregir errores.

¿Te gustaría una versión mejorada del software? Ver figura54.

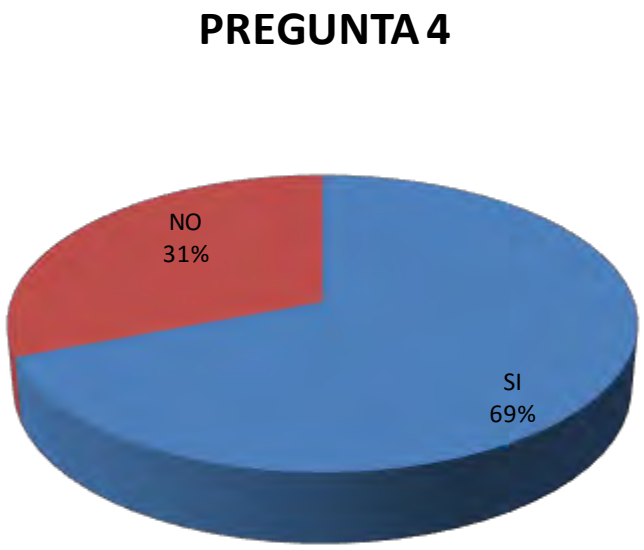

Figura 54.- pregunta 4 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría afirmaron que si, dado que el software solo usa la tonalidad de **"do"**, a ellos les gustaría para su mejor uso, que tuviera por lo menos las otras, seis tonalidades principales en la música como son **"re"**, **"mi"**, **"fa"**, **"sol"**, **"la"** y **"si"**. Además de añadirles las demás reglas de armonía existentes, ya que el software solo cuenta por el momento con las primeras 21 reglas de armonía musical de algunos fragmentos de piezas musicales usados como ejercicios de armonía musical por Isabel Jeremías Lafuente y Enrique Cordero R. de diversos compositores, para un estudio primario de armonía musical.

Dejando estos detalles para posteriores trabajos y estudios tanto de programación como de análisis del problema para un futuro.

¿Notaste algunas fallas al usar el software? Ver figura55.

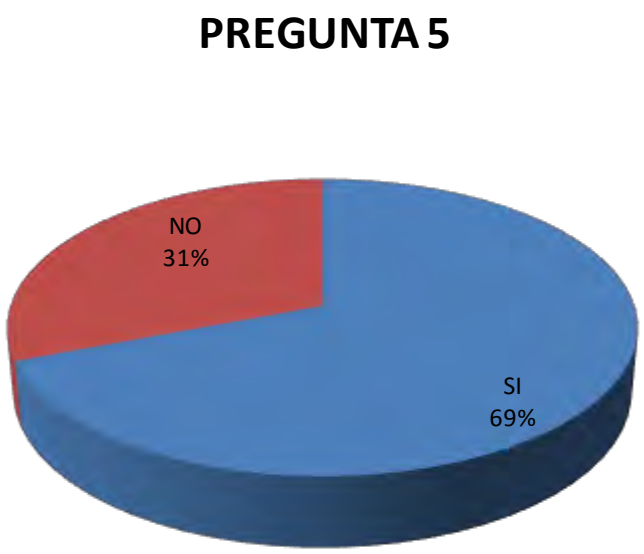

Figura 55.- pregunta 5 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría marco como falla en el software la carencia de las seis tonalidades principales faltantes dado que el programa solo cuenta por el momento con doce compases y tres figuras musicales, básicas como son cuarto, medio y unidad. Dado que con estas figuras y estos compases son suficientes y necesarios para hacer ejercicios de armonía. Además de comentar el pintado de las notas de una forma lenta.

El programa para futuros trabajos se pretende sea una herramienta más poderosa y compleja.

¿Crees necesario la implementación del software en Internet? Ver figura56.

Figura 56.- pregunta 6 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría expreso que es buena idea el software esté disponible en Internet. Para así estar al alcance de otros alumnos en otras partes del mundo, como herramienta de ayuda en los ejercicios de armonía musical, y ellos mismos cuando naveguen por internet disponer del mismo. También les gustaría que hubiera actualizaciones del software vía Internet. Para que la funcionalidad de este mejore.

¿Esperabas una aplicación diferente al usar este software para armonía? Ver figura57.

Figura 57.- pregunta 7 para evaluar el software *armonía M@Puma.* Fuente propia

Considerarón poco complejo el software además de ser muy específico en su forma de usar. Los alumnos esperaban una complejidad mayor en el sistema.

¿Usarías el software para verificar tus tareas de armonía? Ver figura58.

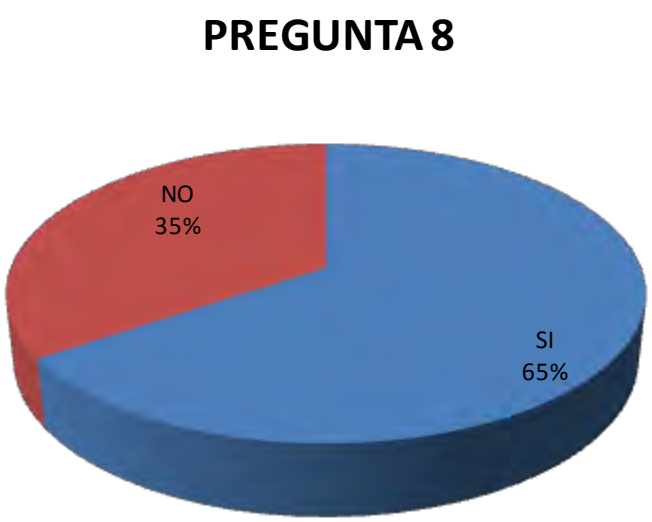

Figura 58.- pregunta 8 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría usarían el software para hacer sus tareas escolares de armonía, dado que les ayudaría a tener una revisión, de sus ejercicios más a fondo y corregirían los errores que se presentan en las tareas.

¿Crees que el software te puede ayudar en el estudio de la armonía? Ver figura59.

