Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Tecnológico

Backing Track Generator

Héctor Ignacio Marín Aréchiga
2112004030
Asesores de proyecto terminal

Dra. Laura Elena Chávez Lomelí
Profesor Titular
Departamento de Ciencias Basicas e Ingeniería

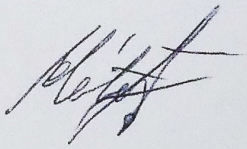
Trimestre 2017 Invierno

3 de marzo de 2017

Yo, Laura Elena Chávez Lomelí, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Laura E. Chávez L.

Yo, Héctor Ignacio Marín Aréchiga, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



1. Resumen

En el ambito musical la creatividad y la práctica son temas de mucha importancia, ya que sin su desarrollo es imposible que el músico alcanze una habilidad que le permita desenvolverse de manera exitosa en el medio. Por lo que en este proyecto se intentará contribuir al proceso de desarrollo de la creatividad, a su vez que se intenta compensar la falta de compañeros de práctica, y mejorar la eficacia de la misma.

Índice

2. Introducción

Hoy en día la música es una industria gigantesca en la cual trabajan millones de personas, su producto principal son varios conjuntos de notas con ritmo, que forman compases, y a su vez esto, en un patrón definido, forma algo llamado canción. Por tal motivo el objetivo de la aplicación Backing Track Generator es ayudar al músico de cualquier nivel, a practicar sus habilidades, aportando una canción, generada a través de un Algoritmo aleatorizado, una nota tónica y un género. Esta podrá ser utilizada por el músico en cualquier forma que encuentre adecuada.

3. Antecedentes

A continuación se listan diversos proyectos existentes de la UAM Azcapotzalco y otras universidades que tienen alguna semejanza con el proyecto que se propone, siguiendo de una breve descripción y las diferencias con este Proyecto:

- "Sistema Clasificador de Música"[1]. Utiliza Redes Neuronales para la clasificación las cuales son una técnica conocida de Cómputo suave, también es de temática musical pero no genera una pista o un archivo de audio
- "Ajuste de Parámetros para Composición Musical" [2]. Hace uso del Algoritmo MMC de Román Mora para realizar cambios en los parámetros de composición Musical, en el caso del proyecto pertinente a este documento, haremos composición musical utilizando un algoritmo aleatorizado propio.
- "Composición Musical a través del uso de Algoritmos Genéticos" [3], Se utilizan algoritmos genéticos y redes Neuronales para estudiar melodías ya existentes e intentar mejorarlas. A diferencia de este, se intentará componer melodías desde cero, sin ninguna base.
- "Modelo Generativo de Composición melódica con expresividad" [4], utiliza modelos estadísticos y estocásticos, para con base en melodías ya existentes crear nuevas con algoritmos evolutivos, nosotros las crearemos con teoría de gráficas y un algoritmo aleatorizado propio.
- "Compositor Automático de Música Aleatoria siguiendo una Melodía Patrón" [5], utiliza una melodía patrón para generar la nueva Composición, nosotros la escribiremos desde cero.
- "Teoría de la Probabilidad en la Composición Musical Contemporánea."[6] estudia la probabilidad y estadística en la música pero no propone ninguna aplicación tecnológica, es un estudio realizado.

4. Justificación

Para el músico la práctica diaria es muy importante, ya que su trabajo depende de que tan bien pueda interpretar y/o construir ideas con sentido en su instrumento, sin embargo al buscar la forma más eficiente de practicar, se encuentra el problema de que practicar escalas, acordes, arpegios, etc. no es suficiente, se requiere una práctica más eficaz.

La creatividad no viene por sí sola, es un trabajo constante, por esto se busca también la mejor manera de fomentarla o al menos una forma de lograr esto.

Es muy importante tomar en cuenta los puntos anteriores, ya que un músico rara vez trabaja en solitario, y aunque sea el caso, las piezas musicales se interpretan con más de un instrumento, por lo que es necesario que la práctica diaria fortalezca la habilidad de trabajar en conjunto.

Por todo esto, esta aplicación ayudará al músico a practicar improvisando, conceptos de armonía y melodía aplicada, sabiendo la escala en la que se encuentra la pieza, y con mas instrumentos de fondo, con lo cual se fortalecerá la habilidad del mismo para trabajar con otros músicos, y poder expresar sus sentimientos en forma de música de la mejor forma que le parezca y sin importar cuantos intentos requiera.

5. Objetivos

5.1. Objetivo general

- Crear una aplicación que sirva para ayudar al Artista musical en su labor de practicar y desarrollarse musicalmente, además de su uso recreativo en la improvisación musical.

5.2. Objetivos específicos

1. Crear un algoritmo aleatorizado que a través de teoría de conjuntos, gráficas y música, sea capaz de construir estructuras de armonía y melodía, representándolas de manera abstracta en el programa en arreglos de objetos.

2. Cambiar la Representación de las estructuras de armonía y melodía de tal modo que puedan ser escuchados por el usuario del programa.

3. Crear piezas musicales a través de software que sean armónicamente correctas.

6. Marco teórico

- Algoritmo aleatorizado – algoritmo que utiliza cierto grado de aleatoriedad en su lógica, para la toma de decisiones en su comportamiento.[7]

- Nota – sonido determinado por una vibración cuya frecuencia fundamental es constante. Así pues, el término «nota musical» se emplea para hacer alusión a un sonido con una determinada frecuencia
- Nota tónica – en el sistema tonal hace referencia al primer grado de la escala musical, que es la nota que define la tonalidad.
- Escala musical – conjunto de sonidos ordenados notas de un entorno sonoro particular (sea tonal o no); de manera simple y esquemática.
- Compás – Entidad Métrica musical que se compone por varias unidades de tiempo, llamadas figuras musicales.

7. Desarrollo del proyecto

El desarrollo del proyecto se llevó a cabo con base en los siguientes módulos, utilizando la librería JMusic 1.6.4 [8], la cual se encarga de la generación de audio y notas en formato mid.

- **Módulo Algoritmo Aleatorizado Generador** Se desarrolló este módulo para llevar a cabo las operaciones de organizacion de las partes de la pieza, organizar la ejecucion de los demas módulos de acuerdo a la entrada del usuario y con ayuda de la libreria JMusic [8] regresar como salida una pieza en formato mid.
- **Módulo Escala** Se desarrolló este módulo a fin de organizar todos los tonos provenientes de la libreria JMusic [8] en escalas de cualidad mayor, a fin de poder utilizarlas en el programa en la creación de lineas melódicas o bases.
- **Módulo Grafo** Se desarrolló este módulo para llevar a cabo todas las operaciones de compatibilidad entre arreglos y crear nuevos arreglos con base en un arreglo base o primordial, teniendo como estructura un grafo, donde los vertices son las notas de la escala y las aristas son el grado de la misma.
- **Módulo Percusiones** Se desarrolló este módulo, para organizar los instrumentos de percusión en diferentes arreglos, que permitan su utilización en la generacion de piezas.
- **Módulo gridRitmo** Se desarrolló este módulo para encargarse de la creación de las figuras rítmicas para todos los instrumentos a excepción de los de percusión, esto lo hace creando patrones rítmicos aleatorios según la entrada del usuario.
- **Modúlo NotaMusical** Se desarrolló este módulo para encargarse de los métodos de creación de notas con ayuda de la librerica JMusic [8], usando

aleatoriedad para elegir la nota a ser creada, o para crear notas arbitrariamente, que puedan ser usadas en otros pasos del programa.

