UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
INSTITUTO DE COMPUTAÇÃO

Gustavo Lêdo Silva

DESENVOLVENDO UM SOFTWARE PARA ESCRITA E LEITURA DE CIFRAS MUSICAIS

Niterói
2010

Gustavo Lêdo Silva

DESENVOLVENDO UM SOFTWARE PARA ESCRITA E LEITURA DE CIFRAS MUSICAIS

Trabalho de Conclusão apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

ORIENTADOR: Anselmo Antunes Montenegro

NITERÓI

2010

Dedico este trabalho, a minha esposa, Luciana e minha Filha, Ana Luiza. Pessoas fundamentais não só para a construção deste trabalho, mas para a minha vida.

Agradecimentos

A todos os professores que contribuíram para a minha formação profissional e em especial ao meu orientador o professor Dr. Anselmo Antunes Montenegro o qual me fez acreditar que este projeto era realmente possível.

A todos os companheiros de trabalho no ADDLabs, os quais foram fundamentais na minha formação profissional.

Ao Felipe Figura, amigo sempre presente durante esta jornada.

"Se eu tenho uma maçã e você tem uma maçã, podemos trocá-las, e você continuará tendo uma maçã e eu continuarei tendo uma maçã. Porém, se eu tiver uma idéia e você tiver uma idéia, podemos trocá-las, e cada um de nós terá duas idéias."

George Bernard Shaw

RESUMO

As cifras musicais representam um sistema de escrita musical utilizado em todo mundo por músicos e profissionais ligados a música, sendo um sistema de escrita eficaz, porém sem muitas ferramentas computacionais relacionadas a ele. A notação musical é uma forma de representar graficamente uma peça musical. Na música ocidental as formas de escrita musical mais usadas são: pentagrama, tablatura e cifra. Cifras musicais representam, através de símbolos gráficos e letras, os acordes que devem ser tocados por um instrumento musical. Elas são muito utilizadas para delinear a harmonia de uma música, podendo ser usadas isoladamente ou como auxílio a outras formas de escrita musical citadas acima. Apesar de todas as facilidades que as cifras proporcionam, atualmente não há no mercado de softwares ferramentas computacionais voltadas para ajudar no momento de escrever e ler especificamente uma cifra musical. Esse trabalho tem como objetivo criar um software para edição e leitura de cifras musicais, proporcionando um ambiente simples para a criação e edição e auxiliando o músico na execução de cifras e linhas harmônicas. Além disso, com o uso somente da escala cromática desenvolver uma nova forma de representação dos acordes musicais, criando um software leve e portável.

Palavras chave: Computação musical, cifras musicais, escrita musical, Diagramas de acorde, acordes.

Sumário

1.INTRODUÇÃO 11

1.1 Panorama histórico 11

1.2 Motivação 12

1.3 Objetivos 13

2. FUNDAMENTOS DA NOTAÇÃO MUSICAL 16

2.1 Cifras Musicais 16

2.2 Acordes Musicais 18

3. O PROBLEMA DE REPRESENTAÇÃO DE ACORDES 19

4. METODOLOGIA UTILIZADA PARA CONSTRUÇÃO E REPRESENTAÇÃO DOS ACORDES MUSICAIS 22

4.1 Reconhecendo acordes 22

4.2 Mapeamento 27

4.3 Montando os acordes 29

5. IMPLEMENTAÇÂO DO SISTEMA 32

5.1 Classes da camada de interface 32

5.2 Classes da camada de controle 32

5.2.1 Classes para execução sonora 33

5.2.2 Classes para desenho do diagrama de acordes 33

5.3 Classes da camada de modelo 33

5.4 Reprodução sonora 33

6.SOFTWARE 35

6.1 Interface do sistema 36

6.2.Funcionalidades 40

6.2.1 Diagramas 40

6.2.2 Reproduzindo o acorde 43

6.3 Determinando as notas a serem tocadas 44

7. CONCLUSÃO 46

REFERÊNCIAS 47

APÊNDICE A – DIAGRAMAS UML 49

Diagrama de casos de uso 49

Diagrama de Classes 50

APÊNDICE B – CÓDIGO FONTE 56

Índice de Figuras:

Figura 1: Árvore do acorde tríade 19
Figura 2: Árvore do acorde tétrade..... 20
Figura 3: Guitarra com 27 trastes 21
Figura 4: Autômato de validação de Cifras..... 23
Figura 5: Representação de uma tríade, com o mapeamento das possibilidades de variações entra as notas fundamentais. 25
Figura 7: tela inicial do software..... 36
Figura 8: Usuário seleciona um arquivo salvo anteriormente 37
Figura 9: janela para o usuário digitar a cifra desejada. 38
Figura 10: visualizando o diagrama do acorde de Ré maior. 38
Figura 11: Software com uma cifra já inserida..... 39
Figura 12: Exibição da janela "Sobre" 39
Figura 13: Violão 40
Figura 14: Diagrama de Acorde..... 41
Figura 15: representação do acorde dó maior exibido na figura 10, com a imagem do braço do violão. 41
Figura 16: Acorde de Lá maior..... 42
Figura 17: Acorde de Lá menor com sétima maior 42

Lista de abreviaturas e siglas:

COSM *- Composite Object Sound Modeling*
IPS - Instrumental Performance System
MIDI - Musical Instrument Digital Interface
MPEG - Moving Picture Experts Group
Mp3 - MPEG-1 audio layer 3

1.INTRODUÇÃO

1.1 Panorama histórico

Com o surgimento do transistor que foi criado nos Laboratórios da Bell Telephone por Bardeen e Brattain em 1948 (idPT, 2009a), houve um “boom” tecnológico que se estendeu por diversas áreas, inclusive a área musical, resultando em uma evolução de equipamentos musicais. Os amplificadores, anteriormente somente valvulados, passaram pouco a pouco a ser substituídos pelos transistorizados, deixando de usar válvulas. Além das vantagens do uso de transistores em relação às válvulas, demonstradas em 1958, por Jack Kilby da Texas Instruments (idPT, 2009b), os amplificadores passaram a ter menor consumo de energia, maior durabilidade, menor tamanho e custo. Porém, na década de 80 surgiu um movimento contrário aos transistores, pois apesar do avanço tecnológico e da maior durabilidade dos equipamentos, isto não se refletia no avanço de qualidade sonora do equipamento e não somente os músicos mais exigentes e conservadores, mas toda a comunidade musical contestava o uso dessas novas tecnologias devido ao decaimento da qualidade sonora. O som gerado pelos transistores era considerado artificial. Apesar disso, devido ao custo mais baixo, equipamentos transistorizados entraram fortemente no mercado e ficaram marcados por terem um custo baixo e uma qualidade sonora ruim ou pelo menos inferior aos valvulados.

Com isto, a idéia de novas tecnologias no meio musical ficou acompanhada do sentimento de um equipamento com menor custo, mas com o som pasteurizado e de baixa qualidade sonora. O que não era somente uma fama ruim, de fato era o que ocorria para a maioria dos músicos.

No final da década de 90, com os crescentes avanços tecnológicos e o desenvolvimento e pesquisa de algoritmos de simulação e processamento de sinais como, por exemplo, a tecnologia *COSM (Composite Object Sound Modeling)* criada pela *Boss Roland*

(*Boss,* 2009) uma grande empresa de produtos musicais que atualmente produz equipamentos digitais de alta qualidade e aceitação no mercado musical, os equipamentos altamente tecnológicos deixam de ser associados a equipamentos de baixa qualidade e passam a ser usados por músicos profissionais.

Atualmente, usando um computador com hardware adequado é possível fazer gravações e produções musicais de alta qualidade. Hoje em dia temos no mercado diversos produtos como, processadores de áudio digitais, gravadores digitais, simuladores entre outros. Podemos citar como exemplo o guitarrista *Joe Satriani* (Satriani, 2010), indicado 15 vezes ao Grammy de musica instrumental. Conhecido por ser um músico extremamente exigente, no intuito de buscar uma nova possibilidade sonora e comprovar este novo conceito entre a união da música e a tecnologia, gravou o álbum *Engines of Creation* apenas com o uso de sua guitarra e o computador. Isto ilustra a capacidade de processamento de áudio, e as possibilidades musicais proporcionadas por um computador com o hardware e software adequados na atualidade.

Nesta mudança de relação entre a música e a tecnologia e o novo olhar dos músicos para esta associação, surge um crescente mercado de softwares musicais. Alguns já considerados como consagrados devido ao seu sucesso e aprovação no mercado como, por exemplo, o *ProTools(Avid, 2010)* usado em grandes estúdios para gravações e produção musical, *Sound Forge (*Sound Forge,2010) usado para edição musical, o *Finale (*Finale, *2010)* para escrita de partitura e produção musical entre muitos outros.

1.2 Motivação

Com o advento da internet, surge uma forma rápida e eficaz para a troca de informações musicais, proporcionando um ambiente facilitador para aprendizagem de um instrumento musical, devido ao grande número de tutorias, vídeo aulas e sites voltados para o ensino

musical. Porém, ao se pensar na realidade brasileira, onde são recentes as escolas musicais e nem todos podem ter acesso ao ensino formal de música, com o aprendizado de leitura e escrita de partitura, métodos mais populares e mais simples de escrita são utilizados.

Um destes métodos de escrita musical é conhecido como **cifras musicais**. Este método é extremamente utilizado e devido a sua simplicidade, se tornou uma forma de escrita musical usada em música popular, sendo disseminada em larga escala entre os músicos.

Os Sistemas de Performance Instrumental (IPS) permitem aos usuários visualizarem a música sendo executada no instrumento de sua escolha. Entretanto, os IPS atuais encontram limitações, como por exemplo, a inadequação dos mesmos para quem baseia sua leitura musical em cifras de acordes, principalmente nos IPS específicos para violão/guitarra (Cabral, 2001).

Sendo assim, como motivação deste trabalho, podemos destacar o momento tecnológico e cultural favorável ao uso de novas tecnologias associadas à produção musical, à falta de produtos voltados especificamente para edição de cifras no mercado de softwares musicais e da grande quantidade de utilizadores do sistema de escrita musical baseado em cifras. Além da vivência do autor no meio musical e consultas a vários músicos profissionais, professores de música e também músicos amadores, foi possível constatar a falta de uma ferramenta com o objetivo especifico para a edição e escrita de cifras musicais.

1.3 Objetivos

Este trabalho tem como objetivo, construir uma ferramenta que seja simples e fácil de manusear, que ajude no processo de leitura e escrita de cifras musicas. Diferentemente de produtos famosos no mercado, como o *Finale* voltado para a escrita de partitura e do *Guitar-Pro* (Guitar Pro, 2010) usado para a escrita de tablatura, o Cifras

NotePad, nome dado ao produto fruto deste trabalho, é uma ferramenta especifica para a escrita e leitura de cifras musicais.

Com este objetivo foram desenvolvidas algumas funcionalidades as quais o músico pode ter acesso às informações mais importantes no processo de cifragem musical, que são as notas que compõem o acorde e, como este acorde pode ser executado. Além disso, o usuário poder ouvir o acorde escolhido quer seja de forma simultânea, isto é, todas as notas do acorde ao mesmo tempo dando ao musico o "*feeling"* de como o acorde irá soar harmonicamente ou então de forma arpejada, ou seja, nota por nota, proporcionando ao usuário escutar cada nota que compõem o acorde, da mesma forma que ela é apresentada pelo diagrama de acordes.

Estas funcionalidades, que permitem ao executante da música e para o usuário que esteja escrevendo a cifra, visualizar o diagrama do acorde desejado e poder ouvir o som do acorde, tanto de forma arpejada como simultânea, é de grande valia em ambas as situações. Segundo Roads (Roads, 1996), a representação dos acordes sobre a imagem do instrumento permitiu aos aprendizes uma visão mais intuitiva de como os acordes devem ser executados, entretanto, não se pode esquecer que acordes são genéricos a todos os instrumentos harmônicos. Recursos multimídia podem ser usados para obter-se uma notação mais rica.

Sendo assim, o desafio deste trabalho é desenvolver um sistema que seja capaz de interpretar a cifra escolhida, representá-la graficamente, através de uma possibilidade de representação do acorde, o qual deve ser construído dinamicamente, sem o prévio armazenamento de imagens e tocar exatamente as notas apresentadas, seja de forma simultânea ou arpejada.

Neste trabalho serão abordados fundamentos da notação musical para explicar a forma desenvolvida para a representação dos acordes musicais, a criação de uma metodologia para a construção de acordes através de um algoritmo genérico capaz de determinar como organizar as notas no instrumento para a formação do acorde desejado, o

desenvolvimento do sistema e suas funcionalidades. Além disso, devemos ressaltar o desenvolvimento de uma nova forma de definição de um acorde a partir da escala Cromática (Schoenberg, 1911).

Este trabalho está organizado da seguinte forma: no capítulo 2, são abordados os fundamentos da notação musical, onde é descrito como as cifras são utilizadas para representar os acordes musicais; no capítulo 3, é apresentado o problema de como representar os acordes computacionalmente; o capítulo 4, aborda a metodologia desenvolvida neste trabalho, para a construção e representação dos acordes musicais; o capítulo 5 descreve a arquitetura do sistema e forma de implementação do sistema; O capítulo 6 descreve o software construído neste trabalho e suas funcionalidades e no capítulo 7 são apresentadas algumas conclusões e propostas para trabalhos futuros.

2. FUNDAMENTOS DA NOTAÇÃO MUSICAL

2.1 Cifras Musicais

Cifras são símbolos criados para representar o acorde de uma maneira prática. A cifra é composta de letras, números e sinais. É o sistema predominantemente usado em musica popular para qualquer instrumento (Chediack, 1986). As cifras musicais representam as notas que devem ser executadas pelo músico ao tocar uma determinada harmonia. Uma cifra musical delineia a harmonia que deve ser tocada, porém não necessariamente precisa ser executada exatamente como escrita; a escolha da forma mais adequada do acorde fica a cargo do músico executante.

Com o uso de cifras substituímos os nomes das notas musicais pelas sete letras iniciais do alfabeto.

A – LÁ
B – SI
C – DÓ
D – RÉ
E – MI
F – FÁ
G - SOL

Com o uso de símbolos e números podemos indicar as alterações ou acidentes que são, respectivamente, sustenido ou bemol (#, b), qualidades como menor ou maior e extensões que são as notas adicionadas no acorde.

As letras são consideradas como notação básica e serão utilizadas neste trabalho para representar os acordes, sendo X o símbolo utilizado para representar uma cifra qualquer (C, D, E, F, G, A, B).

Popularmente o padrão de cifragem musical adota alguns pressupostos como, por exemplo, somente o X indica que o acorde é maior sem necessidade de indicação extra e um acorde X7 indica que esta sétima é menor.

Questões mais profundas sobre intervalos musicais e explicações mais detalhadas sobre a formação dos acordes serão suprimidas deste trabalho, pois o intuito é apenas dar uma noção do funcionamento das cifras musicais para mostrar o funcionamento e o processo de elaboração do sistema. Porém o conhecimento e pesquisa das possibilidades de escritas musicais e formação de acordes foram fundamentais na etapa de construção do conhecimento e elaboração do sistema.

Na Tabela 1, temos a representação e o significado de alguns acordes, com o intuito de ilustrar a representação dos acordes através das cifras musicais.