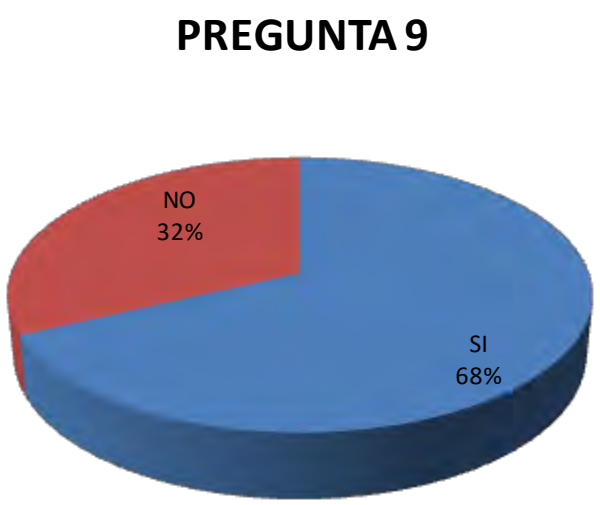

Figura59.- pregunta 9 para evaluar el software *armonía M@Puma.* Fuente propia

El 68% de los encuestados dieron una respuesta positiva, dictaminando el software como una ayuda en los errores, de armonía por su forma de marcar las fallas, al instante de hacer los ejercicios en el programa.

¿Tienes algún comentario o sugerencia sobre el software? Ver figura60.

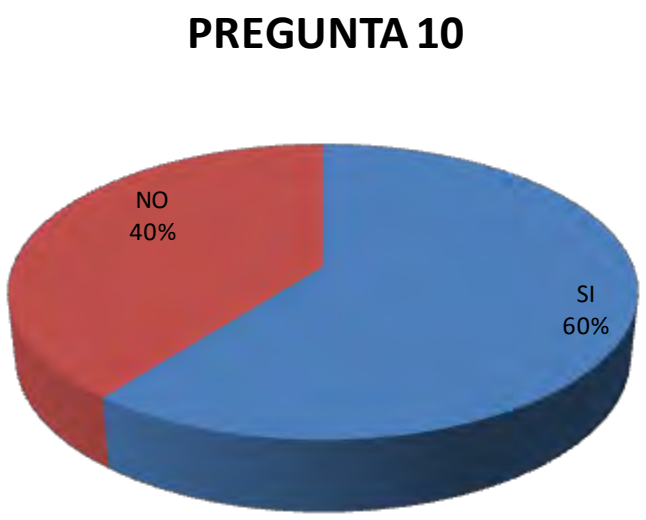

Figura 60.- pregunta 10 para evaluar el software *armonía M@Puma.* Fuente propia

La mayoría de los alumnos encuestados dieron su opinión, sobre el software las cuales estipulan.

Les gustaría un mejor diseño visual del software, no les gusto la forma poco estética, como está diseñado, el tamaño de los botones, los ítems tan simple.

Que sea una herramienta con más tonalidades musicales. ya que solo cuenta por el momento con la tonalidad de **"do"**, faltando las otras seis tonalidades principales como **"re"**, **"mi"**, **"fa"**, **"sol"**, **"la"** y **"si**" y dejando estos inconvenientes para futuros trabajos.

Dado al tiempo en que vivimos, con dispositivos telefónicos con *touch,* les gustaría que el software contara con una herramienta, la cual les ayudaría a pintar las notas de una forma más rápida y fácil. Que los errores sean marcados con color rojo al momento de hacerlos.

Conclusiones

El software de carácter educativo se está, volviendo ampliamente aceptada como ayuda de enseñanza en las escuelas. Con el desarrollo de esta tesis se tuvo la oportunidad de practicar el uso de la manipulación de imágenes para crear pentagramas musicales, aplicando archivos de audio en sincronía y un proceso de desarrollo para guiar un proyecto a través del análisis y diseño con éxito.

La escuela de Bellas de Artes de Naucalpan, el Conservatorio Nacional de Música y la escuela de Mariachis Ollin Yoliztli de Garibaldi desarrollan los ejercicios de armonía musical sin un apoyo de corrección, guiado solamente en el enfoque del maestro en turno. El problema principal es la complejidad al momento de revisión de los ejercicios de armonía musical.

La aplicación del software *"Armonía M@Puma"* para el apoyo a la materia armonía musical tuvo un efecto inmediato en conectar a los alumnos con las reglas de armonía musical. Por que proporcionó a los alumnos una interfaz grafica apoyado con *frames* conteniendo la terminología bien definida, del problema de armonía musical a trabajar, entre los individuos y grupos fue una forma interactiva rápida de estar vinculado con la armonía musical.

La metodología para el análisis de la armonía musical con el apoyo de un software libre en estado alfa con el sistema *"Armonía M@Puma"* los siguientes beneficios:

- facilito la comunicación y comprensión de las reglas de armonía musical establecidos en el software aplicando un formato grafico.
- reforzó la habilidad de los alumnos para resolver ejercicios de armonía musical.
- proporcionó una herramienta de apoyo para estudios posteriores de armonía musical.
- aumentó el interés por la materia armonía musical tanto teórico como auditivo.
- exitosamente entrego una aplicación computacional para verificar las tareas de armonía musical.

Además de haber obtenido las siguientes experiencias con los ejercicios de armonía musical realizados en el salón de clase apoyados del sistema *Armonía M@Puma* fueron las siguientes:

- en la nota Si4 en la voz tenor trataban de pintar la nota como nota auxiliar, impidiendo el sistema esta acción, dado que no es una nota auxiliar

- en la escuela de Bellas Artes de Naucalpan, ayudó el software a reforzar varias carencias en la materia debido al poco interés de algunos alumnos por aprender temas básicos como el rango de las voces, que acordes se forman con las diversas triadas etc. Problemática que no sucedió en el Conservatorio Nacional de Música dado que en esa escuela el nivel académico tiene es alto y estricto.

- Algunos errores del sistema fueron que al pintar las notas sin marcar los pasos estipulados en el apéndice para uso correcto del sistema se generaban algunas notas fuera del rango de los pentagramas que entorpecían a los ejercicios de armonía musical por lo que es necesario una revisión a fondo de este problema en trabajos posteriores.