8. Resultados

Se obtuvieron varias muestras tanto de pistas mid por instrumento como pistas de piezas completas, con instrumentos como son Piano, Bajo, Guitarra y Batería sonando concurrentemente

9. Análisis y discusión de resultados

Aunque el usuario elija un género para la pista, es claro que debido a la aleatoriedad, la pieza obtenida no es precisamente lo que se esperaba, sin embargo dentro de las pruebas se han encontrado ejemplos bastante interesantes, esto debido a que muestran patrones melódicos que se acercan en gran medida a lo que un compositor crea en sus ratos libres, y que podrían ser de gran utilidad en el aprendizaje de nuevos ritmos para el músico o nuevos arreglos, que por falta de creatividad un exceso de frustración no se encontrarían de una manera tan sencilla, además de permitir la improvisación sobre las pistas creadas utilizando el instrumento de preferencia del músico, lo cual demuestra la utilidad de este software.

10. Conclusiones

Es claro que la creacion de pistas musicales es un problema bastante difícil para una computadora, ya que no es tan sencillo crear un algoritmo de composición que se asemeje a la forma en la que los humanos componen la música.

Sería más fácil crear un programa con una biblioteca enorme de rítmicas de versos, coros, puentes, etc. que elija aleatoriamente y cree piezas con ellas.

Los géneros musicales son una parte de la música que es definida por la ritmica y las notas empleadas además de otros recursos, la mayoría de las rítmicas se repiten en un mismo género con pocas variaciones, es decir, son intencionales, y es por esto que no pueden ser fácilmente creadas aleatoriamente.

Referencias

[1] S. Rivera Bernal, *"Sistema Clasificador de Música usando el concepto de Memoria Asociativa"*, Proyecto terminal, división de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México. 2010.

[2] D. C. Mercado González, *"Ajuste de Parámetros para el método de Composición Musical"*, Proyecto terminal, división de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México. 2014.

[3] E. MOLDAVER, *"Composición Musical a través del uso de Algoritmos Genéticos"*,esis de Grado de Ingeniería en Informática, Facultad de Ingeniería, Universidad de Buenos Aires, Argentina, pp.11-12

[4] H. A. GARCÍA SALAS, *"Modelo Generativo de Composición melódica con expresividad"*, Tésis de grado Doctoral de Ciencias de la Computación, Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico,2012. Pp.35 -42

[5] M. J. Inoñan Moran, *"Compositor Automático de Música Aleatoria siguiendo una Melodía Patrón"*, esis de grado de Ingeniería en electrónica, Pontificia Universidad Católica del Perú, Perú ,2010, pp. 2 - 10

[6] S. Tiburcio Solís, *"Teoría de la Probabilidad en la Composición Musical Contemporánea"*. Tésis de Grado Licenciatura, Benemérita Universidad Autónoma de Puebla Escuela de Artes,México, pp.81-100

[7] R. Motwani *"Randomized Algorithms"*, Libro, Cambridge university Press,Estados Unidos De America, 1995.

[8] Andrew Sorensen and Andrew Brown, *"jMusic Music Composition in Java"*, Exploding Art http://explodingart.com/jmusic, 1998