Tabela 1: Representação Gráfica dos acordes musicais

Símbolo e números	Significado
X	Acorde Maior
X#	Acorde Sustenido
Xb	Acorde Bemol
Xm	Acorde Menor
X7M ou XMaj7	Acorde com sétima maior
X7	Acorde com sétima menor
Xm7M	Acorde menor com sétima maior
Xm7	Acorde menor com sétima menor
X5Aum	Tríade aumentada
Xm(5b)	Tríade Diminuta
X#m	Acorde Sustenido Menor
X#7M ou XMaj7	Acorde Sustenido com sétima maior
X#7	Acorde Sustenido com sétima menor
X#m7M	Acorde Sustenido menor com sétima maior
X#m7	Acorde Sustenido menor com sétima menor
X#5Aum	Tríade Sustenida aumentada
X#m(5b)	Tríade Sustenida Diminuta
Xbm	Acorde Bemol Menor
Xb7M ou XMaj7	Acorde Bemol com sétima maior
Xb7	Acorde Bemol com sétima menor
Xbm7M	Acorde Bemol menor com sétima maior
Xbm7	Acorde Bemol menor com sétima menor
Xb5Aum	Tríade Bemol aumentada
Xbm(5b)	Tríade Bemol Diminuta

2.2 Acordes Musicais

Acorde é uma combinação de sons simultâneos ou sucessivos, quando arpejados, e cifras são símbolos criados para representar acordes, sendo compostos de letras, números e sinais. Acordes possuem três (tríades) ou quatro sons (tétrades) simultâneos: já os que possuem cinco ou mais sons são tétrades com notas acrescentadas (Chediak, 1984).

Os símbolos que representam acordes através das cifras variam de cultura para cultura e estão intimamente relacionados com o estilo musical dos músicos, sendo o idioma um dos fatores envolvidos nesta questão, não havendo ainda um padrão estabelecido, o que pode dificultar a leitura da musica. Entretanto, estes símbolos são largamente utilizados nas notações musicais mais simples e populares (Sher,1991) e se concentram no componente harmônico da música, supondo o conhecimento da melodia e do ritmo por parte do músico. Visando a um público crescente que demanda simplicidade, é comum haver uma separação dos diversos elementos musicais (West et. al 1991).

Por mais que não exista um padrão de cifras musicais existe um consenso de formas básicas onde a escrita deve buscar ser sempre clara e objetiva; sendo assim, por mais que existam variações, um músico experiente é capaz de se guiar por diversos padrões de cifras musicais.

Neste trabalho está sendo considerado o padrão difundido por autores brasileiros e utilizado na música popular brasileira e bossa-nova. Além deste padrão ainda são consideradas algumas notações diferenciadas que são encontradas no padrão americano, as quais são utilizadas em revistas especializadas e por alguns músicos ligados ao Jazz. Sendo assim com a síntese destas duas notações este trabalho é capaz de cobrir as notações mais comuns e mais importantes na música ocidental.

3. O PROBLEMA DE REPRESENTAÇÃO DE ACORDES

Este capítulo lida com o problema de como representar os acordes existentes, e como apresentar o diagrama desses acordes. Uma abordagem natural seria criar um banco de imagens armazenando os diagramas dos acordes possibilitando a exibição dos mesmos quando requisitada. Porém, teríamos um problema, pois muitos são os diagramas possíveis de serem representados. Além disso, um acorde pode ter mais de um diagrama capaz de representar o mesmo, ou seja, existe um número muito grande de acordes possíveis. Se pensarmos somente em acordes tríades maiores e menores com suas extensões básicas, ou seja, maiores e menores, quintas aumentadas e diminutas, teríamos 72 acordes distintos, sem considerar acordes tétrades, notas acrescidas e inversões, além de variações na exibição do diagrama.

Nas Figuras 1, temos respectivamente, uma abstração de como o acorde é formado através da utilização de grafos orientados, partindo da tônica para a terça e a quinta, que são as notas que formam um acorde tríade. Já na Figura 2 temos a representação do acorde tétrade.

A Figura 1, ilustra a grande possibilidade de variação de um acorde, pois temos 12 tônicas possíveis: a terça pode ser maior ou menor e a quinta pode ser justa, menor ou maior.

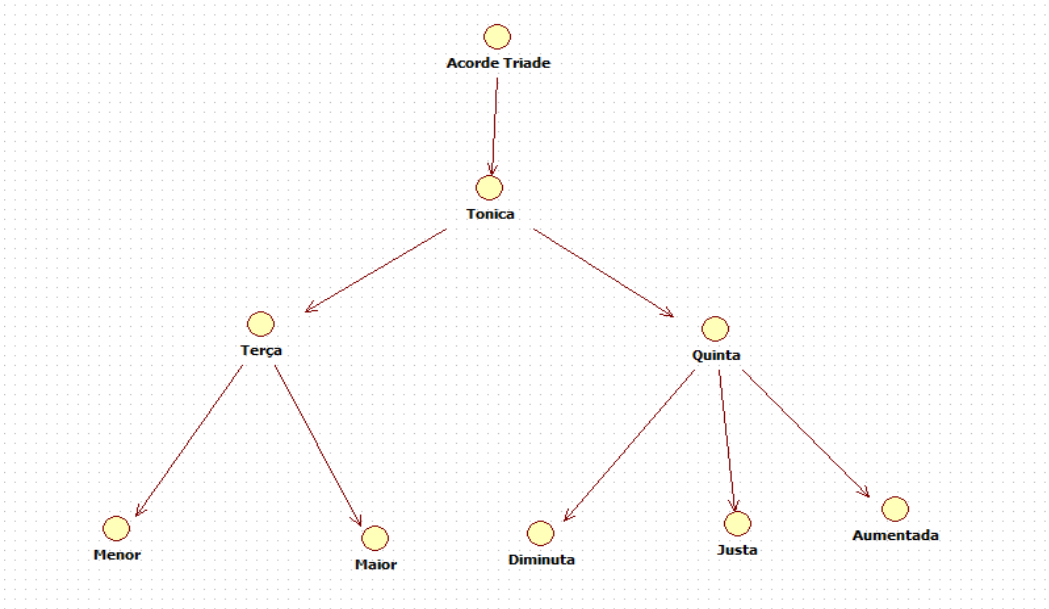


Figura 1: Árvore do acorde tríade

Ao pensar nos acordes tétrades o número de possibilidades aumenta, pois, além das mesmas possibilidades do acorde tríade, temos o acréscimo da sétima que pode ser menor ou maior. Sendo assim, temos 96 possibilidades, como mostra a Figura 2.

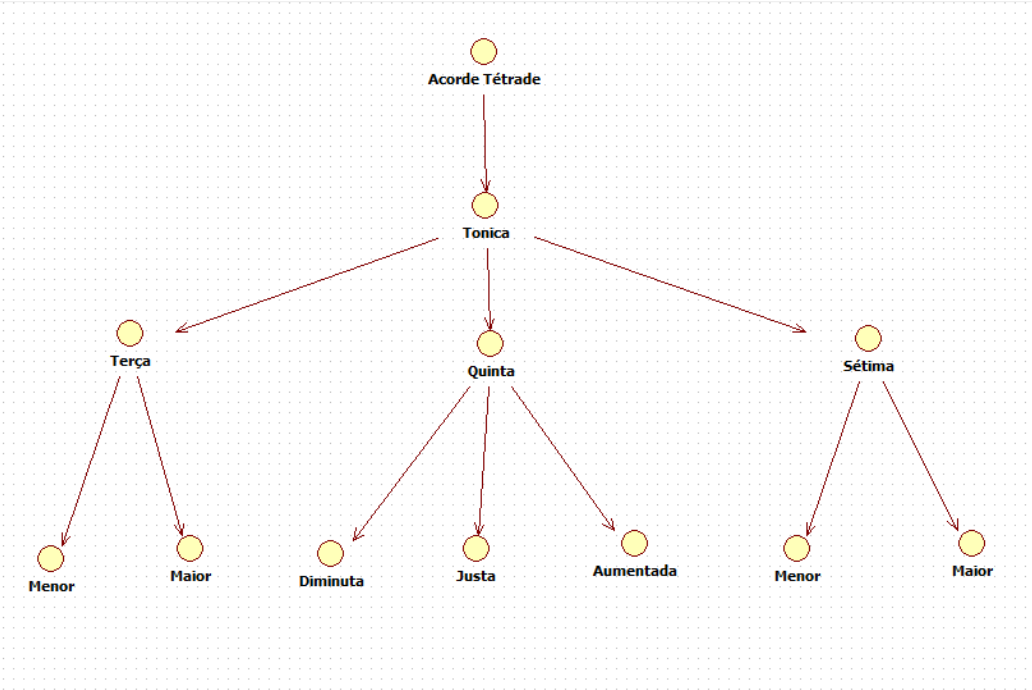


Figura 2: Árvore do acorde tétrade

Além das possibilidades mostradas, é importante ressaltar que não estão sendo consideradas as notas adicionais como nona, sexta, décima primeira, décima terceira, quarta suspensa entre outras. Com isto podemos ilustrar que para a construção de um software com um conjunto básico de acordes seria necessário armazenar em um banco de dados aproximadamente 432 acordes. Devido ao grande número de acordes possíveis armazenar imagens previamente de todos eles, é uma solução inviável para os objetivos deste trabalho. Além disso, só está sendo considerado o problema da construção do diagrama do acorde, pois ao pensar na funcionalidade de reproduzir o som de um acorde da mesma forma que ele é realmente feito no instrumento temos um problema bem maior para ser superado.

O violão ou guitarra é um instrumento que possui seis cordas, (isto para instrumentos convencionais, pois atualmente, em alguns

instrumentos especiais, podemos ter variações destes números), ou seja, são seis possibilidades de notas a serem tocadas a cada casa do instrumento. Em um instrumento com 19 casas, por exemplo, teremos 114 notas para serem escolhidas para o acorde desejado. Vale ressaltar que atualmente no mercado existem guitarras com até 27 casas (Figura 3), aumentando esta possibilidade.

Figura 3: Guitarra com 27 trastes

Sendo assim, este trabalho tem como desafio desenvolver uma forma eficiente de representar computacionalmente os acordes musicais, através de um mapeamento que possibilite a visualização do acorde e posteriormente a reprodução sonora do mesmo. Além disso, vale ressaltar que este trabalho se engloba na área de computação musical, uma área pouco explorada na ciência da computação, tendo uma grande carência em termos de publicações e trabalhos para serem usados como referência. Podemos, então, apontar também como problema a falta de trabalhos acadêmicos com a proposta de unir conceitos musicais para a construção de uma ferramenta computacional ou usar um software computacional como uma ferramenta musical, pois ambos são conceitos muito recentes em termos acadêmicos.

4. METODOLOGIA UTILIZADA PARA CONSTRUÇÃO E REPRESENTAÇÃO DOS ACORDES MUSICAIS

Para tratar o problema apresentado foi desenvolvida uma nova forma de representar os acordes buscando construir um padrão mais genérico e flexível de representar um acorde e que permitisse uma manipulação computacional de maneira otimizada.

A idéia utilizada é fundamentada na escala Cromática e usa o conceito de formação de acordes baseado no modelo de intervalos, possibilitando desenvolver um algoritmo que seja capaz de representar os acordes possíveis de uma forma genérica.

Por exemplo, um acorde tríade (três sons) é formado pela primeira, conhecida como Tônica, terceira e quinta nota de sua escala correspondente, ou seja, baseando-se na escala da tonalidade do acorde desejado, a partir da Tônica, acrescentarmos a terça e a quinta, para formarmos um acorde maior.

A partir da Tônica definimos uma base, e com o tipo do acorde, seja, maior, menor, alterado dentre outros, definimos quais intervalos serão usados para montar o acorde. Assim, é possível construir um algoritmo genérico em relação a qual acorde será montado, sem a necessidade do armazenamento de varias escalas e estruturas. Todos os acordes serão montados a partir do algoritmo desenvolvido neste trabalho.

4.1 Reconhecendo acordes

Em seu trabalho sobre cifras personalizáveis, Leandro Costalonga (Costalonga, 2003), utiliza um autômato para a validação de cifras, sendo assim definindo uma forma de reconhecer qual o acorde que está sendo utilizado. Para representar uma cifra primeiramente é necessário reconhecer qual acorde o usuário deseja inserir ou o que se busca indicar com a cifra dada pelo usuário do software.

As regras de formação dos acordes em conjunto com a linguagem definida tornam possível a construção de um autômato finito para validação de cifras, representado pela quíntupla A = (∑, Q, δ, q0, F) onde:

a) ∑ é o alfabeto de símbolos de entrada, ∑ = nota U alt U sus U var
nota = {A, B, C, D, E, F, G}
alt = {#, b}
var = {º, m, 5}
susN = {sus2, sus4, sus9, sus11}
b) Q é conjunto de estados possíveis, Q = {S1, S2, S3, S4, S5, S6, S7}
c) δ é a função parcial de transição δ:Q x ∑ → Q
d) q0 é o estado inicial, q0 = S1
e) F é o conjunto de estados finais, F = {S3, S4, S5, S6, S7}

Graficamente o autômato A, está representado na Figura 4.

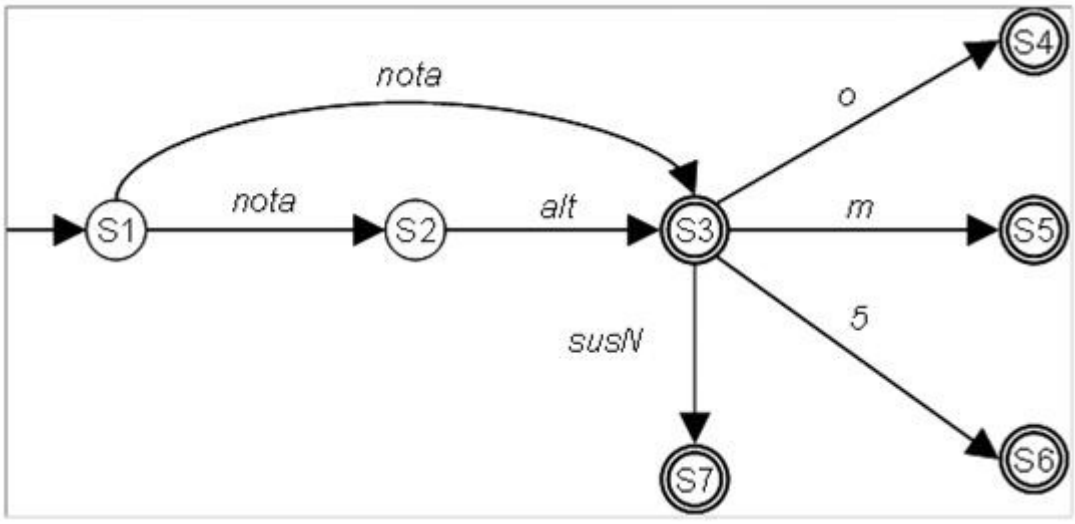


Figura 4: Autômato de validação de Cifras

Para este trabalho foi considerada uma abordagem similar, porém um pouco mais simplificada, onde temos as notas principais,

representadas pelas letras do alfabeto. Neste caso usamos as próprias cifras na representação, buscando uma notação simplificada e similar à usada musicalmente. As alterações são as indicações de sustenido ou bemol, pois toda cifra pode ser sustenida ou bemol, seguidas das variações no acorde e pelas inversões.

nota = {A, B, C, D, E, F, G}
alt = {#, b}
var = {º, m, 5, 7}
inv = {1, 2}

Na estrutura criada na Figura 5, temos um pequeno exemplo de como reconhecer um acorde partindo do conceito de sua própria formação e como os intervalos musicais podem variar. Em nosso exemplo, temos um acorde tríade simples sem notas adicionais. Começamos avaliando o primeiro digito da cifra, o qual indica a nota inicial do acorde, caso o próximo digito seja "#" temos uma acorde sustenido, caso seja "b" temos um acorde bemol e, caso após o primeiro dígito, não apareça nenhum dos dois símbolos temos um acorde natural. Sendo assim, em nossa primeira avaliação determinamos a nota inicial e a natureza do acorde, o qual pode ser natural, sustenido ou bemol e a partir deste estado avaliamos o próximo dígito. Ele indica se o acorde é maior ou menor, de acordo com a alteração na terça do acorde e posteriormente alterações nas notas subseqüentes usadas para formar o acorde.

