UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA TREINAMENTO AUDITIVO DE MÚSICOS EM DISPOSITIVOS MÓVEIS UTILIZANDO JME

MARCELO RICARDO KESTRING

BLUMENAU
2009

2009/1-13

MARCELO RICARDO KESTRING

PROTÓTIPO DE SOFTWARE PARA TREINAMENTO AUDITIVO DE MÚSICOS EM DISPOSITIVOS MÓVEIS UTILIZANDO JME

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo César Rodacki Gomes, Dr. - Orientador

BLUMENAU
2009

2009/1-13

PROTÓTIPO DE SOFTWARE PARA TREINAMENTO AUDITIVO DE MÚSICOS EM DISPOSITIVOS MÓVEIS UTILIZANDO JME

Por

MARCELO RICARDO KESTRING

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo César Rodacki Gomes, Dr. – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Dr. – FURB
Membro:	Prof. Adilson Vahldick, Ms. – FURB

Blumenau, 07 de julho de 2009.

Dedico este trabalho aos meus pais, Silvestre Kestring e Irene Kestring, e a todos os meus amigos que, de alguma maneira, me ajudaram na realização deste.

AGRADECIMENTOS

À minha família e aos meus amigos, pelo apoio e incentivo.

Ao meu orientador, Paulo César Rodacki Gomes, que acreditou na minha capacidade e conseqüentemente na conclusão do trabalho.

O livro é uma extensão da memória e da imaginação.

Jorge Luis Borges

RESUMO

Este trabalho tem por objetivo descrever o desenvolvimento de um protótipo de aplicação capaz de gerar exercícios interativos para o treinamento da percepção musical. Primeiramente, é apresentado um estudo sobre a teoria musical e seus conceitos, em particular o modelo inteiro de alturas musicais, além do estudo da tecnologia MIDI e da tecnologia JME. Em seguida, é apresentado o desenvolvimento de um protótipo, em forma de um estudo de caso, com sua especificação contendo o diagrama de casos de uso, de atividades, de sequência, de classes e de navegabilidade, bem como o código fonte dos principais métodos implementados em ambiente de programação Eclipse. O protótipo baseia-se apenas na parametrização das regras da teoria musical, aplicada à tecnologia MIDI para geração aleatória de exercícios de intervalos, escalas e acordes com a tecnologia JME para dispositivos móveis.

Palavras-chave: Percepção musical. Teoria musical. Modelo de inteiros. MIDI. JME. Geração aleatória de exercícios. Dispositivos móveis.

ABSTRACT

This work aims at describing the development of a prototype application able to create interactive exercises for the training of musical perception. Firstly, it is presented a study of musical theory and its concepts, in particular the integer model of musical pitches, as well as the study of MIDI and JME technologies. Next, it is shown the development of a prototype, as a case study, with its specification containing the diagram of use cases, of activities, of sequence, of classes and of navigability, as well as the source code of the main methods implemented in the programming environment Eclipse. The prototype is based only on the parameterization of the rules in music theory, applied to the MIDI technology for random generation of interval exercises, scales and chords with JME technology for mobile devices.

Key-words: Musical perception. Musical theory. Model of integers. MIDI. JME. Random generation of exercises. Mobile devices.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Na área da música é imprescindível que se treine a audição e também se estude a teoria musical. Esta junção coloca o músico em um caminho adequado no aprendizado para o exercício da musicalidade.

> A audição tem lugar quando assimilamos e compreendemos na nossa mente a música que acabamos de ouvir executar, ou que ouvimos executar num determinado momento do passado. [...] A percepção auditiva tem lugar quando ouvimos realmente um som, no momento em que ele está sendo produzido. [...] Mas só audiamos realmente um som depois de o termos auditivamente percebido. (GORDON, 2000, p. 16)

Percepção musical é a habilidade que uma pessoa, geralmente um músico, possui para reconhecer auditivamente notas, intervalos e acordes musicais. Segundo Tomedi (2002, p. 1), "a habilidade de distinguir diferentes combinações de notas musicais pode ser chamada tecnicamente de percepção musical".

Há algumas ferramentas e sistemas disponíveis para realizar o treinamento de percepção musical, mas que na realidade não são úteis quando se trata de mobilidade[1], ou seja, quando se vai a lugares como trabalho, casa de amigos, ônibus, entre outros. Isto se deve ao fato de que essas ferramentas são voltadas para computadores *desktop*.