- Otro error marcado fue la falta de un sistema de armonía musical cromático es decir con mas tonos ya que el sistema solo cuenta con una de las doces tonalidades existentes en la música, originando este problema que los ejercicios tenían que transportarlos a la única tonalidad existente en el software que es **do.**

- la gran mayoría remarco que escribir las notas es muy lento, ya que están acostumbrados a usar herramientas como *sibelius* que tiene opciones flexibles en la escritura musical.

- El porcentaje de verificación de errores del sistema con respecto los ejercicios de armonía musical fue 100% cumplido

- El porcentaje de satisfacción de los alumnos con la aplicación *Armonía M@Puma* según lo arrojado por las encuestas fue de un 68% pero remarcando un 94% que se debe mejorar de tal forma que sea dinámico.

- los profesores de armonía musical consideran una herramienta necesaria para la verificación de los ejercicios de armonía musical dado que hace falta un apoyo académico en la materia.

De acuerdo a los resultados que arrojan las encuestas que hicieron los alumnos de la escuela de Bellas Artes de Naucalpan, el Conservatorio Nacional de Música y la escuela de Mariachis Ollin Yoliztli de Garibaldi el software Armonía M@Puma ayuda a detectar errores en los ejercicios de armonía musical con éxito es viable el uso de software libre de armonía musical que ayuda a verificar los ejercicios de armonía musical para su educación, la cual confirma nuestra hipótesis.

Apéndice

Sistema para la ayuda de

Ejercicios para armonía musical

Armonía M@Puma

A.1 Presentación del sistema *Armonía M @Puma*

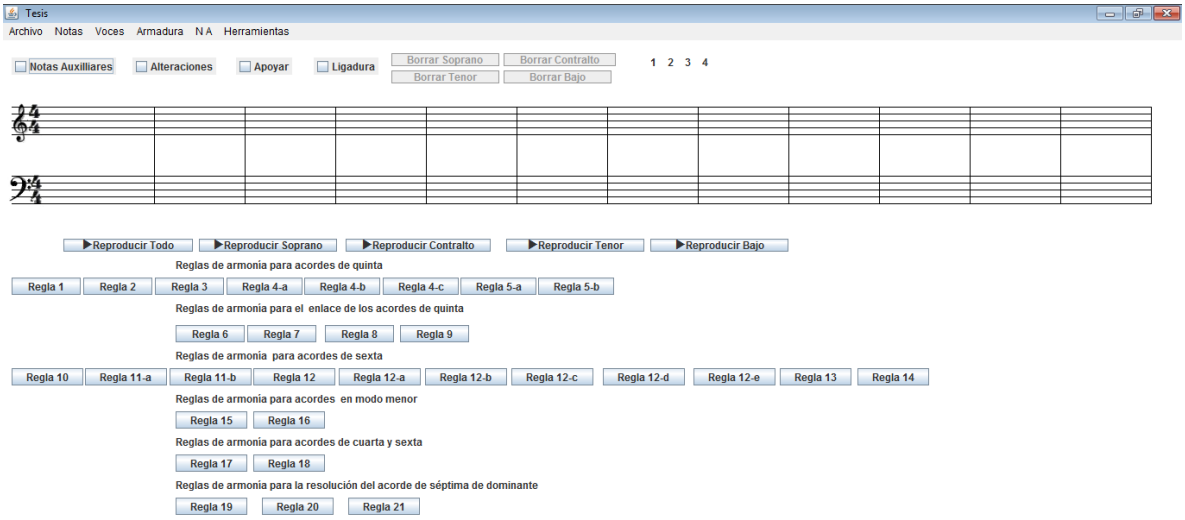

Fig. A.1 Pantalla de inicio.

La página principal permite a los usuarios iniciar con el sistema.

En la parte superior los ítems con los cuales podrá seleccionar los signos

Musicales correspondientes para su ejercicio de armonía.

En la parte central los dos pentagramas donde podrá escribir las notas.

En la parte inferior se encuentran los botones donde evaluará las reglas de armonía musical correspondiente al ejercicio resultante.

A.2 Selección de la voz soprano en el sistema *Armonía M @Puma*

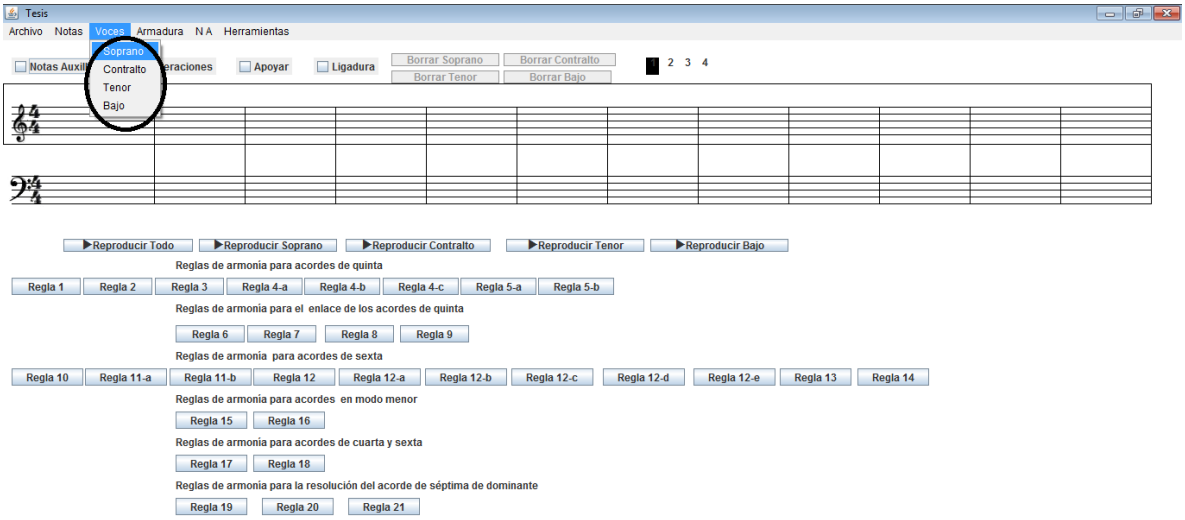

Fig. A.2 Pantalla de selección de voz.