11. Entregables comprometidos en la propuesta

- **Las pistas se adjuntarán en el disco del Proyecto**
- **Código Fuente**
- Módulo Algoritmo Aleatorizado Organizador

```
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;
import jm.music.data.*;
import jm.JMC;
import jm.audio.*;
import jm.music.tools.Mod;
import jm.util.*;

/**
 *
 * @author MARIN
 */
public class algoritmoAleatorizadoOrganizador implements JMC {
```

```
public algoritmoAleatorizadoOrganizador() {

}

public static void main(String Args[]) {

    int SILENCE = -2147483648;
    ArrayList<Part> DRock = new ArrayList<>();
    ArrayList<Part> DPop = new ArrayList<>();
    ArrayList<Part> DBalad = new ArrayList<>();
    ArrayList<Part> DBlues = new ArrayList<>();
    ArrayList<notaMusical> notasbase = new ArrayList<>();
       //arreglo de notas para el grid de ritmo
    ArrayList<notaMusical> notasacom = new ArrayList<>();
    ArrayList<notaMusical> notasacc2 = new ArrayList<>();
    ArrayList<notaMusical> notasbase2 = new ArrayList<>();
       //arreglo de notas para el grid de ritmo
    ArrayList<notaMusical> notasacom2 = new ArrayList<>();
    ArrayList<notaMusical> notasacc22 = new ArrayList<>();
    ArrayList<notaMusical> notasbase3 = new ArrayList<>();
       //arreglo de notas para el grid de ritmo
    ArrayList<notaMusical> notasacom3 = new ArrayList<>();
    ArrayList<notaMusical> notasacc23 = new ArrayList<>();
    double sum = 0; //doble para la suma de ritmos y su
       terminacion
    Percusiones g0 = new Percusiones();
    g0.createDrums(0.0,DPop, DRock, DBlues, DBalad);  //
       init for drums
    Part v1 = new Part("Vprimero", FRETLESS, 0);
    Part v2 = new Part("Vsegundo", GUITAR, 1);
    Part v3 = new Part("Vtercer", PIANO, 2);
    Part v12 = new Part("Cprimero", FRETLESS, 0);
    Part v22 = new Part("Csegundo", GUITAR, 1);
    Part v32 = new Part("Ctercer", PIANO, 2);
    Part v13 = new Part("Pprimero", FRETLESS, 0);
    Part v23 = new Part("Psegundo", GUITAR, 1);
    Part v33 = new Part("Ptercer", PIANO, 2);
    Scanner sc = new Scanner(System.in);


    System.out.println("Bienvenido a Backing Track
       Generator.");
    System.out.println("Favor de ingresar Escala (mayor).")
       ;
    String escala = sc.next();
    System.out.println("Favor de ingresar Tiempo para la
       pieza (bpm),");
    int tempo = sc.nextInt();
    System.out.println("Favor de ingresar genero deseado.")
       ;
    System.out.println("Opciones: Rock, Pop, Ballad, Blues"
       );
```

```
String genre = sc.next();
System.out.println("Ingrese  instrumentos seguidos de
    un enter");
String ins1 = sc.next();
System.out.println("");
String ins2 = sc.next();
System.out.println("");
String ins3 = sc.next();
System.out.println("");
Escala e1 = new Escala(escala.toUpperCase());
Grafo gx = new Grafo();

Part drums = g0.getRdrums(genre, DPop, DRock, DBlues,
    DBalad);

g0.createDrums(16.0,DPop, DRock, DBlues, DBalad);
        Part drums2 =  g0.getRdrums(genre, DPop, DRock,
            DBlues, DBalad);

g0.createDrums(32.0,DPop, DRock, DBlues, DBalad);
Part drums3 = g0.getRdrums(genre, DPop, DRock, DBlues,
    DBalad);

g0.createDrums(48.0,DPop, DRock, DBlues, DBalad);
Part drums4 =  g0.getRdrums(genre, DPop, DRock, DBlues,
     DBalad);

g0.createDrums(64.0,DPop, DRock, DBlues, DBalad);
Part drums5 = g0.getRdrums(genre, DPop, DRock, DBlues,
    DBalad);

g0.createDrums(80.0,DPop, DRock, DBlues, DBalad);
Part drums6 =  g0.getRdrums(genre, DPop, DRock, DBlues,
     DBalad);

gridRitmo ritmobase = baserit("base", ins1, genre);
notasbase = base_notes("base", ins1, ritmobase, e1);
Phrase basephrase = base_phrase(notasbase);

gridRitmo ritmobase2 = baserit("base", ins1, genre);
notasbase2 = base_notes("base", ins1, ritmobase2, e1);
Phrase basephrase2 = base_phrase(notasbase2);

gridRitmo ritmobase3 = baserit("base", ins1, genre);
notasbase3 = base_notes("base", ins1, ritmobase3, e1);
Phrase basephrase3 = base_phrase(notasbase3);




Phrase vhpr2 = verse("VersoAc1", ins2, genre, gx,
    ritmobase, notasbase, e1);
```

```
Phrase vhpr22 = verse("CoroAc1", ins2, genre, gx,
    ritmobase2, notasbase2, e1);
Phrase vhpr23 = verse("PuenteAc1", ins2, genre, gx,
    ritmobase3, notasbase3, e1);

Phrase vhpr3 = verse("VersoAc2", ins3, genre, gx,
    ritmobase, notasbase, e1);
Phrase vhpr32 = verse("coroAc2", ins3, genre, gx,
    ritmobase2, notasbase2, e1);
Phrase vhpr33 = verse("PuenteAc2", ins3, genre, gx,
    ritmobase3, notasbase3, e1);

Score verse = new Score();




v1.add(basephrase);
v1.appendPhrase(basephrase);
v1.appendPhrase(basephrase);
v1.appendPhrase(basephrase);

v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);

v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);
v1.appendPhrase(basephrase2);

v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);
v1.appendPhrase(basephrase3);

v2.add(vhpr2);
v2.appendPhrase(vhpr2);
v2.appendPhrase(vhpr2);
v2.appendPhrase(vhpr2);
v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);

v2.appendPhrase(vhpr23);
v2.appendPhrase(vhpr23);
v2.appendPhrase(vhpr23);
```

```
v2.appendPhrase(vhpr23);

v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);
v2.appendPhrase(vhpr22);

v2.appendPhrase(vhpr23);
v2.appendPhrase(vhpr23);
v2.appendPhrase(vhpr23);
v2.appendPhrase(vhpr23);

v3.add(vhpr3);
v3.appendPhrase(vhpr3);
v3.appendPhrase(vhpr3);
v3.appendPhrase(vhpr3);

v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);

 v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);

 v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);
v3.appendPhrase(vhpr32);

v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);
 v3.appendPhrase(vhpr33);

verse.addPart(v1);
verse.addPart(v2);
verse.addPart(v3);
verse.addPart(drums);


verse.setTempo(tempo);

Mod.normalise(verse);
Mod.repeat(drums, 4);


verse.add(drums2);
verse.add(drums3);
verse.add(drums4);
verse.add(drums5);
```

```
        verse.add(drums6);
        Mod.repeat(drums2,4);
        Mod.repeat(drums3,4);
        Mod.repeat(drums2,4);


        Write.midi(verse, "finaltest.mid");


    }

    static gridRitmo baserit(String name, String ins1, String
        genre) {
        gridRitmo ritmobase = new gridRitmo(4.00, ins1, genre);
        return ritmobase;
    }

    static ArrayList<notaMusical> base_notes(String name,
        String ins1, gridRitmo ritmobase, Escala e1) {
        Phrase verphr = new Phrase();
        ArrayList<notaMusical> notasbase = new ArrayList<>();
        double sum = 0;
        for (int i = 0; i <= ritmobase.grid.size() - 1; i++) {
            double x = ritmobase.grid.get(i);
            notasbase.add(new notaMusical(x, e1.nE, ins1));
                //creacion de notas para la base
            sum += x;
        }

        double restante = 4.0 - sum;
        if (sum < 4.0000) { //si la parte no esta completa en
            tiempos la completamos con silencio
            notasbase.add(new notaMusical(-2147483648, restante
                , ins1));
        }

        return notasbase;
    }

    static Phrase base_phrase(ArrayList<notaMusical> notasbase)
        {
        Phrase verphr = new Phrase();
        for (notaMusical x : notasbase) {

            verphr.add(x.not);
        }
        return verphr;
    }

    static Phrase verse(String name, String inst, String genre,
            Grafo gx, gridRitmo ritmobase,
            ArrayList<notaMusical> notasbase, Escala e1) {
```

```
        int SILENCE = -2147483648;
        Phrase verphr2 = new Phrase();
        ArrayList<notaMusical> notasacom = new ArrayList<>();
        gridRitmo acompa amiento = new gridRitmo(4.0, inst,
            genre);
        notasacom = gx.arregloCreate(ritmobase, notasbase,
                gx.compatChart, acompa amiento, e1);

        double sumac = 0;
        for (int i = 0; i < notasacom.size(); i++) {
            notasacom.get(i).not.setDuration(acompa amiento.
                grid.get(i));
        }
        for (double x : acompa amiento.grid) {
            sumac += x;
        }
        if (sumac < 4.0000) {
            notasacom.add(new notaMusical(SILENCE, 4.0000 -
                sumac));
        }
        for (notaMusical x : notasacom) {
            //System.out.println(x.not.getNote() + "  " + x.not
                .getDuration());
            verphr2.add(x.not);
        }

        return verphr2;
    }



}
```