Neste ponto é que se apresenta a proposta do desenvolvimento de uma aplicação para telefones celulares com o intuito de treinar a percepção musical do usuário, utilizando a tecnologia para dispositivos móveis Java Plataform, ou Micro Edition, ou ainda Java ME (JME) (SUN DEVELOPER NETWORK, 2008a).

No estudo musical é necessário que o musico utilize a teoria para auxiliar no processo de padronização musical, mas também para isso, é necessário que ele realize o treinamento auditivo para melhorar a percepção musical. O processo de melhorar a percepção musical nada mais é que a teoria musical associada à percepção auditiva.

> Pense na multiplicidade de benefícios se os jovens instrumentistas fossem ensinados a afinar a sua audição em vez do A ("lá") ou Bb ("si bemol") do seu instrumento. Os instrumentos são mais bem afinados através da audição do que usando auxiliares mecânicos ou electrónicos que estabeleçam um "nível específico" para cada altura[2]. (GORDON, 2000, p. 57)

Neste sentido é que se faz necessário a utilização de alguma tecnologia na qual o músico possa, preferencialmente, estar em qualquer lugar escutando, estudando e treinando a

[1] Microsoft Developer Network (2008) afirma que mobilidade seria "Tudo o que você pode operar a distância ou sem fio é considerado Mobilidade".
[2] "ALTURA é a capacidade de um som ser mais agudo ou grave que outro." (SCLIAR, 1994, p. 1)

percepção musical.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é explanar os conceitos e o desenvolvimento de um protótipo de aplicação voltada ao treinamento auditivo na música, que utiliza uma técnica matemática para parametrizar notas musicais com o objetivo de gerar perguntas, com a ajuda do som, sobre a teoria musical.

Os objetivos específicos do trabalho são:

a) tocar intervalos para que o usuário responda;
b) tocar acordes para que o usuário responda;
c) tocar arpejos para que o usuário responda;
d) receber resposta do usuário via teclado;
e) indicar se a resposta está correta.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos. O segundo capítulo contém a fundamentação teórica necessária para o entendimento do trabalho. Nele são discutidos tópicos relacionados sobre a percepção musical, teoria musical, intervalos, escalas, acordes e arpejos, modelo inteiro de alturas, MIDI, JME e MIDP. Também são comentados alguns trabalhos correlatos a esta aplicação.

O terceiro capítulo comenta sobre o desenvolvimento da aplicação, onde são explanados os requisitos principais da realização do trabalhado, a especificação contendo diagramas de casos de uso, atividades e seqüência. Também são feitos comentários sobre a implementação abrangendo as técnicas e ferramentas utilizadas, operacionalidade e por fim são comentados os resultados e discussão.

O quarto capítulo refere-se às conclusões e extensões do trabalho.

2 MÚSICA, TEORIA MUSICAL E DISPOSITIVOS MÓVEIS

Atualmente, a música em âmbito mundial chega a ter um grande espaço cultural na vida de muitas pessoas. O músico usa na música, melodias e harmonias como forma de expressão, criando sons musicais combinados e sons musicais simultâneos.

Segundo Moretti (2003, p. 15), quando se trata de som musical, o efeito audível é definido de um ou vários movimentos de corpos vibratórios, que produzem o som. Tendo fontes sonoras pode-se produzir o som.

A vibração produzida tem diferentes características que definem as propriedades físicas do som. São chamadas de: altura, intensidade e timbre.

2.1 PERCEPÇÃO MUSICAL

A percepção musical é a junção de dois elementos básicos: a percepção auditiva e a audição.

Segundo Gordon (2000, p. 16), na percepção auditiva lida-se com acontecimentos sonoros imediatos. Já na audição, porém, lida-se com acontecimentos musicais que podem não estar acontecendo, ou seja, estar na imaginação. A audição não deve ser confundida com o que normalmente chama-se de imagística musical, que sugere apenas a imagem vívida ou figurativa do que o som pode representar. A audição é um processo bem mais profundo e requer o entendimento e compreensão do som musical em si, tendo como característica acontecimentos sonoros ocorridos no passado.

O quadro 1 é uma adaptação da classificação da audição musical segundo Gordon (2000, p. 34).