Una vez seleccionado el menú *voces*. El sistema nos mostrará los ítems de las cuatro voces.

Se da clic a alguna de ellas y mostrará con algún rectángulo de color y un número de la parte superior derecho el rango de la voz seleccionada.

En este caso la voz soprano aparece con un rectángulo negro y el número 1 pintado rectangularmente de negro.

A.3 Selección de notas cuarto en el sistema *Armonía M @Puma*

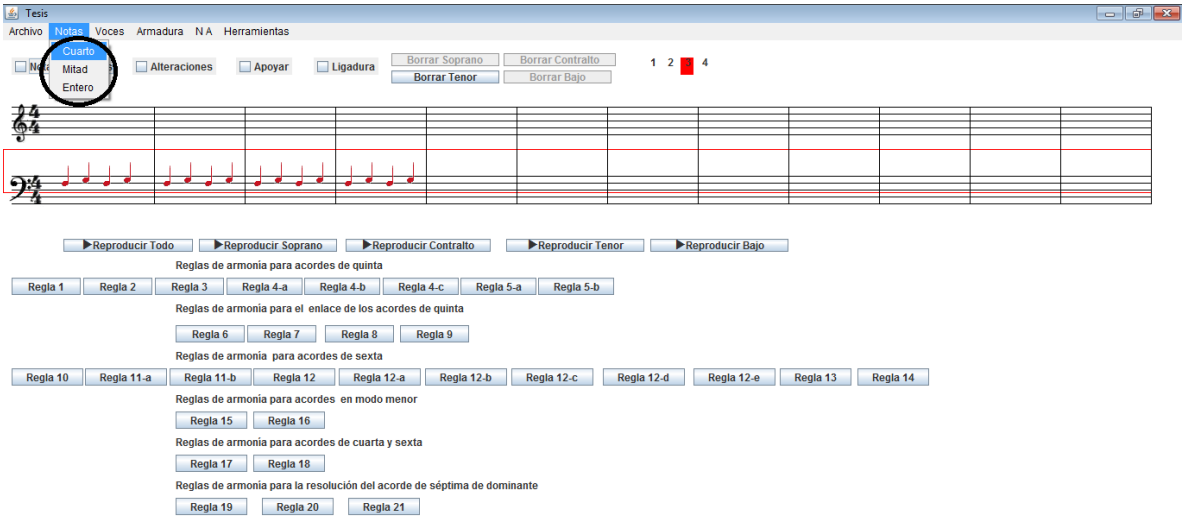

Fig. A.3 Pantalla selección de nota musical.

Para empezar a escribir las notas en alguno de los dos pentagramas, será necesario que después de activar alguna de las cuatro voces, se active el ítem *Notas.* Después se debe seleccionar la nota correspondiente; en este caso se escogió *cuarto*. Enseguida damos clic, a alguna línea o espacio del pentagrama correspondiente, donde deseamos poner la nota en cuestión.

A.4 Borrado de notas en el sistema *Armonía M @ Puma*

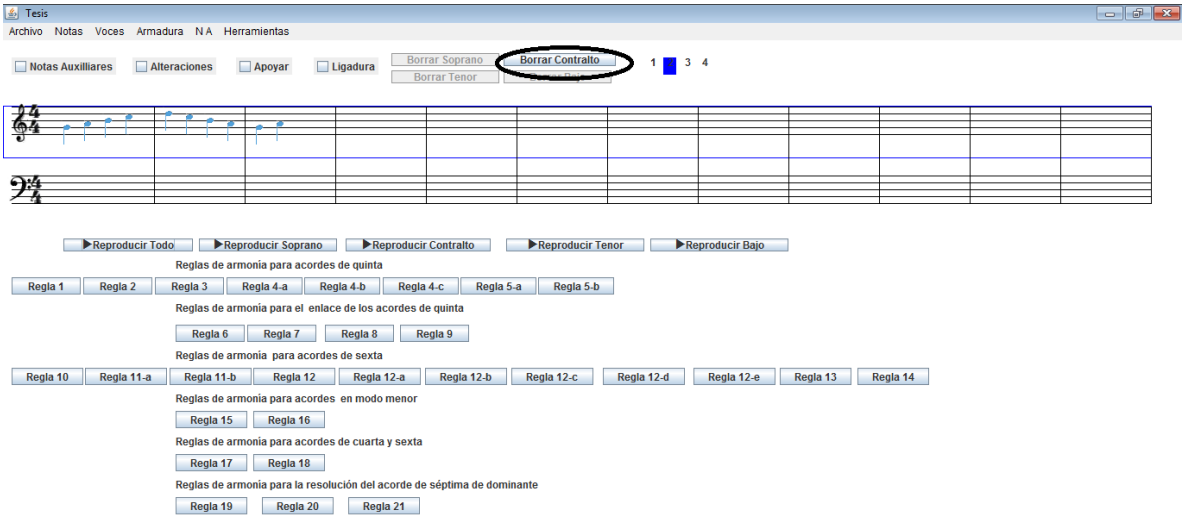

Fig. A.4 Pantalla de borrado de nota musical.

El borrado de notas se ejecuta en forma de cola, es decir la última nota que se pone es la primera que se borrará y así sucesivamente hasta borrar todas las notas o solo las deseadas.

Para borrar alguna nota, hay que posesionarse primero en una de las cuatro voces donde se encuentra la nota a eliminar, luego damos clic a alguno de los cuatro botones de borrado que esta activa dependiendo la voz activa; en este caso la voz Contralto será borrada dando clic con el botón *Borrar Contralto*.