Portanto, neste trabalho o reconhecimento das cifras se baseia na própria formação do acorde verificando o estado de cada nota que compõe o acorde através dos estágios de avaliação pré determinados que possam ser estendidos de acordo com a complexidade da cifra.

A seguir (Figura 5), temos a representação de uma tríade que pode ser natural, sustenida ou bemol representando as possíveis alterações e as possíveis variações da sua terça, sendo maior ou menor, levando

um uma quinta justa como é chamado um intervalo de quinta que não sofre alterações.

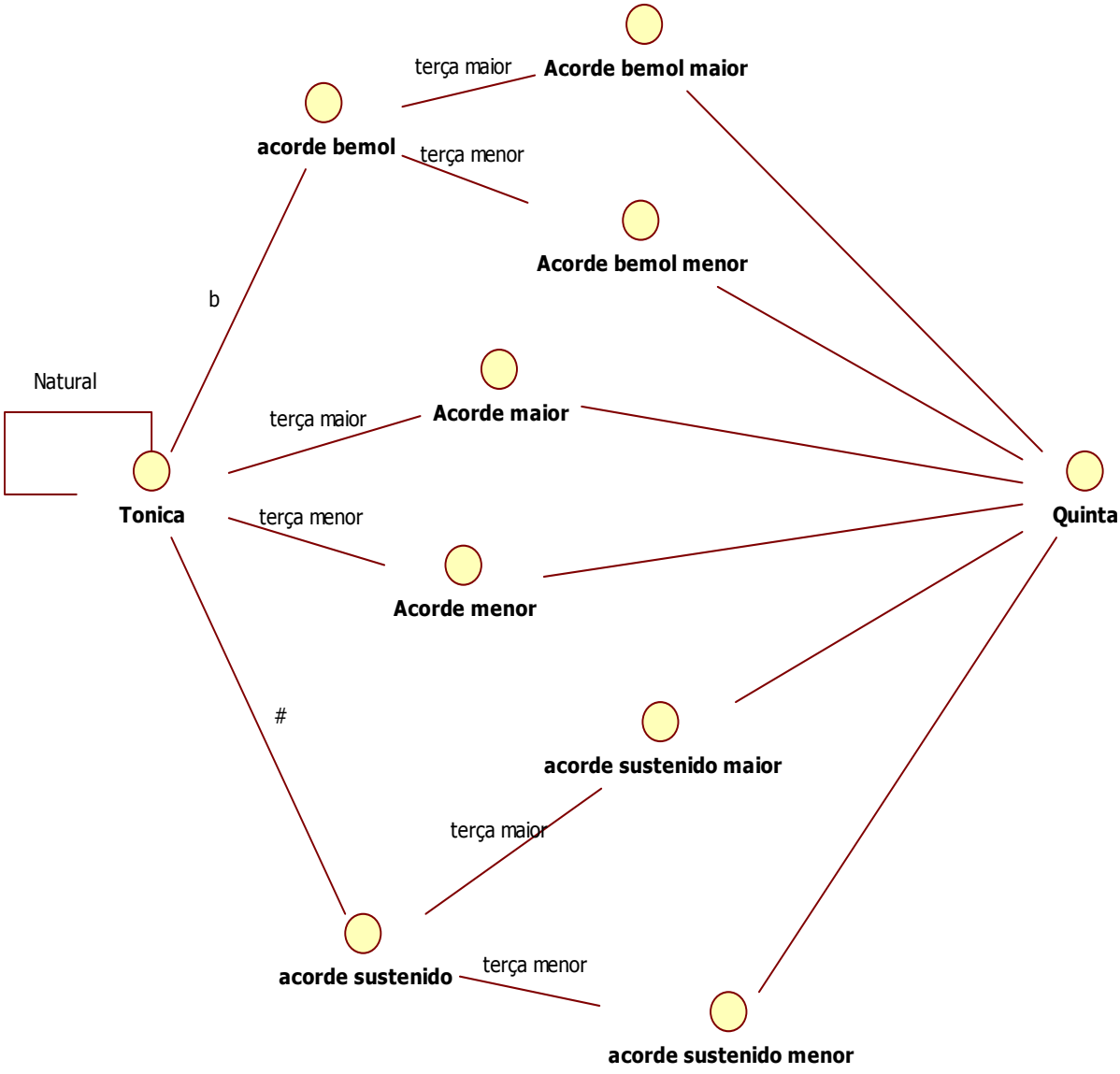


Figura 5: Representação de uma tríade, com o mapeamento das possibilidades de variações entra as notas fundamentais.

Como neste trabalho o acorde é identificado de acordo com a sua formação, inicialmente é verificada a tônica utilizada, como mostra o algoritmo 1, implementado para identificação de cifras musicais. Na implementação do algoritmo 1, a tônica é sempre nomeada como base.

```
inteiro verificaBase(String texto)
{
caracter tom = primeiro caracter da string (texto);
    caso  tom = 'C':
        base = 0;
    caso  tom = 'D':
        base = 2;
    caso  tom = 'E':
        base = 4;
    caso  tom = 'F':
        base = 5;
    caso  tom = 'G':
        base = 7;
    caso  tom = 'A':
        base = 9;
    caso  tom = 'B':
        base = 11;

    retorna base;
}
```

Algoritmo 1: reconhecimento das cifras musicais

Após o reconhecimento da tônica é necessário verificar se o acorde possui alguma alteração, ou seja, se o acorde é sustenido(#) ou bemol(b), esta verificação é realizada com o algoritmo para verificação de alteração do acorde (Algoritmo 2).

```
verificaAlteracao(String texto)
{
    caracter tom = segundo caracter da string (texto);

    caso tom == '#':
    {
        base = base + 1;
        alterado = true;
    }
    caso tom == 'b':
    {
        base = base - 1;
        alterado = true;
    }

    retorna base;
}

```

Algoritmo 2: Algoritmo para verificação de alteração do acorde.

Após a identificação da base e se o acorde possui alguma alteração, podemos identificar qual variação o acorde possui e posteriormente se ele possui alguma inversão.

4.2 Mapeamento

Os acordes são formados a partir de sua escala utilizando seus intervalos correspondentes. Por exemplo, partindo de escala de Dó maior podemos montar os acordes maiores a partir dos intervalos de formação do acorde.

Escala de Dó Maior: Dó, Ré, Mi, Fá, Sol, Lá, Si

No estudo musical é comum generalizar os intervalos dando graus às escalas com uso de algarismos romanos.

Graus da escala: I, II , III, IV, V, VI, VII

Escala de Dó Maior: I - Dó, II - Ré, III - Mi, IV - Fá, V - Sol, VII - Lá, VII Si

Escala de Dó Menor: I - Dó, II - Ré, III - Mi bemol, IV - Fá, V - Sol, VI - Lá bemol, VII - Si bemol

Sendo assim, uma tríade maior é formada pelos graus I, III, V da escala maior a partir de sua tônica. No acorde de Dó maior teríamos as notas: Dó, Mi, Sol. Já o acorde de Dó menor é formado pelos graus I, III, V da escala menor, ou seja, Dó, Mi bemol e Sol.

Na teoria musical um acorde é montado a partir do seu campo harmônico (tonalidade), através do qual são estabelecidos os graus que compõem o intervalo do determinado acorde.

Note que, de acordo com esta abordagem baseada na teoria de formação de acordes, para formar um acorde teríamos que determinar qual escala usar e a partir da escala fazer uso da definição do acorde, ou seja, quais intervalos compõem o acorde para poder construir o mesmo.

Para a construção do software foi criada uma nova generalização usando a abordagem de intervalos, porém foi decidido usar somente a escala Cromática como base, pois a escala Cromática já possui todas as notas necessárias para a formação de qualquer acorde. Com isto, não se tem mais a necessidade de armazenar varias escalas para formar diversos acordes, pois todos os acordes serão baseados na escala Cromática.

A escala Cromática, é a escala formada por intervalos de semitom. Sendo assim partindo do Dó ascendentemente, teremos:

Dó, Dó#, Ré, Ré#, Mi, Fá, Fá#, Sol, Sol#, Lá, Lá#, Si

4.3 Montando os acordes

Agora vamos considerar a escala cromática como base, sabendo quais notas compõem um determinado acorde, por exemplo, o acorde de Dó maior que é composto pelas notas Dó, Mi e Sol. Podemos usar a escala cromática e determinar quais intervalos da escala cromática são usados. Pegando a escala cromática e atribuindo um índice a cada valor, vamos montar os acordes com o uso dos índices.

Exemplo:

Dó	Dó#	Ré	Ré#	Mi	Fá	Fá#	Sol	Sol#	Lá	Lá#	Si
0	1	2	3	4	5	6	7	8	9	10	11

Considerando a nossa abordagem para formação do acorde temos que o acorde de Dó maior será 0, 4, 7 (Dó, Mi, Sol).

Porém, a conclusão mais importante que podemos tirar desta diferente abordagem é da nova relação de intervalos para a formação do acorde, ou seja, partindo de uma base, que no nosso caso vamos considerar sempre a tônica, ao andarmos 4 intervalos para frente em nossa escala cromática e depois a partir deste ponto andarmos mais 3 intervalos para frente teremos um acorde maior. Vale ressaltar que se consideramos a escala cromática como uma escala circular após o 11 voltamos para o zero.

Vamos analisar outro exemplo. Agora vamos formar o acorde de Lá maior, e para isto a base será o Lá, que é a nossa tônica neste caso.

Lá = 9 + 4 = 13

Então teremos:

Lá = 9 + 4 = 13 = 1 -> Dó #

Agora vamos partir do Dó# e contar mais 3.

Do# = 1 + 3 = 4 -> Mi

Logo o acorde de Lá maior é composto pelas notas Lá, Dó# e Mi.

Esta mesma abordagem será usada para a construção de todos os outros acordes. Como exemplo, temos o Algoritmo 3, usado para a construção de um acorde maior e o Algoritmo 4, usado para a construção de um acorde tétrade.

```
1  ArrayList AcordeMaior (inteiro base)
2  {
3      Adiciona no acorde(base);
4      Adiciona no acorde (indice((base + 4)));
5      Adiciona no acorde (indice((base + 7)));
6      retorna acorde;
7  }
```

Algoritmo 3: Método usado para construção de um acorde maior.

O método índice é um método que verifica se o valor 11 foi ultrapassado. Caso tenha sido ultrapassado ele recomeça a contagem simulando uma lista circular. Acorde é um ArrayList onde, após a contagem de intervalos, serão armazenados os números correspondestes aquele determinado acorde. O acorde será usado para posteriores visualizações do mesmo, seja na construção do diagrama ou na reprodução sonora.

```
1  ArrayList AcordeMaior7 (inteiro base)
2  {
3      Adiciona no acorde (base);
4      Adiciona no acorde (indice((base + 4)));
5      Adiciona no acorde (indice((base + 7)));
6      Adiciona no acorde (indice(base+ 10));
7      retorna acorde;
8   }
```

Algoritmo 4: acorde maior com sétima

Como um acorde é formado por um conjunto de notas, deve ser definido um subconjunto das notas possíveis no instrumento que irá compor o acorde. Após ter formado o diagrama do acorde nota por nota, para a exibição dos diagramas, já foram escolhidas as notas usadas no instrumento para formar um acorde, ou seja, as cordas e a casas exatamente que serão tocadas para a execução do acorde.

Com isto podemos mapear exatamente as notas que devem ser usadas para a reprodução do acorde através das funcionalidades ligadas a reprodução sonora do acorde.

5. IMPLEMENTAÇÂO DO SISTEMA

Este capítulo irá descrever a arquitetura geral do sistema, explicando as classes do sistema, as decisões de implementação, assim como a abordagem utilizada para reprodução sonora. Informações mais detalhadas sobre a implementação do sistema podem ser obtidas consultando os apêndices A que contém os diagramas UML e o apêndice B, o qual possui o código fonte do sistema.

O sistema está dividido em três camadas, interface, controle e modelo utilizando o Padrão de projeto *MVC* (Model View Control),(Gamma et. al, 1995). Na camada de interface gráfica, ocorre a interação do usuário com o sistema. O usuário realiza uma ação e a camada de interface faz uma requisição à camada de controle, a qual processa esta requisição acessando ou não o modelo dependendo da ação. Na camada de modelo são representadas as entidades musicais na forma de objetos.

5.1 Classes da camada de interface

- Prototipo_1AboutBox – classe responsável pela visualização da janela *about*.
- Prototipo_1View – classe responsável pela visualização de toda a interface gráfica do sistema, incluído o painel principal, *toolbar*, menus, botões e área de texto.

5.2 Classes da camada de controle

- Prototipo_1App – classe *main* da aplicação.
- Desenha – classe de controle principal do sistema responsável pela ligação entre a camada de interface e a camada de modelo.

5.2.1 Classes para execução sonora

- MP3 – classe que possui uma instância do objeto Player, responsável por tocar o arquivo MP3.
- Tocador – abstração da classe MP3 com implementação de threads para a reprodução simultânea de vários arquivos do tipo MP3.

5.2.2 Classes para desenho do diagrama de acordes

- Shapes2JPanel – classe responsável pela exibição do diagrama do acorde.

5.3 Classes da camada de modelo

- Braço – classe responsável pela representação do braço do instrumento, onde temos casas e cordas as quais são usadas para a construção do diagrama de acordes.
- ModeloAcorde – Classe responsável pela representação dos acordes através dos intervalos predefinidos com o auxilio da escala cromática.

5.4 Reprodução sonora

Para está funcionalidade foram gravadas cada nota separadamente, as quais são armazenadas em arquivos separados no formato Mp3. Sendo assim permitindo ao usuário a possibilidade de escutar o acorde exatamente como é representado no diagrama e com o som real do instrumento. A opção de gravar os áudios reais foi escolhida por representar melhor a realidade e dar mais qualidade sonora ao áudio apresentado, pois soluções com o uso de reprodução de *MIDI (Musical Instrument Digital Interface)* (Hurber, 1999), geralmente usado neste tipo de aplicação computacional para gerar

áudio, não possui uma garantia de qualidade sonora, já que não contém o áudio propriamente dito e sim instruções para reproduzi-lo, sendo assim, sua reprodução é influenciada pelo hardware usado. Com os arquivos em mp3 é possível garantir a qualidade de áudio independente da qualidade do computador utilizado ou modelo de "placa de som" e também dar ao usuário o som real do instrumento.