- Módulo Grafo

```
import java.util.ArrayList;
import java.util.List;

/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author  MARIN
 */
public class Grafo {

    public int[][] compatChart;
```

```
public Grafo() {
    char[][] cChart2 = new char[8][8];
    int[][] cChart = new int[8][8];    //matriz de
        compatibilidad de notas

    this.compatChart = cChart;

    for (int i = 0; i < 8; i++) {  //llenamos de  0s D:
        for (int j = 0; j < 7; j++) {
            cChart[i][j] = 0;
        }
    }
    for (int j = 0; j < 8; j++) {
        cChart[j][j] = j + 1;   //diagonal de propio grado
            en la escala

    }
    cChart[0][2] = 3;  //grados de la escala compatibles
        con C
    cChart[0][3] = 4;
    cChart[0][4] = 5;
    cChart[0][6] = 7;
    cChart[0][7] = 7;

    cChart[1][0] = 1; // grados de la escala compatibles
        con D
    cChart[1][3] = 4;
    cChart[1][4] = 5;
    cChart[1][5] = 6;
    cChart[1][7] = 1;

    cChart[2][1] = 2; //grados de la escala compatibles con
         E
    cChart[2][4] = 5;
    cChart[2][5] = 6;
    cChart[2][6] = 7;
    cChart[2][7] = 2;

    cChart[3][0] = 1; // grados de la escala compatibles
        con F
    cChart[3][2] = 3;
    cChart[3][5] = 6;
    cChart[3][6] = 7;
    cChart[3][7] = 3;

    cChart[4][0] = 1; // grados de la escala compatibles
        con G
    cChart[4][1] = 2;
    cChart[4][3] = 5;
    cChart[4][6] = 7;
    cChart[4][7] = 4;
```

```
        cChart[5][0] = 1; //grados de la escala compatibles con
            A
        cChart[5][1] = 2;
        cChart[5][2] = 3;
        cChart[5][4] = 6;
        cChart[5][7] = 5;

        cChart[6][1] = 2; //grados de la escala compatibles con
            B
        cChart[6][2] = 3;
        cChart[6][3] = 4;
        cChart[6][5] = 6;
        cChart[6][7] = 6;

        cChart[7][1] = 2; //grados de la escala compatibles con
            B
        cChart[7][2] = 3;
        cChart[7][3] = 4;
        cChart[7][4] = 1;
        cChart[7][5] = 6;
        cChart[7][6] = 1;

    }

    public ArrayList<notaMusical> arregloCreate(gridRitmo base,
        ArrayList<notaMusical> notasbase, int[][] chart,
       gridRitmo acompa amiento, Escala e) {
        ArrayList<notaMusical> n = new ArrayList<>();
        ArrayList<notaMusical> comp = new ArrayList<>();
        double duracionbase = 0;
        double duracionacompa amiento = 0;
        double sumaBase = 0;
        double sumaAcompa amiento = 0;
        int j = 0;
        int i = 0;
        double sumAcompa amiento = 0;
        double sumBase = 0;
        for (double x : base.grid) {
            sumBase += x;

        }

        for (double x : acompa amiento.grid) {
            sumAcompa amiento += x;

        }
        if (sumBase < 4.00000) {
            double restante2 = 4.00000 - sumBase;
            base.grid.add(restante2);

        }

        if (sumAcompa amiento < 4.00000) {
```

```
            double restante = 4.00000 - sumAcompaamiento;
            acompaamiento.grid.add(restante);

        }

        duracionbase = base.grid.get(j);
        duracionacompaamiento = acompaamiento.grid.get(i);

        while (j < base.grid.size() || i < acompaamiento.grid
            .size()) {
            if (duracionacompaamiento > duracionbase) {
                comp.add(notasbase.get(j));
                j++;
                if (j < base.grid.size()) {
                    duracionbase = duracionbase + base.grid.get
                        (j);
                }
            }
            if (duracionacompaamiento == duracionbase) {
                comp.add(notasbase.get(j));
                j++;
                i++;
                if (j < base.grid.size() && i < acompaamiento
                    .grid.size()) {
                    duracionbase = base.grid.get(j);
                    duracionacompaamiento = acompaamiento.
                        grid.get(i);
                }
                //hacer la compatibilidad
                n.add(comp_note_get(comp,e,chart));
                        comp.clear();
            }
            if (duracionacompaamiento < duracionbase) {
                comp.add(notasbase.get(j));
                i++;

                duracionacompaamiento =
                    duracionacompaamiento
                        + acompaamiento.grid.get(i);
                        n.add(comp_note_get(comp,e,chart));
                        comp.clear();
            }

        }

        return n;
    }

    private notaMusical comp_note_get(ArrayList<notaMusical>
        compat, Escala e, int[][] compatChart) {
        ArrayList<Integer> rows = new ArrayList<>();
        for (notaMusical x : compat) {
                                    //guarda el grado en la
```

```
            for (int i = 0; i < e.notas_dentro.size(); i++) {
                 //escala de cada nota
                if (x.not.getNote() == e.notas_dentro.get(i)) {
                    rows.add(i);
                }     //en una lista
            }

        }//ahora debemos buscar en el grafo representado por la
             lista de adyacencia

        ArrayList<Integer> comparador = new ArrayList<>();
        for (int i = 0; i < 7; i++) {

            if (compatChart[rows.get(0)][i] != 0) {

                comparador.add(compatChart[rows.get(0)][i]);
            }
            for (int x : comparador) {
                int encontrado = 0;
                for (int f = 1; f < rows.size(); f++) {
                    for (int c = 0; c < 7; c++) {
                        if (compatChart[f][c] == x) {
                            encontrado++;
                            if(encontrado == rows.size()-1){
                                return new notaMusical(e,
                                    compatChart[f][c]);
                            }
                        }
                    }
                }
            }
        }

        return new notaMusical(-2147483648,1.0);
    }
}
```