ESTÁDIO 1	retenção momentânea
ESTÁDIO 2	imitação e audição de padrões tonais, e reconhecimento e identificação de um centro tonal
ESTÁDIO 3	estabelecimento da tonalidade e da métrica, objetiva ou subjetiva
ESTÁDIO 4	retenção, pela audição, dos padrões tonais organizados
ESTÁDIO 5	relembrança dos padrões tonais organizados e audiados noutras peças musicais
ESTÁDIO 6	antecipação e predição de padrões tonais

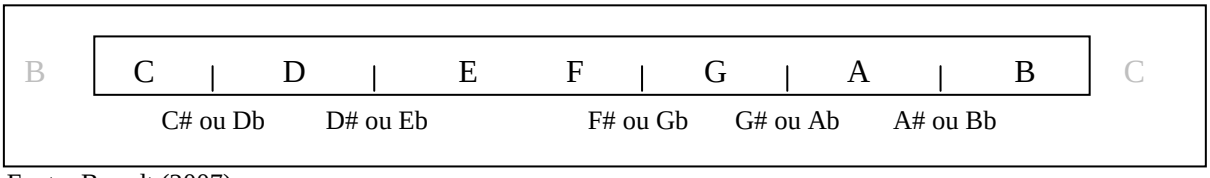
Fonte: Gordon (2000, p. 34).

Quadro 1 – Estádios da audição

2.2 TEORIA MUSICAL: ESCALA, INTERVALO, ACORDE E ARPEJO

Tomedi (2002, p. 8) diz que embora sejam infinitos os sons na música, para representar-se a notação musical pode-se utilizar apenas sete notas naturais: C (Dó), D (Ré), E (Mi), F (Fá), G (Sol), A (Lá) e B (Si).

Há como alterar estas notas de forma ascendente ou descendente, fazendo com que estas tomem o lugar de uma de suas notas adjacentes. Utiliza-se para representar estas notas adjacentes os símbolos sustenido (#) que adiciona um deslocamento ascendente à nota natural e o bemol (b) que adiciona um deslocamento descendente à nota natural. Desta forma, tem-se as doze notas musicais existentes, sete delas naturais e cinco alteradas, conforme apresenta o quadro 2.



Fonte: Brandt (2007).

Quadro 2 – Escala temperada ocidental

Nota-se que todas as sete notas podem ser alteradas, mas duas delas, o E# e o B# possuem os mesmos sons de F e C respectivamente. Chega-se assim as doze notas musicais existentes, que compõem a escala temperada ocidental: C (Dó), C# (Dó sustenido), D (Ré), D# (Ré sustenido), E (Mi), F (Fá), F# (Fá sustenido), G (Sol), G# (Sol sustenido), A (Lá), A#

(Lá sustenido) e B (Si).

2.2.1 Intervalo

Como foi visto anteriormente, há doze notas diferentes na música tradicional que compõem a escala temperada ocidental ou escala cromática. Visto isso, após o Si vem o Dó uma oitava acima do primeiro Dó, e este ciclo continua, conforme a Figura 1.