A.5 Pintado de notas cuarto sostenidas en voz bajo en el sistema *Armonía M@Puma.*

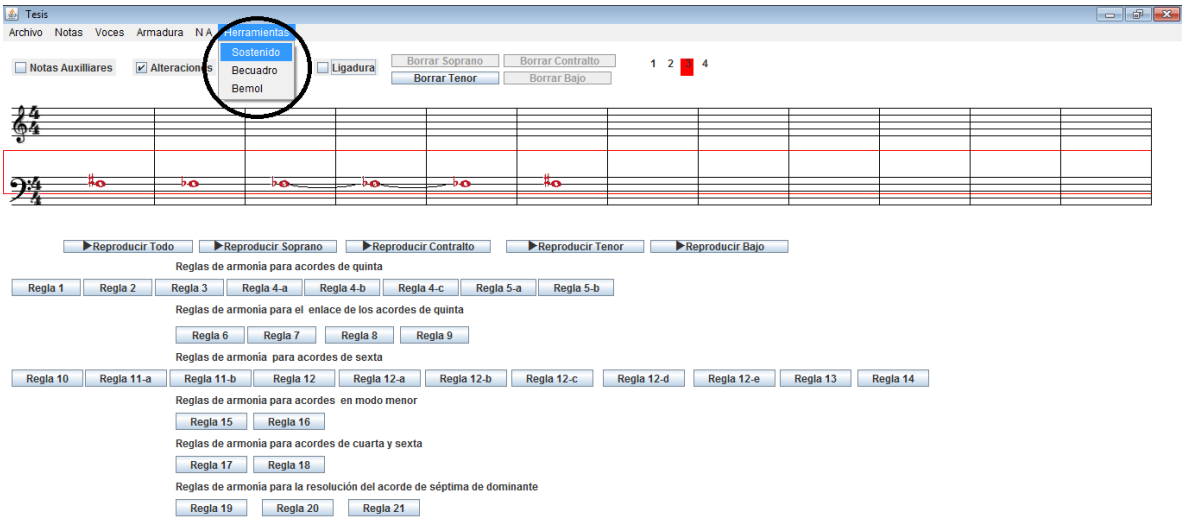

Fig. A.5 Pantalla de selección de alteración musical.

Para colocar notas con alguna alteración musical, ya sea sostenido, bemol, ó becuadro, ya sea con ligadura o sin ligadura.

- seleccionamos la voz donde se va escribir las notas.
- seleccionamos la nota que se desea escribir.
- damos clic al *JCheckBox* Alteraciones la cual quedará seleccionada con una paloma.
- después nos vamos al ítem “Herramientas” y seleccionamos alguna de las alteraciones musicales ya mencionadas.
- para ligar la nota con alteración musical, seleccionamos también el *JCheckBox* Ligadura, la nota se pintará con la alteración y con ligadura.

En este caso la última nota pintada fue unidad con sostenido en el primer compas musical, redonda con sostenido en el segundo compas, en el tercero, cuarto y quinto compases unidad ligada con bemol.

A.6 Pintado de notas con líneas auxiliares en voz soprano en el sistema Armonía *M@Puma*

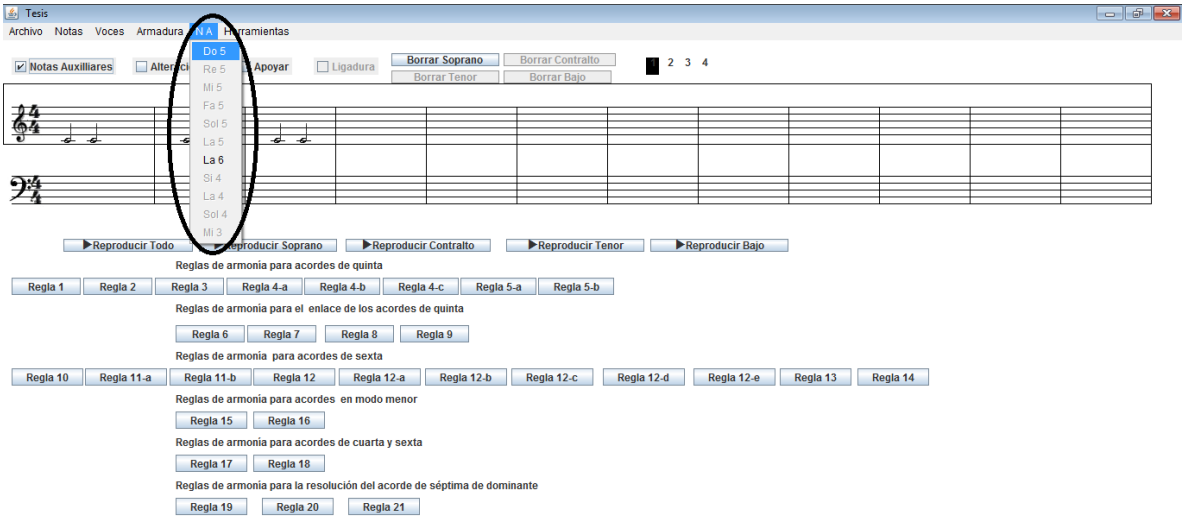

Fig. A.6 Pantalla de selección de nota auxiliar.

El pintado de las notas en las líneas auxiliares se realiza de esta forma:

- se escoge la voz donde se va a colocar dichas notas, ítem *voces*.
- se selecciona la nota, ítem *Notas*.
- se da clic al *JCheckBox* Notas auxiliares la cual queda seleccionada con una paloma.
- nos vamos al ítem *N.A.* escogemos la nota auxiliar correspondiente a la voz escogida.

En este caso fue blanca do5 en la voz soprano, y damos clic a la altura donde se encontraría la nota por ejemplo, do5 está debajo de la primera línea, y automáticamente se irán escribiendo notas do5 consecutivamente conforme vayamos dando clic.

La nota auxiliar que no pertenezca a la voz seleccionada quede inhabilitada.

A.7 Pintado de notas con líneas auxiliares en voz Contralto y alteraciones ligadas o no ligadas en el sistema Armonía M @Puma.