6.SOFTWARE

O *Cifras NotePad*, nome dado ao produto fruto deste trabalho, consiste em um editor de texto com as funcionalidades básicas de edição, como escrita, abrir um arquivo texto, salvar o arquivo ou texto inserido.

Apesar de ser uma ferramenta simples em relação à edição de texto, o *Cifras NotePad*, possui como diferencial funcionalidades especiais voltadas para músicos, o qual é o foco deste trabalho. Independente do nível de conhecimento musical do usuário deste software, ele busca através de suas funcionalidades criar um ambiente de desenvolvimento e execução musical, especialmente para guitarristas e violonistas, pois ele é capaz de exibir os diagramas dos acordes representados pelas cifras escritas no editor e além disso tocar o acorde desejado.

É importante ressaltar que músicos nativos de outros instrumentos também podem fazer uso deste software, pois apesar de suas funcionalidades terem sido projetadas para guitarristas e violonistas, as cifras podem ser lidas por outros músicos de qualquer instrumento, já que as cifras musicais indicam a linha harmônica que deve ser seguida pelo músico independente de qual instrumento esteja sendo utilizado. Apesar da funcionalidade de exibição dos diagramas dos acordes ser específica para os instrumentos citados acima, a notas dos acordes são sempre as mesmas independente do instrumento utilizado, sendo assim, através da reprodução sonora do acorde, o usuário do sistema tem uma referência sonora de como é o acorde, sendo possível esclarecer duvidas em relação ao acorde e até mesmo permitir a reprodução do acorde pelo músico em seu instrumento nativo. Além disso, a reprodução sonora pode ser usada para auxiliar o processo de composição sem o uso de um instrumento musical, pois o usuário pode determinar uma harmonia musical escrevendo as cifras e depois ouvir cada acorde tendo uma prévia de como a música ira soar.

6.1 Interface do sistema

A interface inicial do sistema (Figura 6), para facilitar o uso o campo de edição de texto já vem com título, autor e introdução escritos, pois são campos gerais e comuns na maioria das cifras, porém são campos editáveis e podem ser alterados ou retirados.

Todas as funcionalidades encontradas no menu do sistema, estão presentes na barra de ferramentas e podem ser acessadas através de um simples clique no botão.

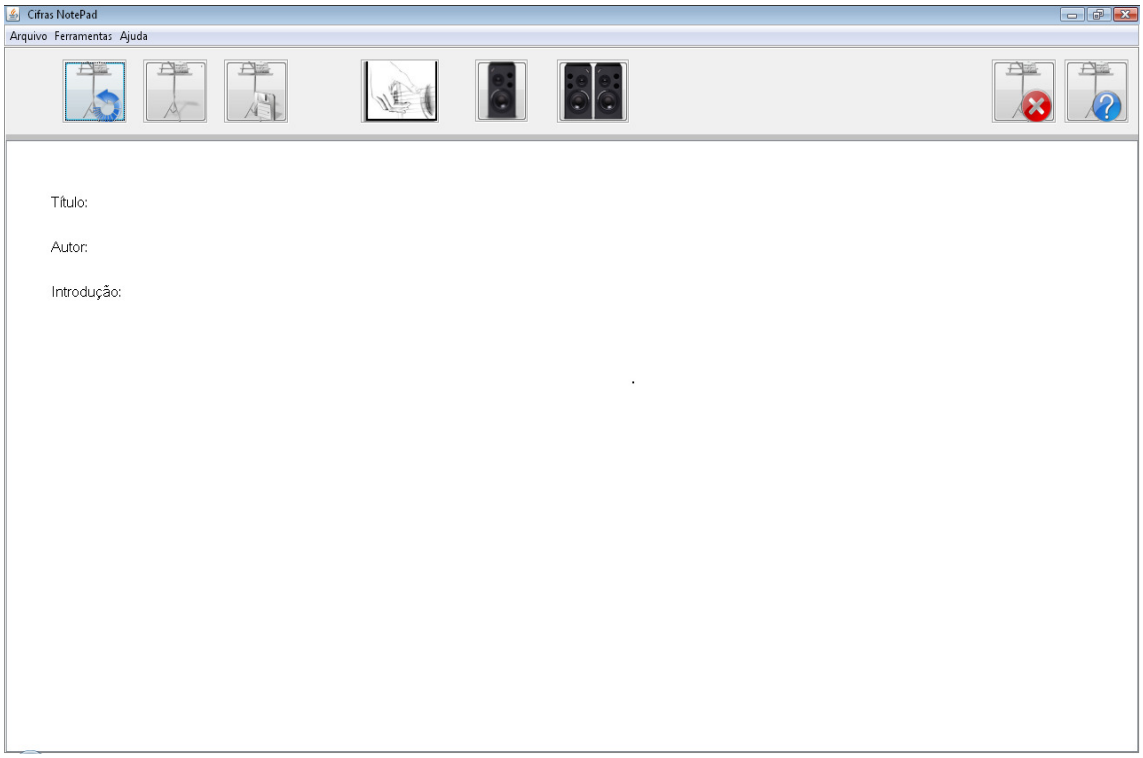


Figura 6: tela inicial do software

Tendo como referência a ordem dos botões da esquerda para direita, os três primeiros botões são relacionados à manipulação de arquivos. Suas funcionalidades são:

- Novo – gera uma tela limpa, como se fosse abrir um arquivo em branco.
- Abrir – abre uma janela para seleção do arquivo que o usuário deseja abrir (Figura 7).

- Salvar – é aberta uma janela similar a usada na ação de abrir um arquivo, onde o usuário pode dar nome ao arquivo criado e selecionar em qual local deseja gravar.

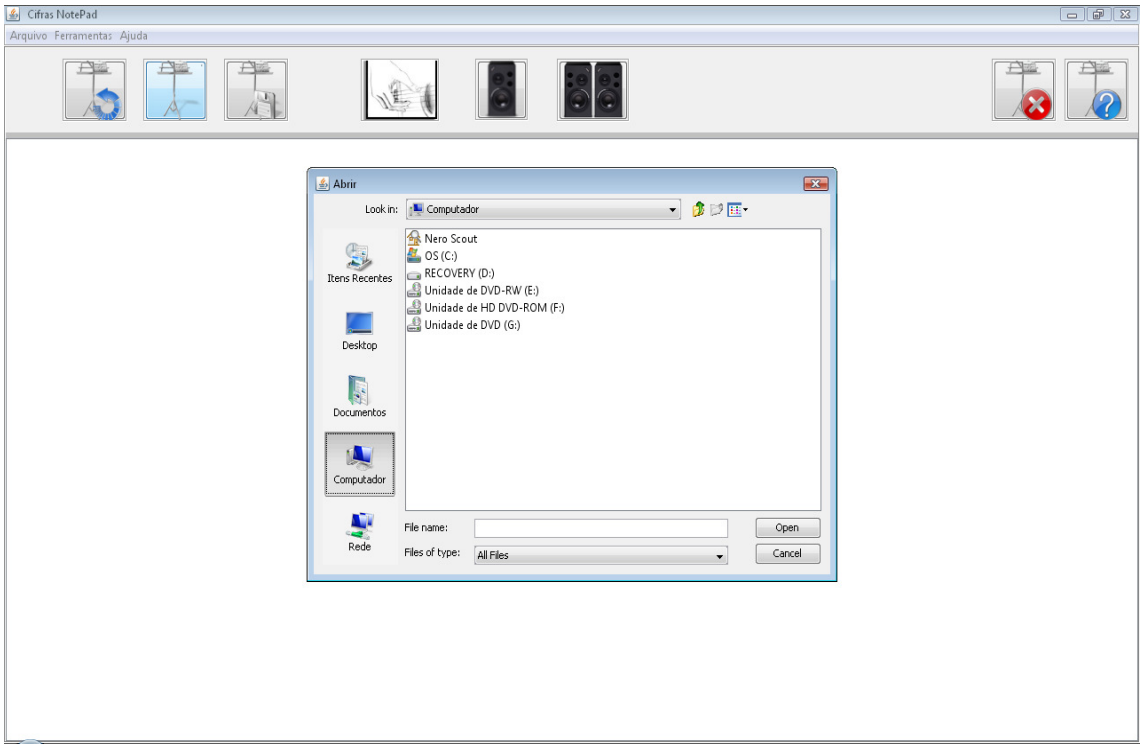


Figura 7: Usuário seleciona um arquivo salvo anteriormente

Os outros 3 botões ao centro, estão relacionados as funcionalidades voltadas para o auxílio musical. Quando um destes botões é selecionado é exibida uma janela para que o usuário digite a cifra desejada (Figura 8). Estes botões possuem respectivamente as seguintes funções:

- Visualizador de diagramas – permite a inserção da cifra desejada (Figura 8) e posteriormente exibe o diagrama do acorde inserido (Figura 10).
- Tocar arpejado – após a inserção da cifra desejada, são reproduzidas sonoramente as notas referentes ao acorde escolhido de forma consecutiva.

- Tocar simultâneo - após a inserção da cifra desejada,são reproduzidas sonoramente as notas referentes ao acorde escolhido de forma simultânea.

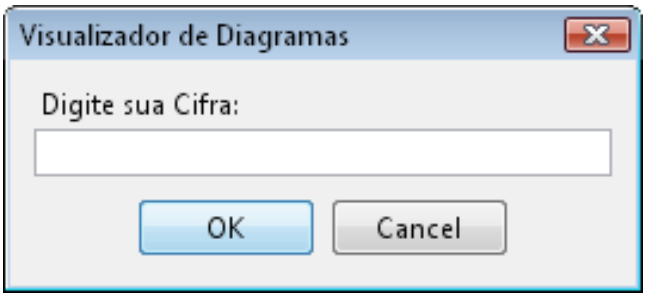


Figura 8: janela para o usuário digitar a cifra desejada.

Nesta interface (Figura 9), o usuário usa o visualizador de diagramas para visualizar uma das cifras da música em questão.

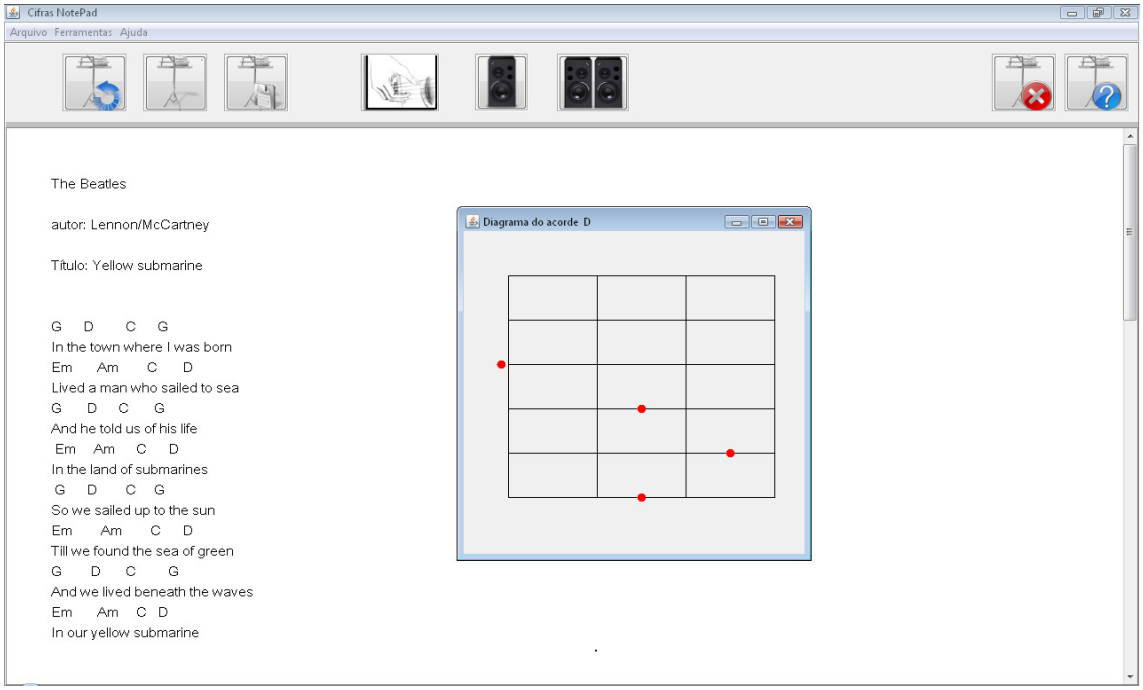


Figura 9: visualizando o diagrama do acorde de Ré maior.

Na figura 10, temos o software já em funcionamento, onde o usuário já tem uma cifra digitada ou aberta ou até mesmo inserida de alguma outra fonte.

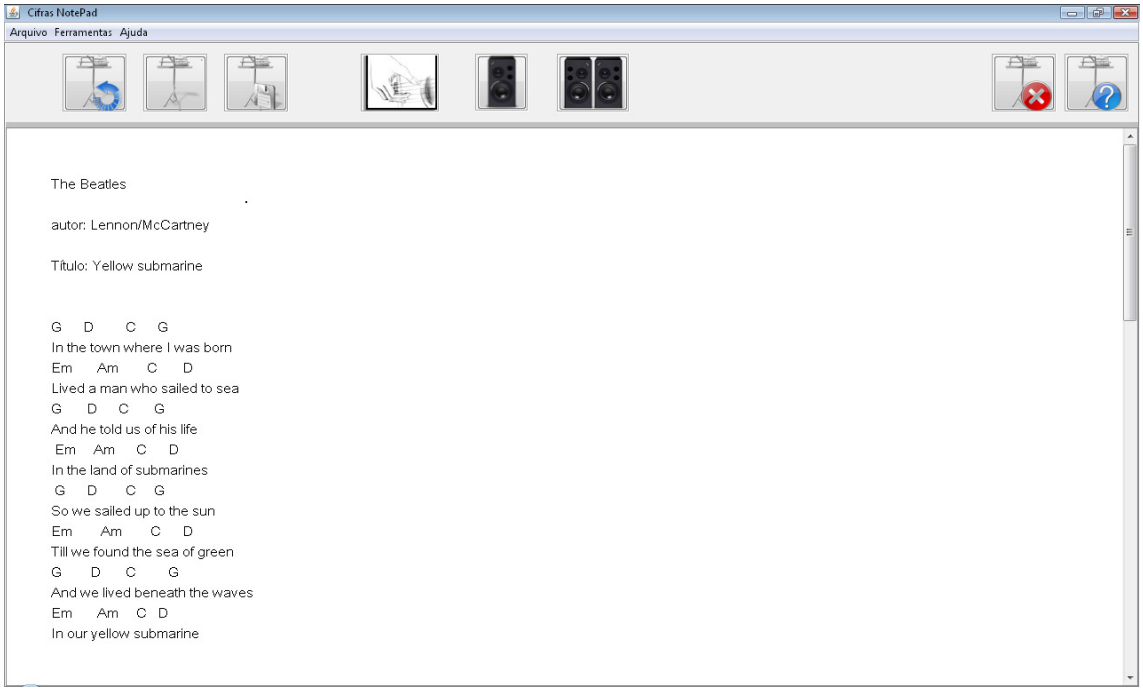


Figura 10: Software com uma cifra já inserida

Os dois botões a direita da interface do sistema são os botões responsáveis pelas ações de sair do sistema e exibição da tela “sobre” (Figura 12), respectivamente.