- Módulo Percusiones

```
import java.util.ArrayList;
import java.util.Random;
import static jm.constants.Durations.Q;
import jm.music.data.Note;
import static jm.music.data.Note.REST;
import jm.music.data.Part;
import jm.music.data.Phrase;

/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
//clase para inicializar las baterias y todo lo que sea
   percusiones para nuestro programa
/**
 *
 * @author  MARIN
 */
public class Percusiones {

    public Percusiones(){}




    public void createDrums(double init,ArrayList<Part> pop,
       ArrayList<Part> rock, ArrayList<Part> blues, ArrayList<
       Part> ballad) {
        // rock
        Phrase bd0 = new Phrase(init);
        Phrase sd0 = new Phrase(init);
        Phrase ch0 = new Phrase(init);
        Phrase bd1 = new Phrase(init);
        Phrase sd1 = new Phrase(init);
        Phrase ch1 = new Phrase(init);
        Phrase bd2 = new Phrase(init);
        Phrase sd2 = new Phrase(init);
        Phrase ch2 = new Phrase(init);
        Phrase bd3 = new Phrase(init);
        Phrase sd3 = new Phrase(init);
        Phrase ch3 = new Phrase(init);
        Phrase bd4 = new Phrase(init);
        Phrase sd4 = new Phrase(init);
        Phrase ch4 = new Phrase(init);
        Phrase bd5 = new Phrase(init);
        Phrase sd5 = new Phrase(init);
        Phrase ch5 = new Phrase(init);
        Phrase bd6 = new Phrase(init);
        Phrase sd6 = new Phrase(init);
        Phrase ch6 = new Phrase(init);
        Phrase bd7 = new Phrase(init);
        Phrase sd7 = new Phrase(init);
        Phrase ch7 = new Phrase(init);
        //pop phrases
        Phrase pbd0 = new Phrase(init);
        Phrase psd0 = new Phrase(init);
        Phrase pch0 = new Phrase(init);
        Phrase pbd1 = new Phrase(init);
        Phrase psd1 = new Phrase(init);
        Phrase pch1 = new Phrase(init);
        Phrase pbd2 = new Phrase(init);
        Phrase psd2 = new Phrase(init);
        Phrase pch2 = new Phrase(init);
```

```
Phrase pbd3 = new Phrase(init);
Phrase psd3 = new Phrase(init);
Phrase pch3 = new Phrase(init);
//balad phrases
Phrase babd0 = new Phrase(init);
Phrase basd0 = new Phrase(init);
Phrase bach0 = new Phrase(init);
Phrase babd1 = new Phrase(init);
Phrase basd1 = new Phrase(init);
Phrase bach1 = new Phrase(init);
Phrase babd2 = new Phrase(init);
Phrase basd2 = new Phrase(init);
Phrase bach2 = new Phrase(init);
Phrase babd3 = new Phrase(init);
Phrase basd3 = new Phrase(init);
Phrase bach3 = new Phrase(init);
//blues phrases
Phrase blbd0 = new Phrase(init);
Phrase blsd0 = new Phrase(init);
Phrase blch0 = new Phrase(init);
Phrase blbd1 = new Phrase(init);
Phrase blsd1 = new Phrase(init);
Phrase blch1 = new Phrase(init);
Phrase blbd2 = new Phrase(init);
Phrase blsd2 = new Phrase(init);
Phrase blch2 = new Phrase(init);
Phrase blbd3 = new Phrase(init);
Phrase blsd3 = new Phrase(init);
Phrase blch3 = new Phrase(init);
//rock 1st drums
for (int r = 0; r < 8; r++) {
                                    //frases para partes
                                       rock
    //frases para partes Pop
    //frases para partes blues
    //frases para partes balada

}
for (int r = 0; r < 4; r++) {
    Note note = new Note(42, Q); //hithats en octavos

    ch0.add(note);          //frases para partes Rock
    ch0.add(note);

                                //frases para partes Pop
    //frases para partes blues
    //frases para partes balada
}

for (int r = 0; r < 2; r++) {
    Note rest2 = new Note(REST, Q); //descanso en
        cuartos
    Note note = new Note(36, Q);    //bass drum cuartos
```

```
Note notesnare = new Note(38, Q); //hithats cuartos
bd0.add(note);          //frases para parte 1  ROCK
bd0.add(rest2);
bd0.add(rest2);
bd0.add(rest2);

sd0.add(rest2);
sd0.add(rest2);
sd0.add(notesnare);
sd0.add(rest2);
//frases Rock 2
bd1.add(note); //bass drum score
bd1.add(note);
bd1.add(rest2);
bd1.add(rest2);

sd1.add(rest2); //snare drum score
sd1.add(rest2);
sd1.add(notesnare);
sd1.add(rest2);
//frase rock 3
bd2.add(note); //bass drum score
bd2.add(note);
bd2.add(rest2);
bd2.add(note);

sd2.add(rest2); //snare drum score
sd2.add(rest2);
sd2.add(notesnare);
sd2.add(rest2);
//frase rock 4
bd3.add(note); //bass drum score
bd3.add(rest2);
bd3.add(note);
bd3.add(rest2);

sd3.add(rest2); //snare drum score
sd3.add(rest2);
sd3.add(notesnare);
sd3.add(rest2);
// frase rock 5
sd4.add(rest2);  //snare drum score
sd4.add(rest2);
sd4.add(notesnare);
sd4.add(rest2);
//frase rock 6
sd5.add(rest2);
sd5.add(rest2);
sd5.add(notesnare);
sd5.add(rest2);
//frase rock 7
sd6.add(rest2);
sd6.add(rest2);
```

```
        sd6.add(notesnare);
        sd6.add(rest2);
        //frase rock 8
        sd7.add(rest2);
        sd7.add(rest2);
        sd7.add(notesnare);
        sd7.add(rest2);

        //frases para partes Pop
        //frases para partes blues
        //frases para partes balada
    }
    Note rest2 = new Note(REST, Q); //descanso en octavos
    Note note = new Note(36, Q);    //bass drum octavos
    bd4.add(note);       //bass drum frase rock 5
    bd4.add(rest2);
    bd4.add(rest2);
    bd4.add(note);
    bd4.add(rest2);
    bd4.add(rest2);
    bd4.add(rest2);
    bd4.add(rest2);
    bd4.add(rest2);

    bd5.add(rest2); //bass drum frase rock 6
    bd5.add(rest2);
    bd5.add(rest2);
    bd5.add(rest2);

    bd5.add(note); //bass drum frase rock 6
    bd5.add(note);
    bd5.add(rest2);
    bd5.add(note);
    //bass drum frase rock 7
    bd6.add(note);
    bd6.add(rest2);
    bd6.add(rest2);
    bd6.add(note);
    bd6.add(note);
    bd6.add(rest2);
    bd6.add(rest2);
    bd6.add(rest2);

    //bassdrum frase rock 8
    bd7.add(note);
    bd7.add(note);
    bd7.add(rest2);
    bd7.add(note);
    bd7.add(note);
    bd7.add(rest2);
    bd7.add(rest2);
    bd7.add(rest2);
    //Partes Rock  ->
```

```
Part r0 = new Part("r0",0,9);
r0.add(ch0);
r0.add(bd0);
r0.add(sd0);
rock.add(r0);
Part r1 = new Part("r1",0,9);
r1.add(ch0);
r1.add(bd1);
r1.add(sd1);
rock.add(r1);
Part r2 = new Part("r2",0,9);
r2.add(bd2);
r2.add(sd2);
r2.add(ch0);
rock.add(r2);
Part r3 = new Part("r3",0,9);
r3.add(ch0);
r3.add(sd3);
r3.add(bd3);
rock.add(r3);
Part r4 = new Part("r4",0,9);
r4.add(ch0);
r4.add(bd4);
r4.add(sd4);
rock.add(r4);
Part r5 = new Part("r5",0,9);
r5.add(ch0);
r5.add(bd5);
r5.add(sd5);
rock.add(r5);
Part r6 = new Part("r6",0,9);
r6.add(ch0);
r6.add(bd6);
r6.add(sd6);
rock.add(r6);
Part r7 = new Part("r7",0,9);
r7.add(ch0);
r7.add(bd7);
r7.add(sd7);
rock.add(r7);
//pop beat 0
Note openhat = new Note(46,Q); //open hithat octavos
Note closhat = new Note(42,Q); //closed hithat octavos
Note snarenote = new Note(38,Q); //snare drum octavos
Note bdnote = new Note(36,Q); // bass drum octavos

pch0.add(openhat);
pch0.add(closhat);
pch0.add(closhat);
pch0.add(closhat);
pch0.add(closhat);
pch0.add(closhat);
pch0.add(closhat);
```

```
        pch0.add(closhat);

        psd0.add(rest2);
        psd0.add(rest2);
        psd0.add(snarenote);
        psd0.add(snarenote);
        psd0.add(rest2);
        psd0.add(rest2);
        psd0.add(snarenote);
        psd0.add(rest2);

        pbd0.add(bdnote);
        pbd0.add(rest2);
        pbd0.add(rest2);
        pbd0.add(rest2);
        pbd0.add(rest2);
        pbd0.add(rest2);
        pbd0.add(rest2);
        pbd0.add(rest2);
        //pop beat 1
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);
        pch1.add(closhat);

        pbd1.add(bdnote);
        pbd1.add(rest2);
        pbd1.add(rest2);
        pbd1.add(rest2);
        pbd1.add(bdnote);
        pbd1.add(rest2);
        pbd1.add(rest2);
        pbd1.add(rest2);

        psd1.add(rest2);
        psd1.add(rest2);
        psd1.add(snarenote);
        psd1.add(rest2);
        psd1.add(rest2);
        psd1.add(rest2);
        psd1.add(snarenote);
        psd1.add(rest2);
        //pop part2
        pch2.add(closhat);
        pch2.add(closhat);
        pch2.add(closhat);
        pch2.add(closhat);
        pch2.add(closhat);
        pch2.add(closhat);
```

```
        pch2.add(closhat);
        pch2.add(closhat);

        pbd2.add(bdnote);
        pbd2.add(rest2);
        pbd2.add(rest2);
        pbd2.add(rest2);
        pbd2.add(bdnote);
        pbd2.add(rest2);
        pbd2.add(rest2);
        pbd2.add(bdnote);

        psd2.add(rest2);
        psd2.add(rest2);
        psd2.add(snarenote);
        psd2.add(rest2);
        psd2.add(rest2);
        psd2.add(snarenote);
        psd2.add(snarenote);
        psd2.add(rest2);

        //pop part3
        pch3.add(openhat);
        pch3.add(closhat);
        pch3.add(closhat);
        pch3.add(closhat);
        pch3.add(closhat);
        pch3.add(closhat);
        pch3.add(closhat);
        pch3.add(closhat);

        pbd3.add(bdnote);
        pbd3.add(rest2);
        pbd3.add(rest2);
        pbd3.add(bdnote);
        pbd3.add(rest2);
        pbd3.add(rest2);
        pbd3.add(rest2);
        pbd3.add(rest2);

        psd3.add(rest2);
        psd3.add(rest2);
        psd3.add(snarenote);
        psd3.add(rest2);
        psd3.add(rest2);
        psd3.add(rest2);
        psd3.add(snarenote);
        psd3.add(rest2);

      //Partes pop ->
        Part p0 = new Part("p0",0,9);
        p0.add(pch0);
        p0.add(psd0);
```

```
p0.add(pbd0);
pop.add(p0);
Part p1 = new Part("p1",0,9);
p1.add(pch1);
p1.add(psd1);
p1.add(pbd1);
pop.add(p1);
Part p2 = new Part("p2",0,9);
p2.add(pch2);
p2.add(psd2);
p2.add(pbd2);
pop.add(p2);
Part p3 = new Part("p3",0,9);
p3.add(pch3);
p3.add(psd3);
p3.add(pbd3);
pop.add(p3);
//blues phrases 0 - 4
blbd0.add(bdnote);
blbd0.add(rest2);
blbd0.add(rest2);
blbd0.add(rest2);
blbd0.add(bdnote);
blbd0.add(rest2);
blbd0.add(rest2);
blbd0.add(rest2);


blsd0.add(rest2);
blsd0.add(rest2);
blsd0.add(snarenote);
blsd0.add(rest2);
blsd0.add(rest2);
blsd0.add(rest2);
blsd0.add(snarenote);
blsd0.add(rest2);

blch0.add(openhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);
blch0.add(closhat);

blbd1.add(bdnote);
blbd1.add(rest2);
blbd1.add(rest2);
blbd1.add(rest2);
blbd1.add(bdnote);
blbd1.add(rest2);
```

```
        blbd1.add(rest2);
        blbd1.add(rest2);


        blsd1.add(rest2);
        blsd1.add(rest2);
        blsd1.add(snarenote);
        blsd1.add(rest2);
        blsd1.add(rest2);
        blsd1.add(rest2);
        blsd1.add(snarenote);
        blsd1.add(rest2);

        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);
        blch1.add(closhat);

        blbd2.add(bdnote);
        blbd2.add(rest2);
        blbd2.add(rest2);
        blbd2.add(rest2);
        blbd2.add(bdnote);
        blbd2.add(rest2);
        blbd2.add(rest2);
        blbd2.add(rest2);


        blsd2.add(rest2);
        blsd2.add(rest2);
        blsd2.add(snarenote);
        blsd2.add(rest2);
        blsd2.add(rest2);
        blsd2.add(rest2);
        blsd2.add(snarenote);
        blsd2.add(rest2);

        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
        blch2.add(closhat);
```

```
blbd3.add(bdnote);
blbd3.add(rest2);
blbd3.add(rest2);
blbd3.add(bdnote);
blbd3.add(bdnote);
blbd3.add(rest2);
blbd3.add(rest2);
blbd3.add(rest2);


blsd3.add(rest2);
blsd3.add(rest2);
blsd3.add(snarenote);
blsd3.add(rest2);
blsd3.add(rest2);
blsd3.add(rest2);
blsd3.add(snarenote);
blsd3.add(rest2);

blch3.add(openhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);
blch3.add(closhat);


//Partes blues ->
Part bl0 = new Part("bl0",0,9);
bl0.add(blbd0);
bl0.add(blsd0);
bl0.add(blch0);
blues.add(bl0);

Part bl1 = new Part("bl1",0,9);
bl1.add(blbd1);
bl1.add(blsd1);
bl1.add(blch1);
blues.add(bl1);
Part bl2 = new Part("bl2",0,9);
bl2.add(blsd2);
bl2.add(blbd2);
bl2.add(blbd3);
blues.add(bl2);
Part bl3 = new Part("bl3",0,9);
bl3.add(blbd3);
bl3.add(blsd3);
bl3.add(blch3);
blues.add(bl3);
```

```
//phrase ballad 0
bach0.add(rest2);
bach0.add(closhat);
bach0.add(closhat);
bach0.add(rest2);
bach0.add(rest2);
bach0.add(closhat);
bach0.add(closhat);
bach0.add(rest2);

babd0.add(bdnote);
babd0.add(bdnote);
babd0.add(rest2);
babd0.add(rest2);
babd0.add(bdnote);
babd0.add(bdnote);
babd0.add(rest2);
babd0.add(rest2);

basd0.add(rest2);
basd0.add(rest2);
basd0.add(snarenote);
basd0.add(rest2);
basd0.add(rest2);
basd0.add(rest2);
basd0.add(snarenote);
basd0.add(rest2);

//phrase ballad 1
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);
bach1.add(closhat);

babd1.add(bdnote);
babd1.add(rest2);
babd1.add(rest2);
babd1.add(rest2);
babd1.add(rest2);
babd1.add(rest2);
babd1.add(rest2);
babd1.add(rest2);

basd1.add(rest2);
basd1.add(rest2);
basd1.add(rest2);
basd1.add(rest2);
basd1.add(snarenote);
```

```
        basd1.add(rest2);
        basd1.add(rest2);
        basd1.add(rest2);

        //phrase ballad2
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);
        bach2.add(closhat);

        babd2.add(bdnote);
        babd2.add(rest2);
        babd2.add(bdnote);
        babd2.add(rest2);
        babd2.add(rest2);
        babd2.add(rest2);
        babd2.add(rest2);
        babd2.add(rest2);

        basd2.add(rest2);
        basd2.add(snarenote);
        basd2.add(rest2);
        basd2.add(snarenote);
        basd2.add(rest2);
        basd2.add(rest2);
        basd2.add(snarenote);
        basd2.add(rest2);

        //ballad 3
        bach3.add(closhat);
        bach3.add(closhat);
        bach3.add(closhat);
        bach3.add(closhat);
        bach3.add(rest2);
        bach3.add(rest2);
        bach3.add(rest2);
        bach3.add(rest2);

        babd3.add(bdnote);
        babd3.add(rest2);
        babd3.add(rest2);
        babd3.add(rest2);
        babd3.add(rest2);
        babd3.add(rest2);
        babd3.add(rest2);
        babd3.add(rest2);

        basd3.add(rest2);
        basd3.add(rest2);
```

```
        basd3.add(snarenote);
        basd3.add(snarenote);
        basd3.add(rest2);
        basd3.add(rest2);
        basd3.add(snarenote);
        basd3.add(snarenote);

        //Partes balada ->
        Part ba0 = new Part("ba0",0,9);
        ba0.add(basd0);
        ba0.add(bach0);
        ba0.add(babd0);
        ballad.add(ba0);
        Part ba1 = new Part("ba1",0,9);
        ba1.add(basd1);
        ba1.add(bach1);
        ba1.add(babd1);
        ballad.add(ba1);
        Part ba2 = new Part("ba2",0,9);
        ba2.add(basd2);
        ba2.add(bach2);
        ba2.add(babd2);
        ballad.add(ba2);
        Part ba3 = new Part("ba3",0,9);
        ba3.add(basd3);
        ba3.add(bach3);
        ba3.add(babd3);
        ballad.add(ba3);









    }

    public Part getRdrums(String genre, ArrayList<Part> pop,
        ArrayList<Part> rock, ArrayList<Part> blues, ArrayList<
        Part> ballad) {
        Part r = new Part();
        Random x = new Random();
        if(genre.equals("Rock")){r = rock.get(x.nextInt(rock.
            size()));}
        else if(genre.equals("Pop")){r = pop.get(x.nextInt(pop.
            size()));}
        else if(genre.equals("Ballad")){r = ballad.get(x.
            nextInt(ballad.size()));}
        else if(genre.equals("Blues")){r = blues.get(x.nextInt(
            blues.size()));}

        return r;
```

```
    }

}
```