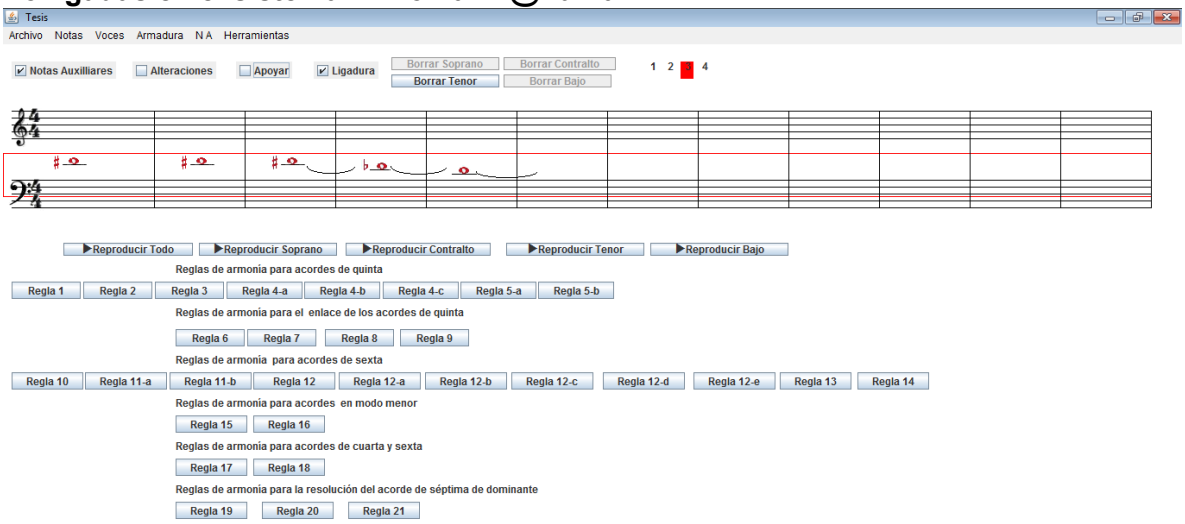

Fig. A.7 Pantalla para selección de nota musical alterada y ligada.

El pintado de las notas en las líneas auxiliares con alteraciones ligadas o no ligadas se realiza de esta forma:

- se escoge la voz donde se va a colocar dichas notas, ítem *voces*.
- se selecciona la nota, ítem *Notas*.
- se da clic al *JCheckBox* Notas auxiliares la cual queda seleccionada con una paloma.
- nos vamos al ítem *N.A.* escogemos la nota auxiliar correspondiente a la voz seleccionada.
- se da clic al *JCheckBox* Alteraciones la cual queda seleccionada con una paloma. Se escoge la alteración deseada para la nota ya sea sostenido, becuadro o bemol en el ítem “Herramientas”.
- Si se desea la nota alterada y ligada en una línea auxiliar se realizan los pasos antes mencionados para pintar las notas con su respectiva alteración y línea

auxiliar en la voz seleccionada. Después se da clic al *JCheckBox* Ligadura y al *JCheckBox* Apoyar así la nota estará alterada y ligada en alguna línea auxiliar.

En este caso fue redonda **re5** en la voz soprano, y damos clic a la altura donde se encontraría la nota por ejemplo, re5 está arriba de la quinta línea del segundo pentagrama, y automáticamente se irán escribiendo notas **re5** consecutivamente conforme vayamos dando clic.

A.8 Rol del alumno en el uso de los botones de las reglas de armonía en el sistema *Armonía* M @Puma.

Una vez ingresado al sistema las notas en su voz correspondiente, se podrá evaluar la regla del acorde al ejercicio de armonía a evaluar.

Se aprecian todos los botones con los cuales cuenta el sistema para verificar reglas de armonía para su corrección. Los botones están seccionados en seis partes, según muestran las etiquetas posesionadas arriba de cada grupo de los botones como son:

- Reglas de armonía para acordes de quinta.
- Reglas de armonía para los enlaces de acordes quinta.
- Reglas de armonía para acordes de sexta.
- Reglas de armonía para acordes en modo menor.
- Reglas de armonía para acorde de cuarta sexta.
- Reglas de armonía para la resolución del acorde de séptima dominante.

Cada sección cuenta con los botones que evaluaran las reglas de armonía ya mencionadas en el capítulo III. Si se desea evaluar la regla 1 en un ejercicio de armonía debe hacer clic al botón marcado como Regla1 y así sucesivamente con las demás reglas establecidas en este trabajo. Después aparecerá un panel en donde el encabezado confirma, de que regla se trata, que se evaluó y abajo le marcara, por renglón en que tiempo si es nota cuarto ó en que compás si es nota entero y el error o acierto, en su ejercicio de armonía; cada reglón estará separado por una línea.

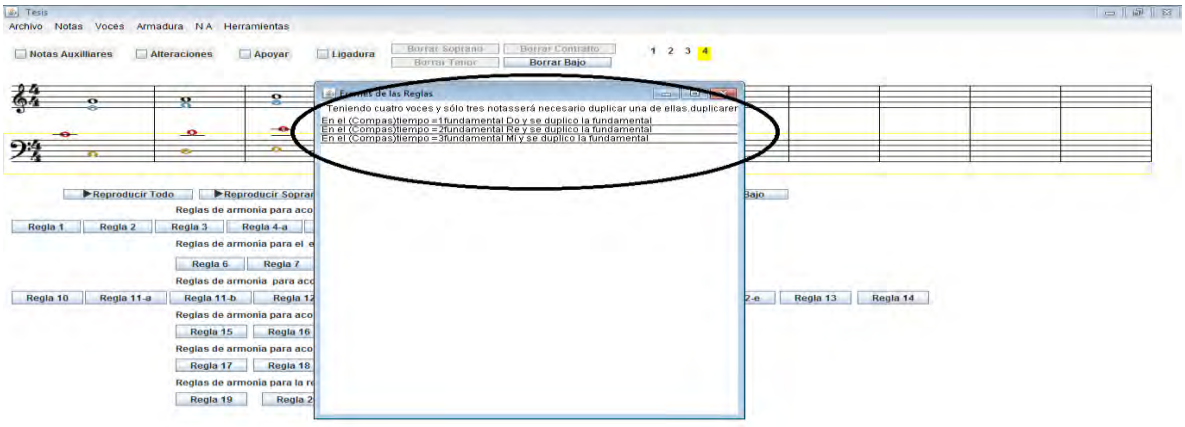

Fig. A.8 Pantalla de corrección de la regla 1 de armonía musical.