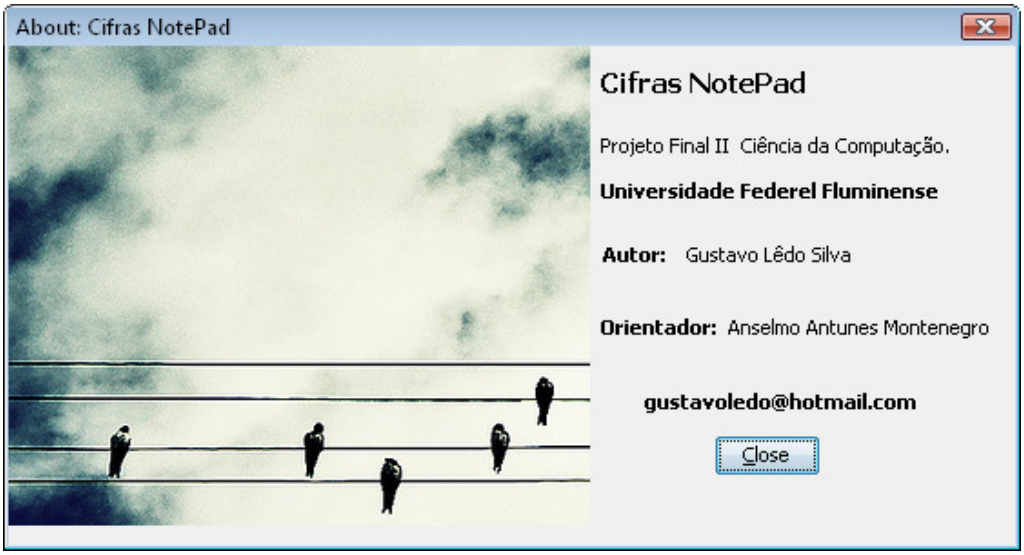


Figura 11: Exibição da janela "Sobre"

6.2.Funcionalidades

6.2.1 Diagramas

O braço do violão ou guitarra é dividido em casas as quais são separadas por trastes, conforme a figura abaixo:

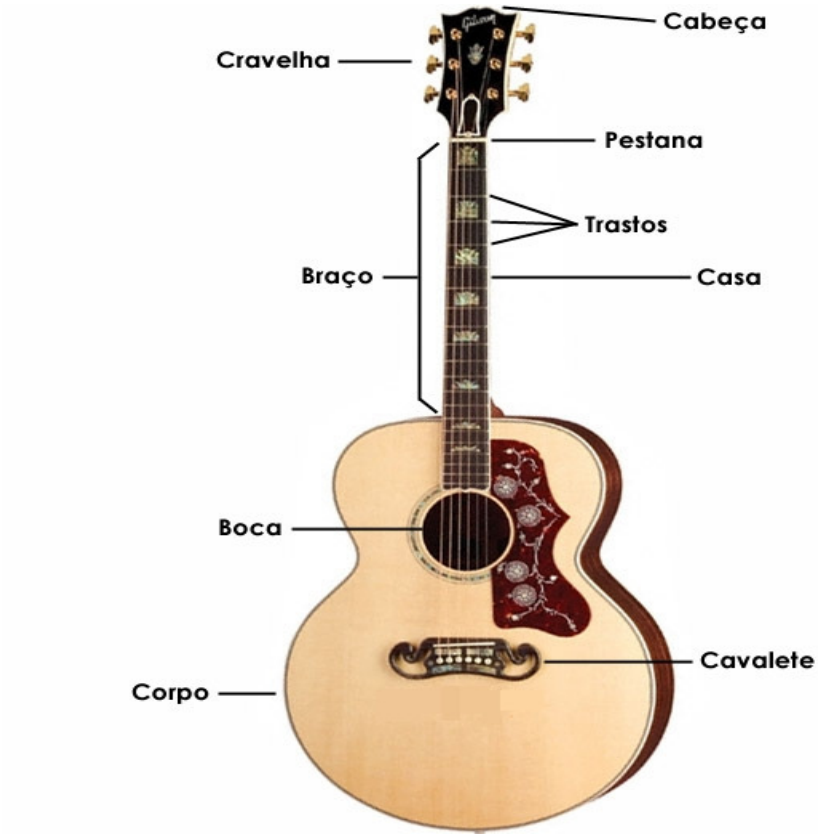


Figura 12: Violão

O diagrama do acorde é exibido fazendo uma alusão ao braço do instrumento o qual é representado pelos trastes, casas e as cordas. Os pontos vermelhos representam as notas as quais devem ser tocadas. Neste exemplo (Figura 13), temos o acorde de Dó maior.

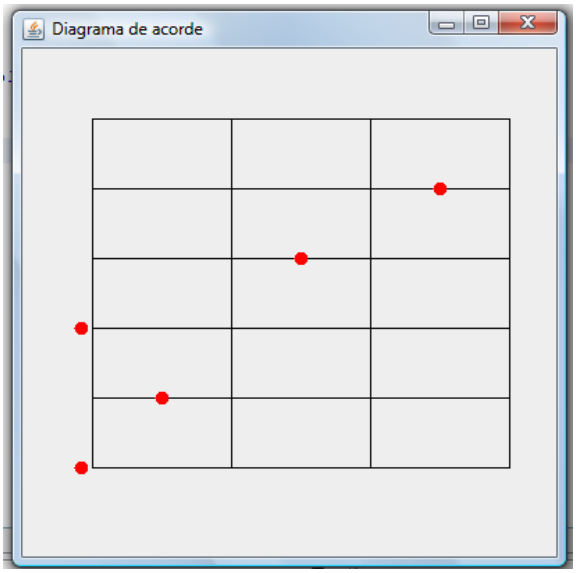


Figura 13: Diagrama de Acorde

No diagrama temos uma representação onde, cada coluna do diagrama representa uma casa, cada linha vertical representa um traste do violão ou guitarra e cada linha horizontal representa uma corda. As marcações vermelhas antes da primeira coluna indicam que aquela corda deve ser tocada solta, ou seja, sem nenhum dedo pressionando a corda e as indicações vermelhas dentro do diagrama indicam que aquela corda deve ser pressionada naquela casa (Figura 14).

Na figura 10 temos o mesmo acorde do diagrama, mas com a visão do braço do violão.

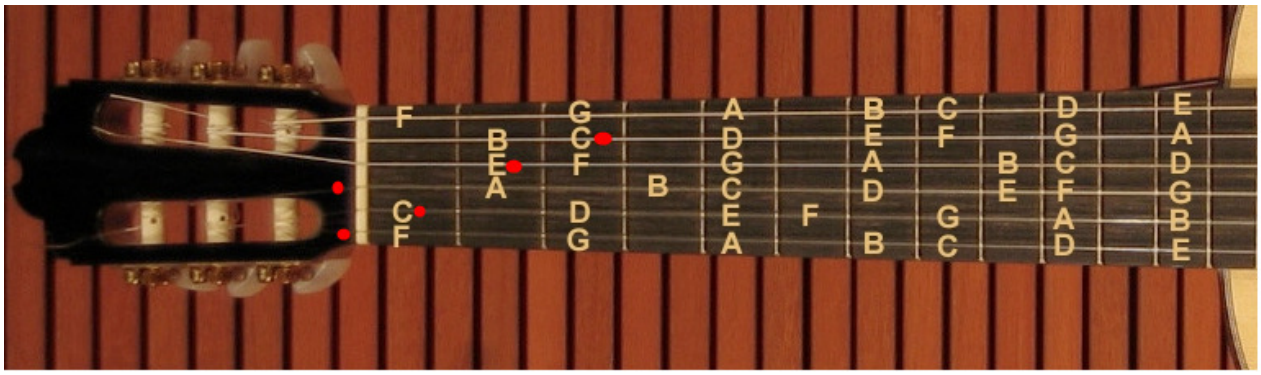


Figura 14: representação do acorde dó maior exibido na figura 10, com a imagem do braço do violão.

Outros exemplos de Diagramas (Figuras:15,16):

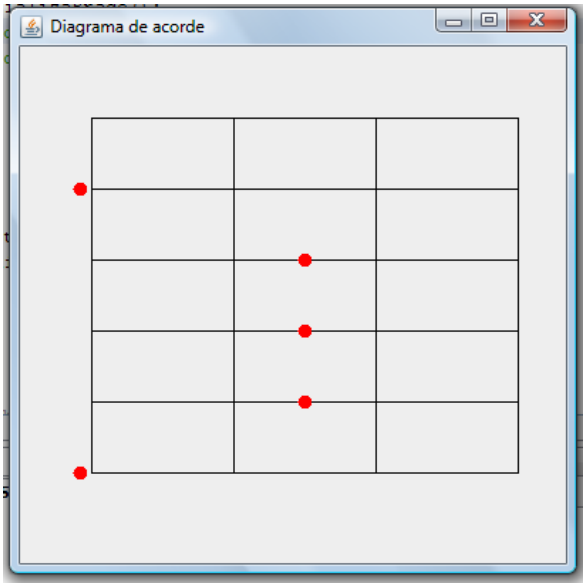


Figura 15: Acorde de Lá maior

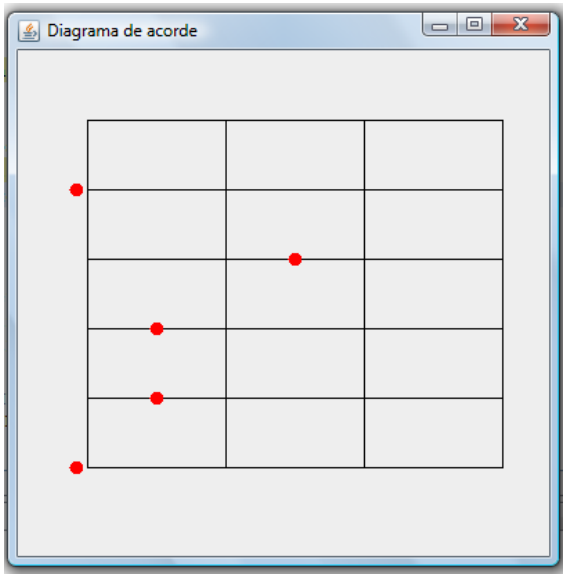


Figura 16: Acorde de Lá menor com sétima maior

Para a construção do diagrama de acordes foi utilizada a API Java 2D, onde são utilizadas linhas para a formação das grades e pequenas elipses centralizadas na linha para a construção das marcações em vermelho as quais indicam os dedos.

As linhas são traçadas a partir de coordenadas cartesianas, calculadas de forma a se obter uma grade (Algoritmo 5).

```
inteiro y = 50;
inteiro x = 50;
// desenha as linhas
Para (i = 0; i<6; i++)
   {
       Desenha linha com as cordenadas cartesianas( 50, y, 350, y ) );
       y= y+50;
    }

 // desenha as colunas
Para(i = 0; i<4; i++)
    {
       Desenha linha com as cordenadas cartesianas( x, 50, x, 300 ) );
       x = x+ 100;
    }
```

Algoritmo 5:algoritmo para desenho da grade usada na exibição do diagrama.

As marcações vermelhas (Figura 16) são pequenas elipses centradas nos pontos desejados.

Neste caso as variáveis *casa* e *corda* indicam o ponto onde a elipse deve ser desenhada. A casa e a corda as quais devem ser usadas são calculadas de acordo com o acorde e a determinação de qual nota deve ser tocada.

6.2.2 Reproduzindo o acorde

Além de visualizar o acorde desejado, é possível ouvi-lo. Para isto o software oferece duas opções para reprodução sonora do acorde: reprodução de forma arpejada, onde é tocada cada nota que compõe o acorde de cima para baixo, ou de forma simultânea, onde todas as notas que compõem o acorde são tocadas simultaneamente.

O áudio apresentado corresponde exatamente ao som de cada nota escolhida pelo sistema para formar o acorde, correspondendo exatamente ao acorde que é exibido no visualizador do diagrama do acorde. Com isto, esta funcionalidade proporciona ao usuário do sistema poder ouvir exatamente uma possibilidade de execução da música. Deste modo, é possível saber como os acordes irão soar

somente com o uso do sistema, sem ter um instrumento em mãos e também permitir que músicos que tocam outros tipos de instrumento possam ouvir o acorde e ter uma prévia de como a música esta sendo conduzida harmonicamente.

6.3 Determinando as notas a serem tocadas

Após a identificação do acorde, onde são identificadas quais as notas que compõem um determinado acorde, é necessário mapear as notas no braço do instrumento para a formação do diagrama do acorde.

Para mapear as notas do acorde é necessário percorrer cada corda e cada casa do instrumento procurando a nota desejada. Para esta versão do software foi decidido restringir o número de casas, sendo assim, eliminando a possibilidade do acorde formado ser impossível de ser feito na prática devido ao espaçamento entre notas. Por outro lado, para cada acorde é exibido somente uma possibilidade de diagrama.

Diferentemente de outros instrumentos como, por exemplo, o piano, no violão ou guitarra os acordes não possuem as suas notas organizadas de forma canônica, ou seja, na ordem em que o acorde é formado, devido às particularidades das organizações das notas em cada instrumento.

Sendo assim a determinação das notas a serem tocadas se torna uma tarefa mais complicada. Primeiramente definimos o baixo ou bordão, que é como é conhecida a Tônica quando é a primeira nota a ser tocada. Tendo como referência as cordas de baixo pra cima, o baixo deve ficar localizado em uma das 3 cordas superiores ou seja, na quarta, quinta ou sexta corda. Após definir a posição do baixo, as cordas inferiores da posição escolhida para o baixo são percorridas buscando as notas restantes do acorde, sendo que as notas podem ser repetidas e inclusive a tônica pode aparecer novamente. Porém, vale ressaltar que é fundamental que cada nota que compõe o acorde deva ser tocada pelo menos uma vez.

No Algoritmo 6, temos o método utilizado para definição da posição do baixo no acorde. Após ter a posição do baixo definida, são definidas as outras notas para formar o acorde no braço do instrumento. Ao achar uma determinada nota é armazenada a corda e a casa em um *ArrayList* de posições que vai ser usado para desenhar o diagrama do acorde (Algoritmo 7) .

```
1  Inteiro buscaPosicaoBaixo (ArrayList acorde)
2  {
3      inteiro p = -1;
4      numeroCorda = 6;
5
6      Enquanto(p<0)
7      {
8          p = busca na corda (numeroCorda) o primeiro elemento do acorde (Acorde);
9          numeroCorda--;
10     }
11         retorna p;
12 }
```

Algoritmo 6: método utilizado para desenhar a grade, usada na exibição do diagrama.

```
1  inteiro buscaPosicao (ArrayList acorde)
2  {
3      inteiro p = -1;
4      inteiro   = numeroCorda;
5
6      Para i = numeroCorda; i > 0; i--
7      {
8          p = busca na corda (numeroCorda) o segundo element  do Acorde (Acorde);
9          Se (p<0)
10         {
11             p = busca na corda (numeroCorda) o terceiro elemento  do acorde (Acorde);
12             Se(p<0)
13             {
14                 p = busca na corda (numeroCorda) o primeiro elemento  do ao acorde (Acorde);
15                 Se(p>=0)
16                     Seta Posição(num, p);
17             }
18             else
19                 seta Posicao(num, p);
20         }else
21            seta Posicao(num, p);
22
23         num--;
24         p= -1;
25     }
26
27     retorna p;
28 }
```

Algoritmo 7: algoritmo para definir a posição das notas de um acorde.