- Módulo gridRitmo

```
import java.util.ArrayList;
import java.util.Random;
import static jm.constants.Durations.*;
import jm.music.data.Note;
import static jm.music.data.Note.REST;
import jm.music.data.Part;
import jm.music.data.Phrase;

/**
 *
 * @author  MARIN
 */
public class gridRitmo {

    public ArrayList<Double> grid;


    public ArrayList getGrid() {
        return grid;
    }

    public gridRitmo() {
    }

    public gridRitmo(double timScore, String inst, String gen)
        {
        this.grid = new ArrayList<Double>();
        ArrayList<Double> sampleRythm = new ArrayList<Double>()
            ;

        Random x = new Random();
        double sum = 0;
        int carry;
        if ("Bass".equals(inst) && "Rock" == gen) {
            sampleRythm.add(WN);     //a adimos al arreglo los
                valores de ritmo para este geero
            sampleRythm.add(M);
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);
            sampleRythm.add(TSN);
        }
        if ("Piano".equals(inst) && "Rock" == gen) {
```

```
        sampleRythm.add(WN); //a adimos al arreglo los
            valores de ritmo para este geero
        sampleRythm.add(WN);
        sampleRythm.add(WN);
        sampleRythm.add(WN);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(MT);
        sampleRythm.add(QN);
        sampleRythm.add(QN);
        sampleRythm.add(QN);
        sampleRythm.add(QN);

        sampleRythm.add(DQN);
        sampleRythm.add(EN);
        sampleRythm.add(SN);
        sampleRythm.add(TSN);
    }

    if ("Guitar".equals(inst) && "Rock".equals(gen)) {

        sampleRythm.add(WN); //a adimos al arreglo los
            valores de ritmo para este geero
        sampleRythm.add(WN);
        sampleRythm.add(WN);
        sampleRythm.add(WN);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(M);
        sampleRythm.add(MT);
        sampleRythm.add(QN);
        sampleRythm.add(QN);
        sampleRythm.add(QN);
        sampleRythm.add(QN);

        sampleRythm.add(DQN);
        sampleRythm.add(EN);
        sampleRythm.add(SN);
        sampleRythm.add(TSN);
    }

    if ("Bass".equals(inst) && "Pop" == gen) {
            //a adimos al arreglo los valores de ritmo
                para este genero


        sampleRythm.add(QN);
```

```
            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);

        }
        if ("Piano".equals(inst) && "Pop" == gen) {

            //a adimos al arreglo los valores de ritmo para
                este geero
            sampleRythm.add(WN);


            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);

        }

        if ("Guitar".equals(inst) && "Pop".equals(gen)) {
        } else {
            //a adimos al arreglo los valores de ritmo para
                este geero
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);

            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);
            sampleRythm.add(TSN);
        }
        if ("Bass".equals(inst) && "Ballad".equals(gen)) {
            sampleRythm.add(WN);     //a adimos al arreglo los
                valores de ritmo para este genero
            sampleRythm.add(M);

            sampleRythm.add(QN);
             sampleRythm.add(QN);
              sampleRythm.add(QN);

        }
```

```
        if ("Piano".equals(inst) && "Ballad".equals(gen)) {

            sampleRythm.add(WN);
            sampleRythm.add(M);
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);

        }

        if ("Guitar".equals(inst) && "Ballad" == gen) {

            sampleRythm.add(WN); //a adimos al arreglo los
                valores de ritmo para este geero
            sampleRythm.add(WN);
            sampleRythm.add(WN);
            sampleRythm.add(WN);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);

            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);
            sampleRythm.add(TSN);
        }
        if ("Bass".equals(inst) && "Blues".equals(gen)) {

            sampleRythm.add(QN);
            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);

        }
        if ("Piano".equals(inst) && "Blues".equals(gen)) {

            sampleRythm.add(WN); //a adimos al arreglo los
                valores de ritmo para este geero
            sampleRythm.add(M);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(DQN);
```

```
            sampleRythm.add(SN);
        }

        if ("Guitar".equals(inst) && "Blues".equals(gen)) {

            sampleRythm.add(WN); //a adimos al arreglo los
                valores de ritmo para este geero
            sampleRythm.add(WN);
            sampleRythm.add(WN);
            sampleRythm.add(WN);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(M);
            sampleRythm.add(MT);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(QN);
            sampleRythm.add(DQN);
            sampleRythm.add(EN);
            sampleRythm.add(SN);
            sampleRythm.add(TSN);
        }



        while (sum <= timScore) {

            carry = x.nextInt(sampleRythm.size() - 1);
            double m = sampleRythm.get(carry);
            grid.add(m);
            sum += sampleRythm.get(carry);
            // System.out.println("suma   " + sum);

        }
        while (sum > timScore) {
            sum -= grid.get(grid.size() - 1);
            grid.remove(grid.size() - 1);
        }
    }


}
```