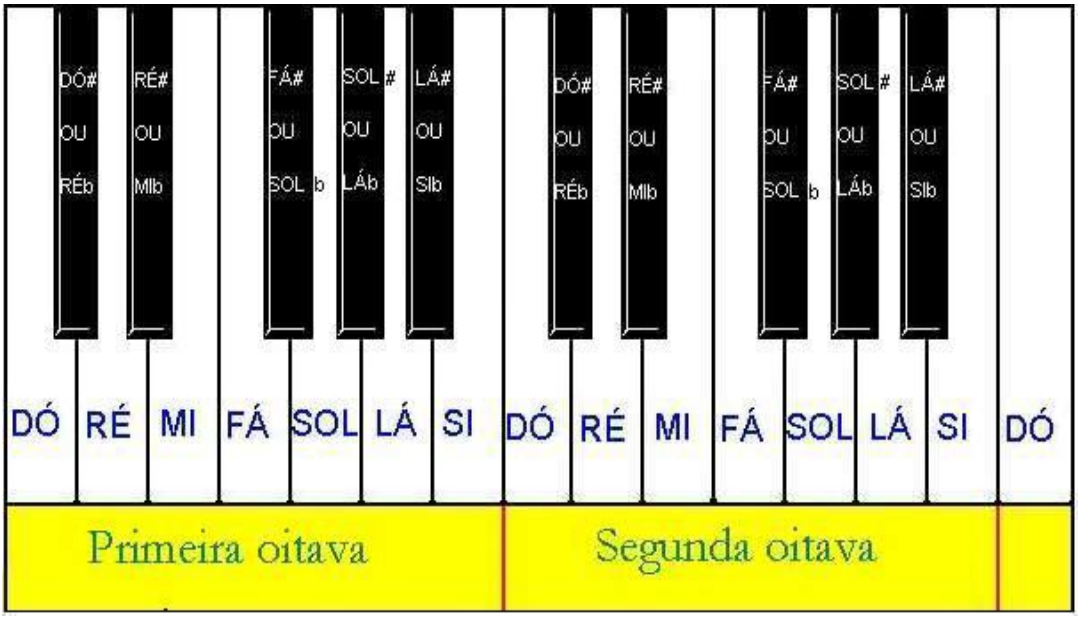


Figura 1 – Relação entre as notas musicais e suas oitavas no piano

Chediak (1986 apud TOMEDI, 2002, p. 9) diz que: intervalo é a distância (diferença de altura) entre dois tons (notas musicais). Podem ser maiores, menores, justos, aumentados e diminutos. A classificação dos intervalos é feita entre a tônica (primeira nota ou nota fundamental, da escala, ou grau I) e os demais graus da escala.

Segundo Brandt (2007), o intervalo entre duas notas (alturas) é definido pelo número de semitons entre elas. Duas notas com a distância de um semitom, como Ré e Ré Sustenido, definem uma segunda menor. As notas que estão dois semitons distantes, como Dó e Ré, definem uma segunda maior. Isto também é chamado um tom inteiro. O quadro 3 abaixo apresenta alguns intervalos.

1°	Segunda menor
2°	Segunda maior
3°	Terceira menor
4°	Terceira maior
5°	Quarta justa
6°	Tritom
7°	Quinta justa
8°	Sexta menor
9°	Sexta maior
10°	Sétima menor
11°	Sétima maior
12°	Oitava
13°	Nona Menor

Fonte: Brandt (2007).

Quadro 3 – Alguns intervalos

Brandt (2007) ainda afirma que a maioria dos intervalos pode ter outros nomes, como por exemplo, um trítono (tritom) é também chamado de quarta aumentada se a notação das notas do intervalo parecer descrever uma quarta. Um intervalo trítono de Ré a Sol sustenido, por exemplo, pode ser chamado de quarta aumentada, porque o intervalo de Ré para Sol é uma quarta justa. E se as notas do intervalo parecerem descrever uma quinta, então o intervalo trítono pode ser chamado de quinta diminuta. Um intervalo chamado aumentado é o resultado da adição em um semitom pela inclusão de um acidente (adicionar um bemol ou sustenido numa nota) em qualquer intervalo maior ou perfeito, e um intervalo chamado diminuto é o resultado da redução em um semitom pela adição de um acidente em qualquer intervalo menor ou perfeito.

2.2.2 Escala

Conforme Brandt (2007), na teoria musical, escalas são simplesmente subconjuntos da escala cromática. A maioria das escalas tem 7 notas diferentes, mas algumas têm 5, 6 ou 8, onde o tom ou nota fundamental é o primeiro grau da escala .

Brandt (2007) ainda afirma que a escala mais simples é a escala Dó Maior, que tem as notas Dó, Ré, Mi, Fá, Sol, Lá e Si, ou seja, as sete notas naturais. Uma escala maior é definida pelos intervalos ou deslocamentos entre essas notas T, T, s, T, T, T e s, onde a letra T indica um tom inteiro (ou seja dois semitons) e a letra s, um semitom. Assim, uma escala de Sol

Maior tem Sol, Lá, Si, Dó, Ré, Mi, Fá Sustenido, com um semitom levando ao sol que iniciaria a próxima oitava. Toda escala maior tem uma escala menor relativa, pois nota-se, por exemplo, que a escala de Lá Menor (Lá, Si, Dó, Ré, Mi, Fá, Sol) é formada das mesmas notas da escala de Dó Maior, mas começa no Lá, ou seja, a escala de Lá Menor é a escala relativa menor da escala de Dó Maior. Tem-se a relativa menor de qualquer escala maior quando se toca as mesmas notas a começar pela sexta nota da escala maior. O quadro 4 apresenta algumas escalas e seus intervalos.

MAIOR:	T	T	s	T	T	T	s
MENOR Natural:	T	s	T	T	s	T	T
MENOR Harmônica:	T	s	T	T	s	T+s	s
MENOR Melódica:	T	s	T	T	T	T	s

Fonte: Brandt (2007).