7. CONCLUSÃO

Como resultado deste trabalho temos a criação do software *Cifras NotePad.* Para a construção do software foi necessária uma pesquisa para o processo de construção do conhecimento, buscando uma forma viável e otimizada de representar computacionalmente entidades musicais, entidades as quais se baseiam na teoria musical que possui séculos de desenvolvimento.

Um dos resultados deste trabalho foi encontrar uma forma de representar o acorde musical. Este foi um dos grandes desafios que foi encontrado no inicio da pesquisa, juntamente com a implementação de um algoritmo capaz de gerar os diversos diagramas dos distintos acordes sem a necessidade do pré- armazenamento de imagens. Tal algoritmo também permitiu posteriormente o mapeamento para a reprodução sonora do acorde.

Devido às escolhas da forma de estruturação do software e de seu algoritmo de mapeamento, foi possível construir um software pequeno e portável que pode ser usado por diversas plataformas.

Como projetos futuros temos a adequação do visualizador de acordes para plataformas móveis como celulares incluindo *iPhones* e P*almtops*, permitir a visualização de variações do diagrama de um mesmo acorde e uso de uma heurística de aprendizagem na qual o usuário possa inserir uma nova variação ou um novo tipo de acorde no software e mesmo assim o sistema tenha autonomia para a sua representação em forma de diagrama e sonora. Outra possibilidade é o uso de inteligência artificial para que o sistema seja capaz de propor o próximo acorde da sequência harmônica, baseando se nos acordes já escolhidos através de parâmetros de escolha para a condução harmônica.

REFERÊNCIAS

Avid. **Pro-Tools** Disponível em <http://www.avid.com/US/products/family/Pro-Tools > Acesso em: 22 de junho de 2010.

Boss Roland. **Processadores de som,** Disponível em < http://www.edirol.com/index.php/en/resources-mainmenu-403/77-general-information/123-cosm-composite-object-sound-modeling>, acesso em 01 de dezembro de 2009.

Cabral, G. et al. **Da Cifra Para o Braço: Estudo dos Problemas de Execução Musical em Violão e Guitarra**, SBC&M 2001.

Chediak, A. **Dicionário de Acordes**. São Paulo: Irmãos Vitale Editores, 1984.

Chediak, A. **Harmonia e Improvisação**, São Paulo: Lumiar Editora, 1986.

Costalonga, L., Mileto, E., Vicari, R. **Cálculo de Acordes com cifras Personalizáveis**, Instituto de Informática – Universidade Federal do Rio Grande do Sul, 2003.

Finale. **Software para escrita de partituras,** Disponível em < http://www.finalemusic.com/> Acesso em: 22 de junho de 2010.

Fritsch, E. **MEPSOM - Método de Programação Sônica para Músicos**. Tese de Doutorado, Universidade Federal do Rio Grande do Sul (PPGC). Porto Alegre, 2002.

Gamma, E., Helm, R., Johnson R., vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**, Addison Wesley Longmam, 1995.

Guitar-Pro . "**Guitar-Pro**" Disponível em < http://www.guitar-pro.com/en/index.php?pg=product> Acesso em: 22 de junho de 2010.

Hurber, David Miles. **The MIDI manual.** Elsevier Science, 1999.

Idpt. "**Transistor**". Disponível em <HTTP://idpt.wordpress.com>. Acesso em: 15 de novembro de 2009.

J. Pierce. **The science of musical sound**. New York: Scientific American Books, 1983.

Roads, C. **The Computer Music Tutorial,** Massachusetts: MIT Press, 1996.

Satriani, J. **Joe Satriani,** Disponível em < http://www.satriani.com/>. Acesso em: 5 de Janeiro de 2010.

Schoenberg, A. **Harmonia**, Unesp, 1911.

Sher, C. **The New Real Book (vol. 1 e 2),** Berkeley: Sher Music, 1991.

Sound Forge(2010). "**Sound Forge**" software edição musical Disponível em < http://www.finalemusic.com/> Acesso em: 22 de junho de 2010.

West, R., Howell, P., & Cross, I. **Musical Structure and Knowledge Representation,** In P. Howell, R. West, & I. Cross (Eds.), *Representing MusicalStructure,* London: Academic Press, 1991 .

APENDICE A – DIAGRAMAS UML

Diagrama de casos de uso

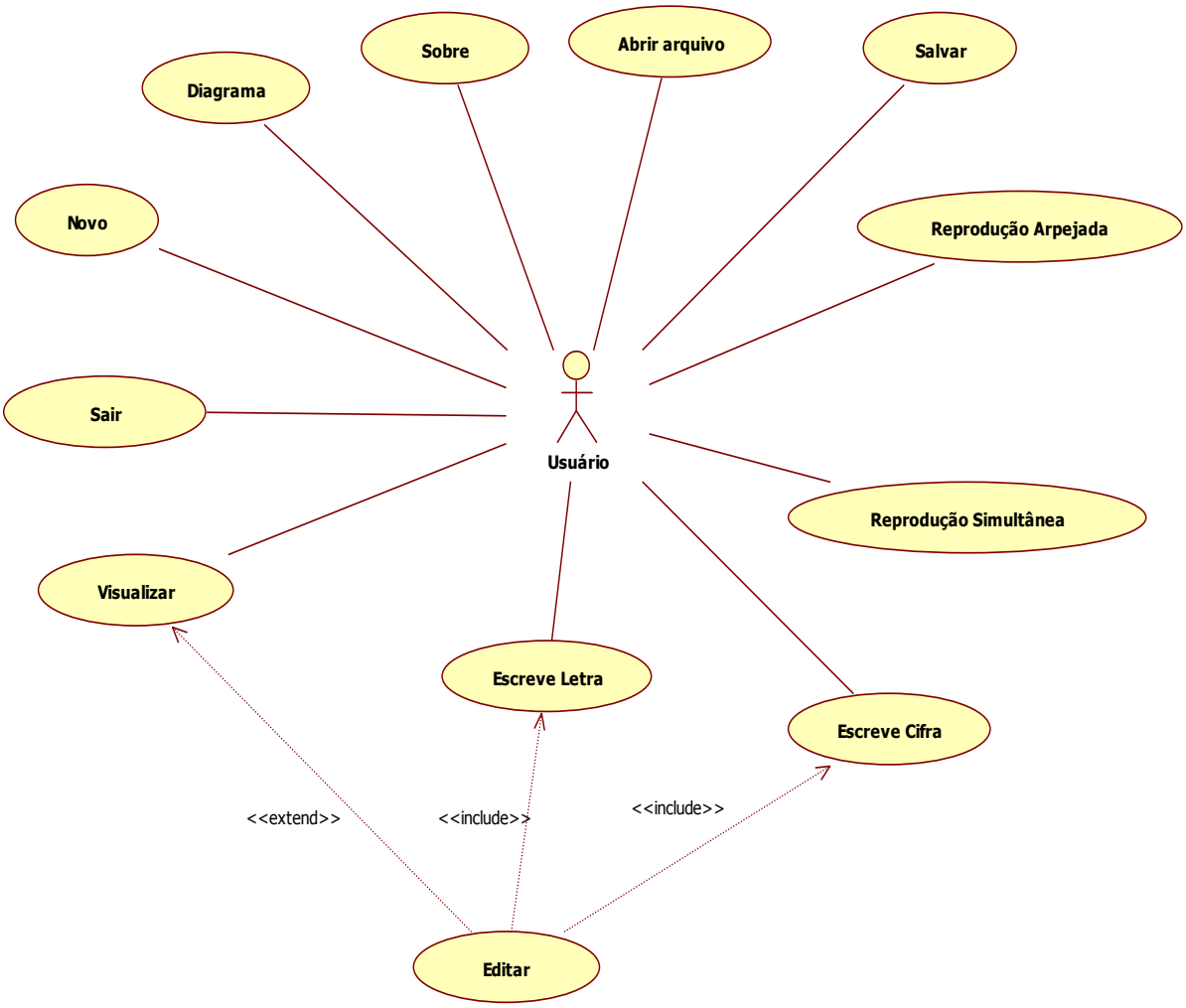


Diagrama de Classes

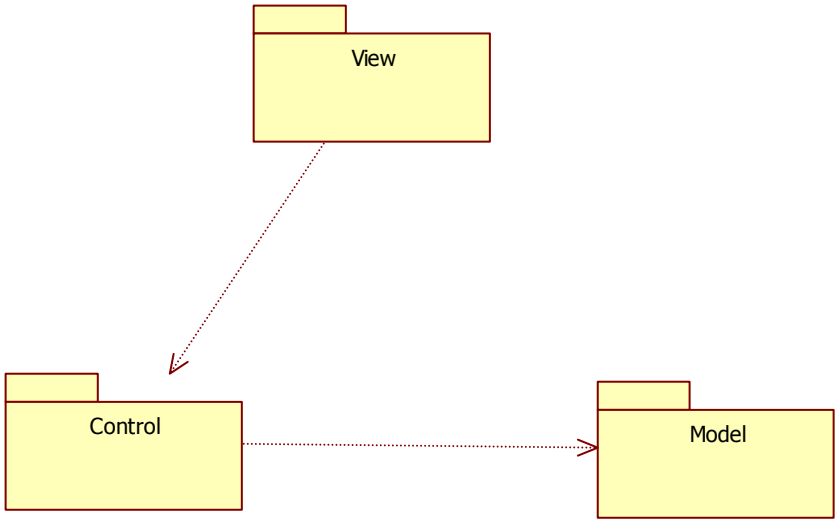
Para facilitar a Visualização os relacionamentos entre as classes serão representados usando o modelo e posteriormente será representada a classe com seus atributos e operações.