- Módulo Escala

```
import java.util.ArrayList;
import static jm.constants.Durations.QUARTER_NOTE;
import static jm.constants.Pitches.C4;
```

```
import jm.music.data.Note;

/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author  MARIN
 */
public class Escala {
    //terminado con todas las escalas
    //trabajo futuro: todos inicien en la nota mas grave
        posible de la escala

    ArrayList nE = new ArrayList<Integer>();
    ArrayList notas_dentro = new ArrayList<String>();
    String Es = "";

    public Escala(String Tonalidad) {
        ArrayList nx = new ArrayList<Integer>();
        this.nE = nx;
        int jump = 0;

        int i;

        switch (Tonalidad) { //hay que a adir silencios
            constnte negativa
            case "C":
                this.Es = "C";
                for (i = 28; i < 38; i++) {
                    jump++;
                    if (jump > 2 & jump < 6) {
                        i++;
                    }
                    if (jump == 7) {
                        i++;
                        jump = 0;
                    }
                    nE.add(i);

                }
                for (i = 40; i < 50; i++) {
                    jump++;
                    if (jump > 2 & jump < 6) {
                        i++;
                    }
                    if (jump == 7) {
                        i++;
                        jump = 0;
                    }
```

```
                nE.add(i);

            }
            for (i = 52; i < 62; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 64; i < 74; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 76; i < 86; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "C#":
            this.Es = "C#";
            for (i = 29; i < 39; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
```

```
            for (i = 41; i < 51; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 53; i < 63; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 65; i < 75; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 77; i < 87; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "D":
            this.Es = "D";
            for (i = 30; i < 40; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
```

```
            i++;
        }
        if (jump == 7) {
            i++;
            jump = 0;
        }
        nE.add(i);

    }
    for (i = 42; i < 52; i++) {
        jump++;
        if (jump > 2 & jump < 6) {
            i++;
        }
        if (jump == 7) {
            i++;
            jump = 0;
        }
        nE.add(i);

    }
    for (i = 54; i < 64; i++) {
        jump++;
        if (jump > 2 & jump < 6) {
            i++;
        }
        if (jump == 7) {
            i++;
            jump = 0;
        }
        nE.add(i);

    }
    for (i = 66; i < 76; i++) {
        jump++;
        if (jump > 2 & jump < 6) {
            i++;
        }
        if (jump == 7) {
            i++;
            jump = 0;
        }
        nE.add(i);

    }
    for (i = 78; i < 88; i++) {
        jump++;
        if (jump > 2 & jump < 6) {
            i++;
        }
        if (jump == 7) {
            i++;
            jump = 0;
```

```
                }
                nE.add(i);

            }

        case "D#":
            this.Es = "D#";
            for (i = 31; i < 41; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 43; i < 53; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 55; i < 65; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 67; i < 77; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);
```

```
        }
        for (i = 79; i < 89; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
    case "E":
        this.Es = "E";
        for (i = 32; i < 42; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 44; i < 54; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 56; i < 66; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 68; i < 78; i++) {
```

```
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 80; i < 90; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "F":
            this.Es = "F";
            for (i = 33; i < 43; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 45; i < 55; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 57; i < 67; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
```

```
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 69; i < 79; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 81; i < 91; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "F#":
            this.Es = "F#";
            for (i = 34; i < 44; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 46; i < 56; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
```

```
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 58; i < 68; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 70; i < 80; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 82; i < 92; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
    case "G":
        this.Es = "G";
        for (i = 35; i < 45; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);
```

```
        }
        for (i = 47; i < 57; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 59; i < 69; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 71; i < 81; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 83; i < 93; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
    case "G#":
        this.Es = "G#";
        for (i = 36; i < 46; i++) {
```

```
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 48; i < 58; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 60; i < 70; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 72; i < 82; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 84; i < 94; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
```

```
                i++;
                jump = 0;
            }
            nE.add(i);

        }
    case "A":
        this.Es = "A";
        for (i = 37; i < 47; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 49; i < 59; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 61; i < 71; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
            nE.add(i);

        }
        for (i = 73; i < 83; i++) {
            jump++;
            if (jump > 2 & jump < 6) {
                i++;
            }
            if (jump == 7) {
                i++;
                jump = 0;
            }
```

```
                nE.add(i);

            }
            for (i = 85; i < 95; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "A#":
            this.Es = "A#";
            for (i = 38; i < 49; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 50; i < 60; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 62; i < 72; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
```

```
            for (i = 74; i < 84; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 86; i < 96; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
        case "B":
            this.Es = "B";
            for (i = 39; i < 49; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 51; i < 61; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
                    i++;
                }
                if (jump == 7) {
                    i++;
                    jump = 0;
                }
                nE.add(i);

            }
            for (i = 63; i < 73; i++) {
                jump++;
                if (jump > 2 & jump < 6) {
```

```
                        i++;
                    }
                    if (jump == 7) {
                        i++;
                        jump = 0;
                    }
                    nE.add(i);

                }
                for (i = 75; i < 85; i++) {
                    jump++;
                    if (jump > 2 & jump < 6) {
                        i++;
                    }
                    if (jump == 7) {
                        i++;
                        jump = 0;
                    }
                    nE.add(i);

                }
                for (i = 87; i < 97; i++) {
                    jump++;
                    if (jump > 2 & jump < 6) {
                        i++;
                    }
                    if (jump == 7) {
                        i++;
                        jump = 0;
                    }
                    nE.add(i);

                }
            }
            for (int j = 5; j < 12; j++) {

                notas_dentro.add(
                        new Note((int) nE.get(j), QUARTER_NOTE)
                        .getNote());


            }
            notas_dentro.add("N/A");

        }
}
```