Pantalla de captura del sistema Armonía Puma evaluando regla1 de armonía musical.

De esta forma el software muestra los errores ó aciertos en los ejercicios de armonía musical.

Para las demás reglas de armonía estipulados en este trabajo se procede de la misma forma.

Bibliografía

Begoña Gros, A. (1997). *Del software educativo a educar con software.* Barcelona España: Gedisa.

Ceballos Sierra, Francisco Javier (2001). *Java2 curso de programación,* Madrid España: Alfaomega Rama.

Cordero R., Enrique. Jeremías, Isabel (2005). *Curso básico de armonía,* Universidad de Costa Rica: Escuela de Artes Musicales.

Ceja Mena, Luis (2000). *Fortran y métodos numéricos,* México D.F.: Libros de México S.A.

Deitel, Paul. Y Deitel, Harvey (2012). *Java como Programar,* Naucalpan de Juárez Estado de México: Pearson.

De Sanctis, Cesare (1886). *Trattato d'armonía,* Roma Italia: G.Ricordi.

Dr. Fernández Aedo C., Raul. Lic. Delavaut Romero E., Martin (2013). *Educación y tecnología un binomio excepcional,* Madrid España: Grupo Editor K.

García Llinás, Luis Fernando (2010). Todo lo básico que debería saber sobre programación básica orientada a objetos en java, Bogotá Colombia: ediciones Uninorte.

Herrera, Enric (2008). *Teoria musical y armonía moderna,* Barcelona España: Antoni Bosch.

Lago Martínez, Silvia (2015). De tecnologías digitales, educación formal políticas: aportes al debate compilado por. Silvia Lago Martínez, Buenos Aires Argentina: Tesea.

Moncada García, Francisco (1997). *La más sencilla, útil y práctica teoría de la música,* México D.F.: Framong.

Moncada García, Francisco (1996). *Software educativo. Guía de uso y metodología de diseño,* Barcelona España: Estel.

Rangel Fermín, Ana Lisette, (2002). *La teoría tras la producción de software educativo y otras reflexiones,* Caracas Venezuela: Fondo Editorial Humanidades.

Ríos Cabrera, Pablo (1998). Concepción del software educativo desde la perspectiva pedagógica ponencia, Barcelona España.

Roxana Cabello, Diego Levis (2007). *Medios informáticos en la educación a principios del siglo XXI,* Buenos Aires Argentina: Prometeo libros editorial.

Sierra Fernández, José Luis (2005). *Estudio de la influencia de un entorno de simulación por ordenador,* Granada España: Ministerio de educación.

REFERENCIAS EN INTERNET.

Oxford music online.

http://www.oxfordmusiconline.com/public/book/omo_gmo

Consultado los días 8 de mayo 2015, 17 septiembre 2015, 21 de octubre 2015 y 2 de noviembre 2015.

google books.

https://books.google.com.mx/books?id=YwxBnoQeRp4C&pg=PA89&dq=software+educativo&hl=es&sa=X&redir_esc=y#v=onepage&q=software%20educativo&f=false

Consultado el día 19 de agosto 2015.

google books.

https://books.google.com.mx/books?id=IwXbRjhn-TsC&printsec=frontcover&dq=software+educativo&hl=es&sa=X&redir_esc=y#v=onepage&q=software%20educativo&f=false

Consultado el día 22 octubre 2015.

google books.

http://books.google.com.mx/books?id=ZCU6gGcr4DkC&printsec=frontcover&dq=software+educativo&hl=es&sa=X&redir_esc=y#v=onepage&q=software%20educativo&f=false

Consultado el día 24 octubre 2105.

google books.

https://books.google.com.mx/books?id=I4HOAAAACAAJ&dq=software+educativo&hl=es&sa=X&redir_esc=y

Consultado el día 27 octubre 2015.

google books.

https://books.google.com.mx/books?id=JaPTzKZxbN4C&printsec=frontcover&dq=java&hl=es&sa=X&ved=0CDUQ6AEwBGoVChMI97WvyqWGyQIVRv5jCh3MBwst#v=onepage&q=java&f=false

Consultado el día 4 noviembre 2015.

google books.

https://books.google.com.mx/books?id=je1Wv9OaRcAC&printsec=frontcover&dq=armonia+musical&hl=es-419&sa=X&sqi=2&ved=0CBoQ6AEwAGoVChMIwNTNk6uGyQIVyhU-Ch1d0wbP#v=onepage&q=armonia%20musical&f=false

Consultado el día 18 noviembre 2015.

GLOSARIO DE INFORMÁTICA

CheckBox

Este componente, permite seleccionar una opción al usuario del programa o tomar una decisión, directamente en pantalla.

Clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje.

Frames

Es un elemento implementado por Netscape, que permite dividir la pantalla en varias áreas independientes unas de otras, y por tanto con contenidos distintos, aunque puedan estar relacionados.

Hoja de cálculo

Es un tipo de documento, que permite manipular datos numéricos y alfanuméricos dispuestos en forma de tablas compuestas por celdas (las cuales se suelen organizar en una matriz bidimensional de filas y columnas).

Inteligencia artificial

es un área multidisciplinaria, que a través de ciencias como las ciencias de la computación, la matemática, la lógica y la filosofía, estudia la creación y diseño de sistemas capaces de resolver problemas cotidianos por sí mismas utilizando como paradigma la inteligencia humana.

Java

El lenguaje de programación Java es un lenguaje de alto nivel que puede ser caracterizado por ser simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, de alto desempeño, *multihilo* y dinámico.

Label

Este componente se utiliza para desplegar textos o mensajes estáticos dentro de las formas, textos tales como encabezados, solicitud al usuario del programa para que proporcione algún dato o información (edad, dame sueldo, etc.).