Pacotes do Sistema

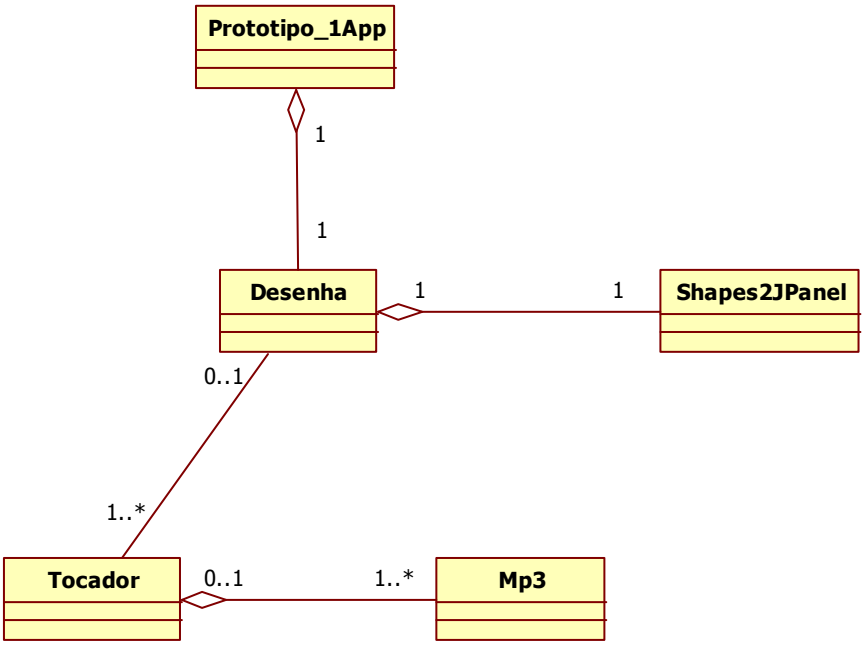
View – Camada de interface

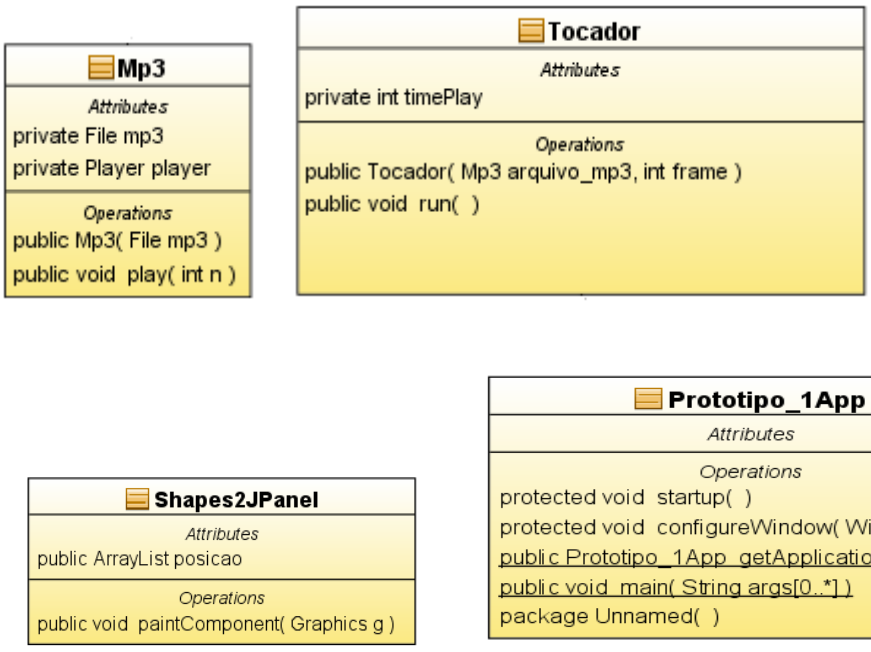
Control – camada de controle

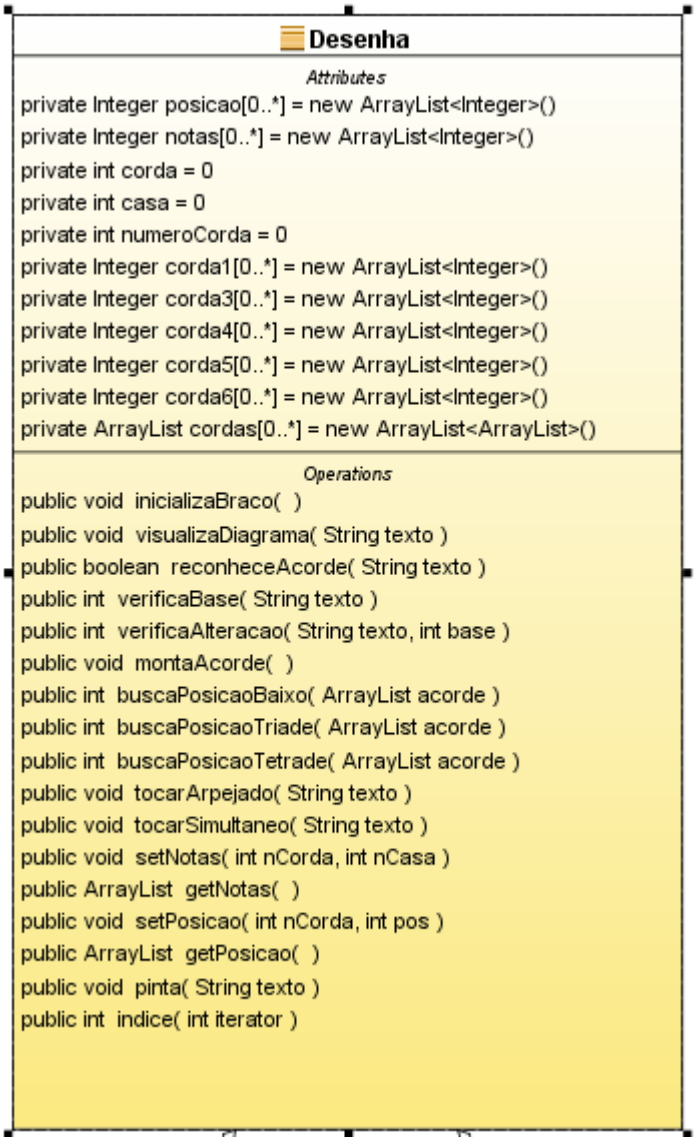
Model – camada de modelo



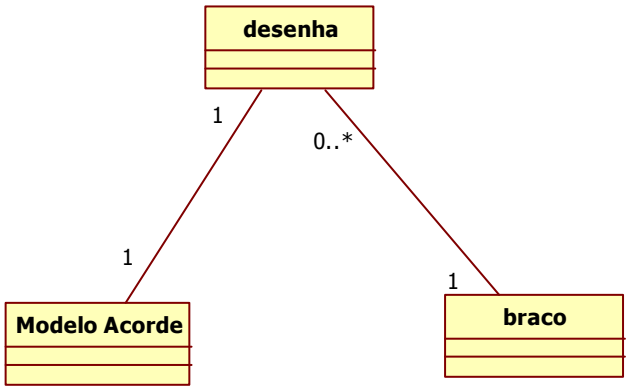
Camada de Controle

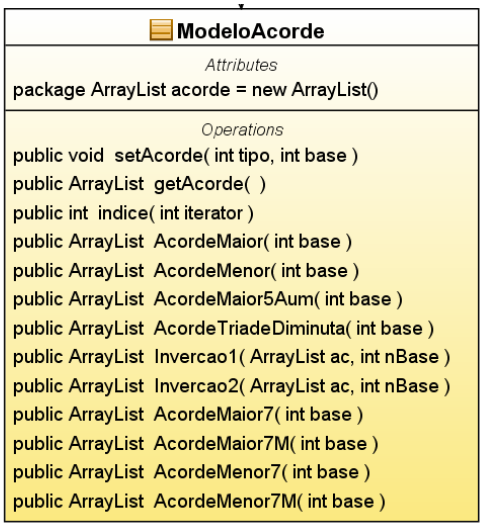


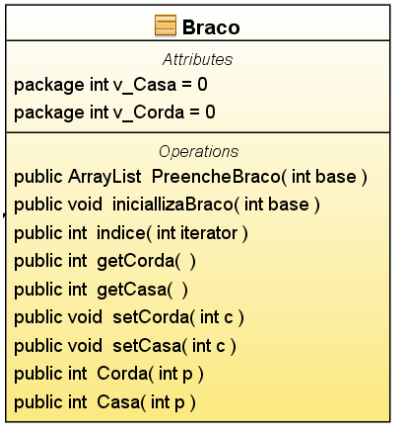



Desenha
Attributes
private Integer posicao[0..*] = new ArrayList<Integer>()
private Integer notas[0..*] = new ArrayList<Integer>()
private int corda = 0
private int casa = 0
private int numeroCorda = 0
private Integer corda1[0..*] = new ArrayList<Integer>()
private Integer corda3[0..*] = new ArrayList<Integer>()
private Integer corda4[0..*] = new ArrayList<Integer>()
private Integer corda5[0..*] = new ArrayList<Integer>()
private Integer corda6[0..*] = new ArrayList<Integer>()
private ArrayList cordas[0..*] = new ArrayList<ArrayList>()
Operations
public void inicializaBraco( )
public void visualizaDiagrama(String texto)
public boolean reconheceAcorde(String texto)
public int verificaBase(String texto)
public int verificaAlteracao(String texto, int base)
public void montaAcorde( )
public int buscaPosicaoBaixo(ArrayList acorde)
public int buscaPosicaoTriade(ArrayList acorde)
public int buscaPosicaoTetrade(ArrayList acorde)
public void tocarArpejado(String texto)
public void tocarSimultaneo(String texto)
public void setNotas(int nCorda, int nCasa)
public ArrayList getNotas( )
public void setPosicao(int nCorda, int pos)
public ArrayList getPosicao( )
public void pinta(String texto)
public int indice(int iterator)

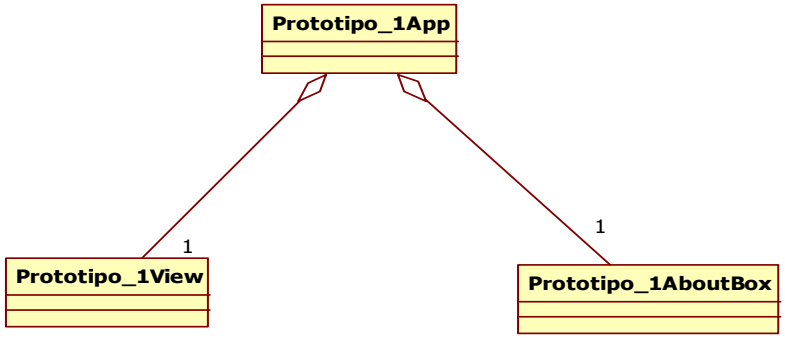
Camada de Modelo

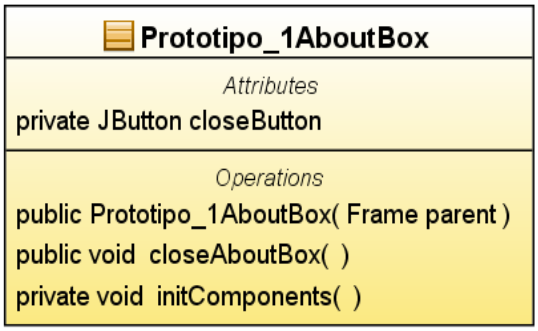


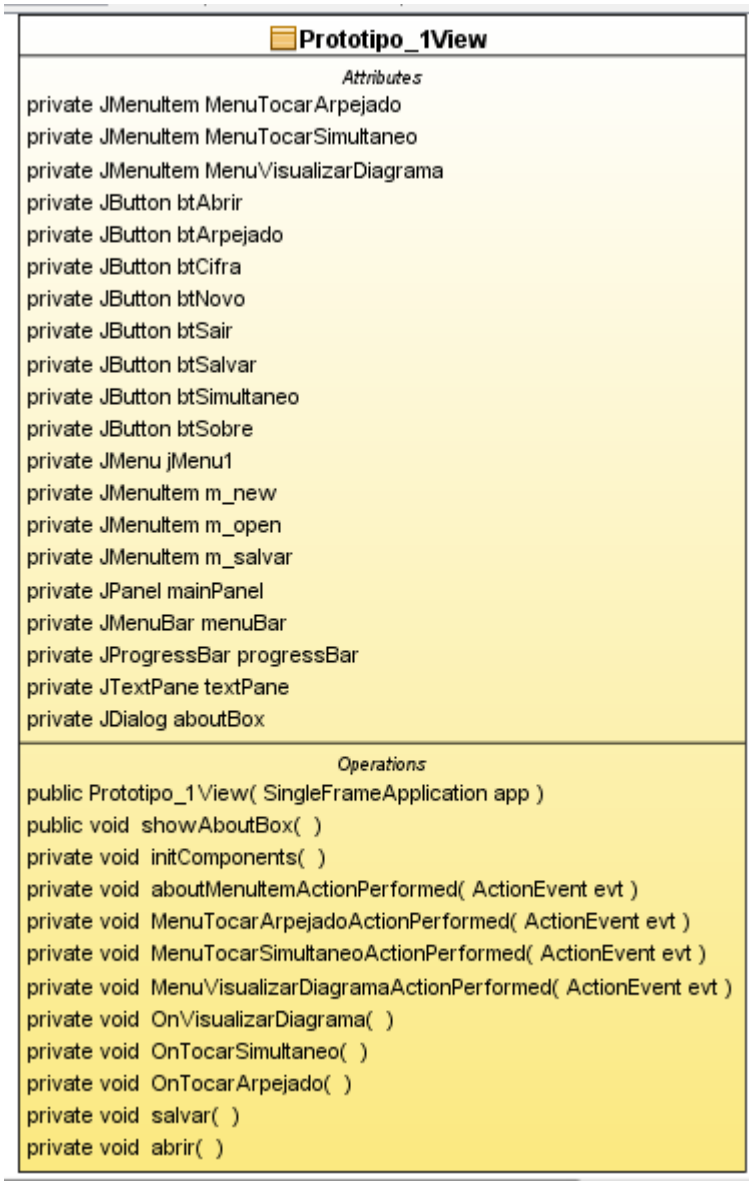




Camada de interface






Prototipo_1View
Attributes
private JMenuItem MenuTocarArpejado
private JMenuItem MenuTocarSimultaneo
private JMenuItem MenuVisualizarDiagrama
private JButton btAbrir
private JButton btArpejado
private JButton btCifra
private JButton btNovo
private JButton btSair
private JButton btSalvar
private JButton btSimultaneo
private JButton btSobre
private JMenu jMenu1
private JMenuItem m_new
private JMenuItem m_open
private JMenuItem m_salvar
private JPanel mainPanel
private JMenuBar menuBar
private JProgressBar progressBar
private JTextPane textPane
private JDialog aboutBox
Operations
public Prototipo_1View(SingleFrameApplication app)
public void showAboutBox( )
private void initComponents( )
private void aboutMenuItemActionPerformed(ActionEvent evt)
private void MenuTocarArpejadoActionPerformed(ActionEvent evt)
private void MenuTocarSimultaneoActionPerformed(ActionEvent evt)
private void MenuVisualizarDiagramaActionPerformed(ActionEvent evt)
private void OnVisualizarDiagrama( )
private void OnTocarSimultaneo( )
private void OnTocarArpejado( )
private void salvar( )
private void abrir( )