- Módulo NotaMusical

```
import java.util.ArrayList;
import java.util.Random;
```

```
import jm.music.data.*;
import jm.JMC;
import jm.audio.*;
import static jm.constants.Durations.QUARTER_NOTE;
import static jm.constants.Pitches.C4;
import jm.util.*;

/**
 *
 * @author  MARIN
 */
public class notaMusical {

    int SILENCE = -2147483648;
    Note not;

    public notaMusical(int note, Double Duracion, String inst)
        {
        Note silence = new Note(note, Duracion);
        this.not = silence;

    }

    public notaMusical(Double Duracion, ArrayList<Integer>
        Escala, String inst) {
        //a adir genero para restricciones de probabilidad de
            aparicion de notas

        Random x = new Random();
        int des = x.nextInt(27);
        if (inst == "Guitar") {
            des = des + 12;
        } //aumenta una octava a cualquier cosa que toque la
            guitarra
        Note d = new Note(C4, QUARTER_NOTE);  //pitches del 0
            al 127  y duraciones de .08333333333333 hasta 4.0
        if (x.nextInt(113) %7 == 0) {
            Note n = new Note(-2147483648, Duracion);
            this.not = n;
        }//silencio
        else {
            Note n = new Note(Escala.get(des), Duracion);   //
                obtiene una nota aleatoria segun una
                distribucion normal
            this.not = n;
        }

    }

    public notaMusical(int note, Double duracion) {

        Note n = new Note(note, duracion);
```

```
        this.not = n;

    }

    public notaMusical(double silence) {
        this.not = new Note(silence, 1);
    }

    public notaMusical(Escala e, int grado) {
        ArrayList<Integer> octavas_nota = new ArrayList<>();
        if (grado > 6) {
            this.not = new Note(SILENCE, 1.0);
        } else {
            for (int i = 5; i < 35; i++) {
                octavas_nota.add(i + grado+24);
                i += 12;

            }
            Random x = new Random();
            Note s = new Note(
                    octavas_nota.get(x.nextInt(octavas_nota.
                        size() - 1)), 1.0);
            this.not = s;
        }
    }
}
```

- **Diagrama Uml**

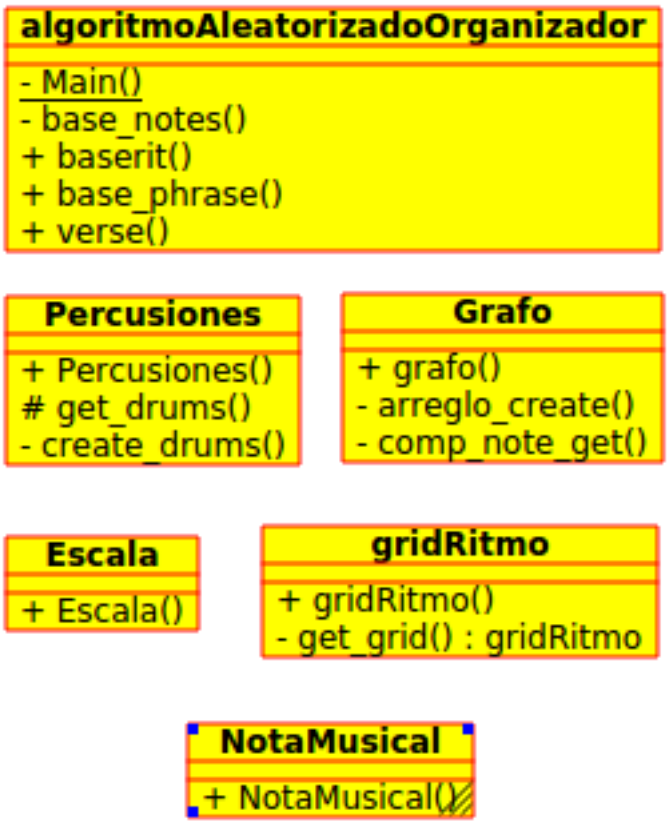


Figura 1: Diagrama UML Backing Track Generator.