Lenguaje de programación

Es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

Mapa conceptual

Es una técnica usada para la representación gráfica del conocimiento. Un mapa conceptual es una red de conceptos. En la red, los nodos representan los conceptos, y los enlaces representan las relaciones entre los conceptos.

Matriz o vector (llamado en inglés *array*)

Es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, los elementos de la matriz. Desde el punto de vista lógico una matriz se puede ver como un conjunto de elementos ordenados en fila (o filas y columnas si tuviera dos dimensiones).

Sistema de gestión de paquetes

También conocido como gestor de paquetes, es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software.

Sistemas expertos

Son llamados así porque emulan el razonamiento de un experto en un dominio concreto, y en ocasiones son usados por éstos.

Sistema informático

Es un sistema que permite almacenar y procesar información; es el conjunto de partes interrelacionadas: hardware, software y personal informático.

Software libre

Refiere el conjunto de *software* que por elección manifiesta de su autor, puede ser copiado, estudiado, modificado, utilizado libremente con cualquier fin y redistribuido con o sin cambios o mejoras.

Versión Alpha / Alfa

Es la primera versión del programa, la cual es enviada a los verificadores para probarla. Algunos equipos de desarrollo utilizan el término alfa informalmente para referirse a una

fase donde un producto todavía es inestable, aguarda todavía a que se eliminen los errores o a la puesta en práctica completa de toda su funcionalidad, pero satisface la mayoría de los requisitos. El nombre se deriva de alfa, la primera letra en el alfabeto griego.

Versión beta o lanzamiento beta

Representa generalmente la *primera versión completa* del programa informático o de otro producto, que es posible que sea inestable pero útil para que las de *inspección previa (preview)* o como una *inspección previa técnica (technicalpreview [TP])*.

GLOSARIO MUSICAL

Acorde

Consiste en un conjunto de tres o más notas diferentes que suenan simultáneamente y que constituyen una unidad armónica.

Alteraciones

Son los signos que se escriben a la izquierda de las notas y sirven para modificar su entonación.

Becuadro

Destruye el efecto del sostenido o del bemol, y hace retornar, por tanto, a su sonido natural a una nota alterada antes con cualquiera de ellos.

Bemol

El bemol hace bajar la entonación de la nota a la que se antepone medio tono.

Blanca

Es una figura musical que equivale a ½ del valor de la figura redonda. Las figuras de blancas se representan con una cabeza de nota ovalada hueca (como la redonda) y con una plica vertical sin adornos (como la negra).

Compás

Es la unidad de medida que sirve para dividir el tiempo en la música.

Clave

Es el signo que se escribe al principio de cada pentagrama y sirve para determinar el nombre y la altura de las notas en la escala general de los sonidos.

Cuarta

Se obtenía con una cuerda de largura tres cuartos de la inicial. Su frecuencia es cuatro tercios de la nota inicial.

Intervalo

Es la diferencia de entonación que existe entre dos sonidos.

Líneas adicionales

Son pequeños fragmentos de línea que, en número limitado, se utilizan para escribir las notas que rebasan las cinco líneas del pentagrama.

Melodía

Melodía es la sucesión de sonidos de diferente altura que, animados por el ritmo, expresan una idea musical.

Modo menor

Tienen como característica más destacada que la distancia entre su primer y tercer grados es de tercera menor (un tono y medio).

Negra

Es una figura musical que equivale a ¼ del valor de la figura redonda. Las figuras de negras se representan con una cabeza de nota ovalada coloreada en negro (de ahí su nombre) y con una plica vertical sin adornos (como la blanca). La dirección de la plica depende de la posición de la nota.

Nota principal

La principal nota de un acorde, a partir de la cual se construyen las siguientes notas del acorde. La nota fundamental coincide con la nota *más grave* del acorde sólo si el acorde está en su posición básica o estado fundamental, es decir que no está invertido.

Notas

Son los signos en forma de ovalo que representan sonidos y sus valores (duración).

Octava

Al intervalo de ocho grados entre dos notas de la escala musical.

Pentagrama

Es el conjunto de cinco líneas horizontales, paralelas y equidistantes donde se escriben los signos musicales.

Primera Inversión

Cuando el bajo canta la 3ª del acorde un acorde de sexta o un acorde en primera inversión.

Quinta

Se obtenía con una cuerda con una ligadura de dos tercios de la inicial. Su frecuencia es de tres medios del sonido inicial.

Redonda

Es una figura musical que posee una duración de cuatro pulsos de negra en la notación musical actual. Las figuras de redondas se representan con una cabeza de nota ovalada hueca (como la blanca), pero sin barra vertical o plica.

Semitono

Es cada una de las dos partes, iguales o desiguales, en que se divide el intervalo de un tono.

Sensible o nota sensible

En el sistema tonal hace referencia al séptimo grado de una escala musical. Según el contexto puede hacer referencia a la séptima nota de la escala, o bien al acorde que se forma sobre dicha nota y/o a la función tonal y sonoridad correspondientes (siendo más frecuente esto último).

Sostenido

El sostenido hace subir la entonación de la nota a la que se antepone medio tono.

Tonalidad

Es la relación establecida entre una serie de sonidos con uno principal llamado tónica, el que rige el funcionamiento de todos los demás.

Tónica o nota tónica

En el sistema tonal hace referencia al primer grado de la escala musical, que es la nota que define la tonalidad.

Tono

Es un intervalo musical que en el sistema temperado es igual a un sexto de octava. La mitad de un tono es un semitono y equivale a un doceavo de octava. En cualquier escala procedente de los modos gregorianos existen 5 tonos que alternan con semitonos siguiendo una secuencia característica de dicha escala.

Unísono

Es un intervalo musical de proporción 1:1 y ningún semitono ni cent. Se considera que dos tonos al unísono tienen la misma altura, pero son perceptibles como si vinieran de fuentes separadas. El unísono es considerado el intervalo más consonante mientras que el semitono es considerado el más disonante. También es el intervalo más fácil de afinar.