APÊNDICE – B

Código fonte:

```
import java.util.ArrayList;

public class Braco {
   int v_Casa= 0;
   int v_Corda = 0;


// este metodo é usado para inicializar as cordas com as suas devidas
notas
public ArrayList PreencheBraco(int base)
{
      ArrayList corda = new ArrayList();
      for (int i = 0; i<4; i++)
      {
            corda.add(indice(base + i));
      }
      return corda;
}


public int indice (int iterator)
{
      int i = iterator;
      if (i > 11){
            i = i - 12;
}
      return i;
}


// metodos para manipular os pontos no braço do instrumento
// usados para o desenho do diagrama
public int getCorda()
{
      return v_Corda;
}

public int getCasa()
{
      return v_Casa;
}

public void setCorda(int c)
{
      v_Corda = Corda(c);
}

public void setCasa(int c)
{
      v_Casa = Casa(c);
}


// o ponto 295 é o ponto inicial vai decrementando, pois esta
aproximando da origem
public int Corda (int p)
```

```
{
        if(p ==0)
                p = 295;
        else
                p = 295 - ((p - 1) * 50);
        return p;
}


// tratamento especial para a casa zero
public int Casa (int p)
{
        if (p == 0)
                p = 37;
        else
                p = 95 + ((p-1) *100);
        return p;
}
}
```

```
import java.util.ArrayList;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.io.File;


public class Desenha
{


   // inicializações
   Braco braco = new Braco();
   ModeloAcorde M_acorde = new ModeloAcorde();
   public ArrayList<Integer> posicao = new ArrayList<Integer>();
   ArrayList<Integer> notas = new ArrayList<Integer>();
   int corda = 0;
   int casa = 0;
   int numeroCorda = 0;
   String texto;
   int base = -1;
   int tipo = -1;
   boolean alterado = false;
   // variavel que define frames de reprodução de audio
   static int n = 57;
   // cordas do Violão
   ArrayList <Integer> corda1 = new ArrayList<Integer>();
   ArrayList <Integer> corda2 = new ArrayList<Integer>();
   ArrayList <Integer> corda3 = new ArrayList<Integer>();
   ArrayList<Integer> corda4 = new ArrayList<Integer>();
   ArrayList<Integer> corda5 = new ArrayList<Integer>();
   ArrayList<Integer> corda6 = new ArrayList<Integer>();



// estrutura que ira armazenar todas as cordas

ArrayList<ArrayList> cordas = new ArrayList<ArrayList>();

//inicializando as cordas com seus devidos valores

public void inicializaBraco()
{
   //inicializando as cordas com seus devidos valores
   // de acordo com a nota inicial de cada corda
   corda6 = braco.PreencheBraco(4);
   corda5 = braco.PreencheBraco(9);
   corda4 = braco.PreencheBraco(2);
   corda3 = braco.PreencheBraco(7);
   corda2 = braco.PreencheBraco(11);
   corda1 = braco.PreencheBraco(4);

   cordas.add(null);
   cordas.add(corda1);
   cordas.add(corda2);
   cordas.add(corda3);
   cordas.add(corda4);
   cordas.add(corda5);
   cordas.add(corda6);
}

public void visualizaDiagrama(String texto)
{
```

```
        if(reconheceAcorde(texto))
        pinta(texto);
        else
        {
                texto = JOptionPane.showInputDialog(null," Acorde Invalido,
                digite novamente: ",
                "Visualizador de Diagramas ",JOptionPane.PLAIN_MESSAGE);
                visualizaDiagrama(texto);
        }
}

// verifica a alteração, o tipo do acorde e a inversão
public boolean reconheceAcorde(String texto)
{
   base = verificaBase(texto);
   char tom = texto.charAt(0);
   String natureza = texto.substring(1); // o reto da string
   if (texto.length() > 1)
   {
      verificaAlteracao(texto);
      if (alterado)
   {
      natureza = texto.substring(2);
    }


if (natureza.equals("")) {
tipo = 1;
}
if (natureza.equals("m")) {
tipo = 2;
}
if (natureza.equals("aum") || (natureza.equals("5#"))) {
tipo = 3;
}
if (natureza.equals("°")) {
tipo = 4;
}
if (natureza.equals("7M")) {
tipo = 5;
}
if (natureza.equals("7")) {
tipo = 6;
}
if (natureza.equals("m7")) {
tipo = 7;
}
if (natureza.equals("m7M")) {
tipo = 8;
}
if ((tipo  != -1) && (base  != -1)) {
M_acorde.setAcorde(tipo, base);
montaAcorde();
return true;
} else
return false;
}


// verifica a tonalidade do acorde
```

```
public int verificaBase(String texto) {
char tom = texto.charAt(0);
switch (tom) {
    case 'C':
    base = 0;
    break;
    case 'D':
    base = 2;
    break;
    case 'E':
    base = 4;
    break;
    case 'F':
    base = 5;
    break;
    case 'G':
    base = 7;
    break;
    case 'A':
    base = 9;
    break;
    case 'B':
    base = 11;
    break;
    }
return base;
}


public void verificaAlteracao(String texto) {
char tom = texto.charAt(1);
switch (tom) {
    case '#': // verifica se o acorde é sustenido
    base = base + 1;
    alterado = true;
    break;
    case 'b': // verifica se o acorde é bemol
    base = base - 1;
    alterado = true;
    }
}




public void montaAcorde() {
    inicializaBraco();
    ArrayList v_acorde = new ArrayList();
    v_acorde = M_acorde.getAcorde();
    int baixo = buscaPosicaoBaixo(v_acorde);
    setPosicao(++numeroCorda, baixo);
    if (v_acorde.size() < 4) {
        buscaPosicaoTriade(v_acorde);
    } else {
        buscaPosicaoTetrade(v_acorde);
    }

}

// metodo que vai buscar uma determinada nota em uma corda
```

```
public int buscaPosicaoBaixo(ArrayList acorde) {
int p = -1;
numeroCorda = 6; // como esta armazenado em cordas começa do zero logo
// o 5 indica a corda 6
while (p < 0) {
    p = cordas.get(numeroCorda).indexOf(acorde.get(0));
    numeroCorda--;
}
return p;
}

// desenha os pontos de uma triade
public int buscaPosicaoTriade(ArrayList acorde) {
int p = -1;
int num = --numeroCorda;
for (int i = numeroCorda; i > 0; i--)
{
    p = cordas.get(num).indexOf(acorde.get(1));
    if (p < 0)
    {
        p = cordas.get(num).indexOf(acorde.get(2));
        if (p < 0) {
            p = cordas.get(num).indexOf(acorde.get(0));
            if (p >= 0) {
                setPosicao(num, p);
            }
        } else {
            setPosicao(num, p);
                }
    } else {
        setPosicao(num, p);
}
    num--;
    p = -1;
}
    return p;
}

// monta um acorde tetrade
public int buscaPosicaoTetrade(ArrayList acorde) {
int p = -1;
int num = --numeroCorda;
for (int i = numeroCorda; i > 0; i--) {
    p = cordas.get(num).indexOf(acorde.get(1));

    if (p < 0) {
        p = cordas.get(num).indexOf(acorde.get(2));
        if (p < 0) {
            p = cordas.get(num).indexOf(acorde.get(3));
            if (p >= 0) {
                setPosicao(num, p);
            }
        } else
            {
            setPosicao(num, p);
             }
    } else
        {
        setPosicao(num, p);
        }
    num--;
```

```
    p = -1;
    }
return p;
}


public void tocarArpejado(String texto)
{
    if(!reconheceAcorde(texto))
    {
        texto = JOptionPane.showInputDialog(null," Acorde Invalido, digite
        novamente: ",
        "Reprodução arpejada ",JOptionPane.PLAIN_MESSAGE);
        tocarArpejado(texto);
    }

    ArrayList<Integer> posicaoNotas = getNotas();

    //String com o caminho do arquivo MP3 a ser tocado
    String path1 = "C:/teste/Notas_proj/" + posicaoNotas.get(0) + ".mp3";
    String path2 = "C:/teste/Notas_proj/" + posicaoNotas.get(1) + ".mp3";
    String path3 = "C:/teste/Notas_proj/" + posicaoNotas.get(2) + ".mp3";
    String path4 = "C:/teste/Notas_proj/" + posicaoNotas.get(3) + ".mp3";
    String path5 = "C:/teste/Notas_proj/" + posicaoNotas.get(4) + ".mp3";
    String path6 = "C:/teste/Notas_proj/" + posicaoNotas.get(5) + ".mp3";

    //Instanciação de um objeto File com o arquivo MP3
    File mp3File1 = new File(path1);
    File mp3File2 = new File(path2);
    File mp3File3 = new File(path3);
    File mp3File4 = new File(path4);
    File mp3File5 = new File(path5);
    File mp3File6 = new File(path6);

    //instancia do Objeto MP3, a qual criamos a classe.
    Mp3 musica1 = new Mp3(mp3File1);
    Mp3 musica2 = new Mp3(mp3File2);
    Mp3 musica3 = new Mp3(mp3File3);
    Mp3 musica4 = new Mp3(mp3File4);
    Mp3 musica5 = new Mp3(mp3File5);
    Mp3 musica6 = new Mp3(mp3File6);
    musica1.play(n);
    musica2.play(n);
    musica3.play(n);
    musica4.play(n);
    musica5.play(n);
    musica6.play(n);
}



public void tocarSimultaneo(String texto)
{
    if(!reconheceAcorde(texto))
    {
    texto = JOptionPane.showInputDialog(null," Acorde Invalido, digite
    novamente: ",
    "Reprodução simultânea ",JOptionPane.PLAIN_MESSAGE);
    tocarSimultaneo(texto);
    }

ArrayList<Integer> posicaoNotas = getNotas();
```

```
//String com o caminho do arquivo MP3 a ser tocado
String path1 = "C:/teste/Notas_proj/" + posicaoNotas.get(0) + ".mp3";
String path2 = "C:/teste/Notas_proj/" + posicaoNotas.get(1) + ".mp3";
String path3 = "C:/teste/Notas_proj/" + posicaoNotas.get(2) + ".mp3";
String path4 = "C:/teste/Notas_proj/" + posicaoNotas.get(3) + ".mp3";
String path5 = "C:/teste/Notas_proj/" + posicaoNotas.get(4) + ".mp3";
String path6 = "C:/teste/Notas_proj/" + posicaoNotas.get(5) + ".mp3";

//Instanciação de um objeto File com o arquivo MP3
File mp3File1 = new File(path1);
File mp3File2 = new File(path2);
File mp3File3 = new File(path3);
File mp3File4 = new File(path4);
File mp3File5 = new File(path5);
File mp3File6 = new File(path6);

//instancia do Objeto MP3, a qual criamos a classe.
Mp3 musica1 = new Mp3(mp3File1);
Mp3 musica2 = new Mp3(mp3File2);
Mp3 musica3 = new Mp3(mp3File3);
Mp3 musica4 = new Mp3(mp3File4);
Mp3 musica5 = new Mp3(mp3File5);
Mp3 musica6 = new Mp3(mp3File6);

// preparando para a reprodução simultanea
Tocador faixa1 = new Tocador(musica1, n);
Tocador faixa2 = new Tocador(musica2, n);
Tocador faixa3 = new Tocador(musica3, n);
Tocador faixa4 = new Tocador(musica4, n);
Tocador faixa5 = new Tocador(musica5, n);
Tocador faixa6 = new Tocador(musica6, n);

// Trheads para que as notas sejam tocadas simultaneamente
faixa1.start();
faixa2.start();
faixa3.start();
faixa4.start();
faixa5.start();
faixa6.start();
}
// notas que serão usadas para a reprudução sonora
public void setNotas(int nCorda,int nCasa)
{
   notas.add(nCorda);
   notas.add(nCasa);
}

public ArrayList getNotas()
{
   ArrayList<Integer> retorno = new ArrayList<Integer>();
   for(int i = 0 ; i<(notas.size()); i= i+2)
   {
      int valor = (notas.get(i)*10);
      valor = valor + notas.get(i+1);
      retorno.add(valor);
      System.out.println(retorno.toString());
   }
if(retorno.size() == 4)
{
   // o violão possui 6 cordas mas nem sempre todas são tocadas
   retorno.add(00);
   retorno.add(00);
```

```
}else if(retorno.size() == 5)
    retorno.add(00);
    return retorno;
}

// seta a corde e a posicçao do dedo
public void setPosicao(int nCorda, int pos)
{
    setNotas(nCorda, pos);
    braco.setCasa(pos);
    braco.setCorda(nCorda);
    posicao.add(braco.getCorda());
    posicao.add(braco.getCasa());
}


//retorna um ArrayList com as posicoes corda e casa a ser tocada
public ArrayList getPosicao()
{
    return posicao;
}

public void pinta(String texto)
{
    // reconheceAcorde(texto);
    //montaAcorde(texto);
    JFrame frame = new JFrame("Diagrama do acorde "+ texto);
    Shapes2JPanel shapes2JPanel = new Shapes2JPanel();
    //shapes2JPanel.texto = texto;
    shapes2JPanel.posicao = getPosicao();
    frame.add(shapes2JPanel); // add shapes2JPanel to frame
    frame.setSize(400, 400); // set frame size
    frame.setLocationByPlatform(true);
    frame.getAlignmentY();
    frame.setVisible(true); // display frame
}


// metodo usado para simular uma lista circular com o indice das notas
public int indice(int iterator)
{
    int i = iterator;
    if (i > 11)
    {
        i = i - 12;
    }
    return i;
}

}
```

```java
import java.util.ArrayList;
/**
*
* @author Gustavo
*/
public class ModeloAcorde
{
    ArrayList acorde = new ArrayList();
    public void setAcorde (int tipo, int base){
    if (tipo == 1)
    acorde = AcordeMaior(base); // triade maior X
    if(tipo == 2)
    acorde = AcordeMenor(base); // triade menor Xm
    if(tipo ==3)
    acorde = AcordeMaior5Aum(base); //triade aumentada X5Aum
    if(tipo ==4)
    acorde = AcordeTriadeDiminuta(base); // triade diminuta
    if(tipo ==5)
    acorde = AcordeMaior7M(base); // tetrade maior 7M
    if(tipo == 6)
    acorde = AcordeMaior7(base); // tetrade maior 7
    if (tipo ==7)
    acorde = AcordeMenor7(base); // tetrade menor 7
    if (tipo ==8)
    acorde = AcordeMenor7M(base); // tetrade menor 7M
    if(tipo==9){
    acorde = AcordeMaior(base); // primeira inversao triade
    ArrayList<Integer> aux = new ArrayList<Integer>();
    aux = acorde;
    int nBase = aux.get(1);
    acorde = Invercao1(aux, nBase);
    }
    if(tipo==10){ // segunda inversao triade
       acorde = AcordeMaior(base);
       ArrayList<Integer> aux = new ArrayList<Integer>();
       aux = acorde;
       int nBase = aux.get(2);
       acorde = Invercao2(aux, nBase);
    }
}


public ArrayList getAcorde()
{
    return acorde;
}

// metodo usado para simular uma lista circular com o indice das notas
public int indice (int iterator)
{
    int i = iterator;
    if (i > 11)
    {
       i = i - 12;
    }
    return i;
}
//XXXXXXXXXXXXXXXXXXX TRIADES XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
// metodo que retorna um acorde maior
public ArrayList AcordeMaior (int base)
{
```

```
    acorde.add(base);
    acorde.add(indice((base + 4)));
    acorde.add(indice((base + 7)));
    return acorde;
}

//triade menor
public ArrayList AcordeMenor (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 3)));
    acorde.add(indice((base + 7)));
    return acorde;
}

// triade aumentada
public ArrayList AcordeMaior5Aum (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 4)));
    acorde.add(indice((base + 8)));
    return acorde;
}

// triade diminuta 3 e 5 menores
public ArrayList AcordeTriadeDiminuta (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 3)));
    acorde.add(indice((base + 6)));
    return acorde;
}


// primeira inversao triade baixo na terça
public ArrayList Invercao1 (ArrayList ac,int nBase)
{
    acorde.set(1,ac.get(0));
    acorde.set(0, nBase);
    acorde.set(2,ac.get(2));
    return acorde;
}

// segunda inversão de triade com a qinta no baixo
public ArrayList Invercao2 (ArrayList ac,int nBase)
{
    acorde.set(2,ac.get(0));
    acorde.set(0, nBase);
    acorde.set(1,ac.get(1));
    return acorde;
}



// XXXXXXXXXXXXXX TETRADES XXXXXXXXXXXXXXXXXXXXXXXXXXXx
// acorde maior com setima menor
public ArrayList AcordeMaior7 (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 4)));
    acorde.add(indice((base + 7)));
    acorde.add(indice(base+ 10));
```

```
    return acorde;
}

public ArrayList AcordeMaior7M (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 4)));
    acorde.add(indice((base + 7)));
    acorde.add(indice(base+ 11));
    return acorde;
}

public ArrayList AcordeMenor7 (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 3)));
    acorde.add(indice((base + 7)));
    acorde.add(indice(base+ 10));
    return acorde;
}

public ArrayList AcordeMenor7M (int base)
{
    acorde.add(base);
    acorde.add(indice((base + 3)));
    acorde.add(indice((base + 7)));
    acorde.add(indice(base+ 11));
    return acorde;
}
}
```

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import javazoom.jl.player.Player;


public class Mp3 {
/**
* Objeto para nosso arquivo MP3 a ser tocado
*/
private File mp3;
/**
* Objeto Player da biblioteca jLayer. Ele tocará o arquivo
* MP3
*/
private Player player;
/**
* Construtor que recebe o objeto File referenciando o arquivo
* MP3 a ser tocado e atribui ao atributo MP3 da classe.
*
* @param mp3
*/
public Mp3 (File mp3) {
this.mp3 = mp3;
}
/**
* Método que toca o MP3
*/
public void play(int n)
{
    try
    {
        FileInputStream fileInputStream = new FileInputStream(mp3);
        BufferedInputStream bis = new BufferedInputStream(fileInputStream);
        this.player = new Player(bis);
        System.out.println("Tocando!");
        this.player.play(n);
        System.out.println("Terminado!");
    }
    catch (Exception e)
    {
        System.out.println("Problema ao tocar " + mp3);
        e.printStackTrace();
    }
}
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import java.util.ArrayList;
import javax.swing.JPanel;

public class Shapes2JPanel extends JPanel
{

public ArrayList posicao;
// draw general paths

public void paintComponent( Graphics g)
{
super.paintComponent( g ); // call superclass's paintComponent
Graphics2D g2d = ( Graphics2D ) g;
// variaveis usadas para montar a grade
int y = 50;
int x = 50;
//XXXXXXXXXXXXX DESENHO DA GRADE XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
// seta a cor do objeto g2d
g2d.setPaint( Color.BLACK );
// desenha as linhas
for (int i = 0; i<6; i++)
{
   g2d.draw( new Line2D.Double( 50, y, 350, y ) );
   y= y+50;
}
// desenha as colunas
for (int i = 0; i<4; i++){
   g2d.draw( new Line2D.Double( x, 50, x, 300 ) );
   x = x+ 100;
}

// XXXXXXXXXX FIM DO DESENHO DA GRADE XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

g2d.setPaint( Color.RED );
// coluna muda X e linha muda Y
int corda; int casa;

// Esta parte pinta os dedos no braço

ArrayList<Integer> mao = new ArrayList<Integer>();
mao = posicao;
int tam= (mao.size()/2);

// desenha as posicoes dada pela array de posicoes
// no array cada indice representa corda e casa respectivamente

int iterator = 0;
for (int i = 0; i< tam; i++){
   corda = mao.get(iterator++);
   casa = mao.get(iterator++);
   g2d.fill( new Ellipse2D.Double(casa, corda, 10, 10));
}
} } // end class Shapes2JPanel
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import javazoom.jl.player.Player;
public class Som {
```

```
/**
* Objeto para nosso arquivo MP3 a ser tocado
*/
private File mp3;
/**
* Objeto Player da biblioteca jLayer. Ele tocará o arquivo
* MP3
*/
private Player player;
/**
* Construtor que recebe o objeto File referenciando o arquivo
* MP3 a ser tocado e atribui ao atributo MP3 da classe.
*
* @param mp3
*/
public Som(File mp3) {
    this.mp3 = mp3;
}
/**
* Método que toca o MP3
*/

public void play() {
try {
    FileInputStream fis = new FileInputStream(mp3);
    BufferedInputStream bis = new BufferedInputStream(fis);
    this.player = new Player(bis);
    System.out.println("Tocando!");
    this.player.play();
    System.out.println("Terminado!");
    }
catch (Exception e) {
    System.out.println("Problema ao tocar " + mp3);
    e.printStackTrace();
    }
    }
}
```

```
public class Tocador extends Thread {
private Mp3 musica;
private int timePlay;

   public Tocador (Mp3 arquivo_mp3,int frame){
   musica = arquivo_mp3;
   timePlay = frame;
   }

   public void run ()
   {
   try {
      musica.play(timePlay);
      }
   catch(Exception e ){
      System.out.print("ERRO: Não é possivel reproduzir o audio");
      }
   }
}
```