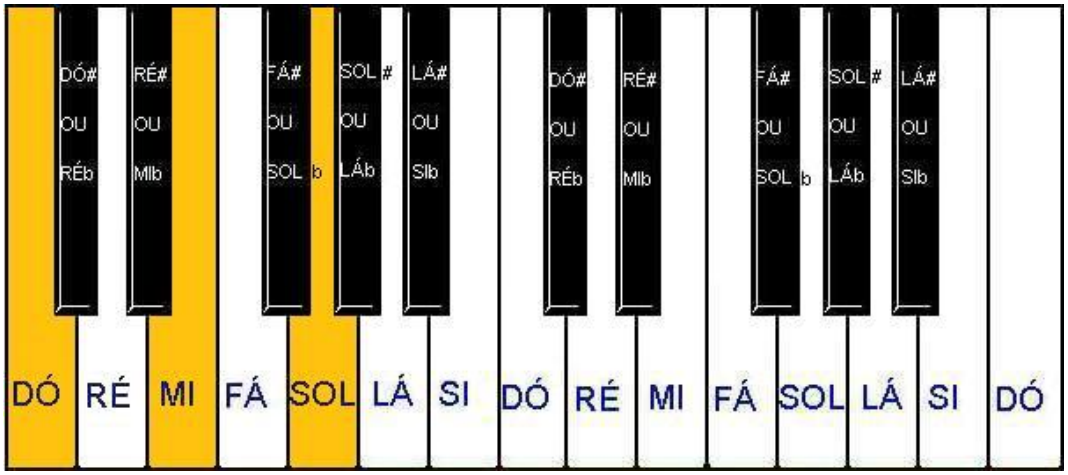
Quadro 4 – Algumas escalas e seus intervalos

2.2.3 Acorde e arpejo

Segundo Chediak (1986 apud TOMEDI, 2002, p. 10), um acorde é composto por três, quatro ou mais sons tocados simultaneamente. Quando formado por três sons é chamado de tríade, por quatro sons de tétrade e por mais de quatro sons, de tétrade com nota acrescentada.

Segundo Dias (2007), arpejo é uma palavra derivada do italiano, *arpeggio*, que quer dizer “à maneira de harpa”, consiste em tocar as notas de um acorde de maneira sucessiva (não simultânea), deixando-as soarem, como se estivesse ouvindo uma harpa em alguns momentos.

Sendo assim, percebe-se que um acorde é um conjunto de notas, geralmente tocadas ao mesmo tempo, enquanto um arpejo seria este mesmo acorde tocado como se fosse uma harpa, onde tanto o acorde quanto o arpejo, formam um relacionamento harmônico específico entre si. Segundo Brandt (2007), o acorde é construído a partir de uma única nota chamada fundamental, onde o acorde mais básico é a tríade. Uma tríade é composta por três notas, como o próprio nome leva a entender, separadas por intervalos de uma terça em relação a nota fundamental. Por exemplo, as notas Dó, Mi e Sol tocadas juntas formam a tríade de Dó Maior (C), conforme a Figura 2.



Fonte: Brandt (2007).

Figura 2 – Tríade de notas para o acorde de DÓ maior (C)

Brandt (2007) define alguns intervalos para alguns acordes como sendo:

a) acorde maior: é feito com a nota fundamental, uma terça maior e uma quinta justa a partir da fundamental;
b) acorde menor: é feito com a nota fundamental, uma terça menor e uma quinta justa a partir da fundamental;
c) acorde diminuto: é feito com a nota fundamental, uma terça menor e uma quinta diminuta a partir da fundamental;
d) acorde com quinta aumentada: é feito com a nota fundamental, uma terça maior e uma quinta aumentada a partir da fundamental;

2.3 O MODELO INTEIRO DE ALTURAS

Surge uma questão muito importante quando se trata de representação de estruturas musicais, como representá-las de forma simplificada, padronizada e de fácil construção?

Com o modelo inteiro de alturas é possível realizar esta representação, pois aplica conceitos matemáticos para relacionar de forma consistente, classificar e agrupar as notas existentes, de maneira a levar ao objetivo deste trabalho.

Segundo Rahn (1980, p. 19), por décadas teorizadores tem trabalhado na resolução do problema da parametrização das notas musicais utilizando os números inteiros para expressar a notação da altura. Levando a perceber que os números inteiros oferecem condições, no relacionamento com duas notas musicais, de descobrir qual altura é maior que a outra, ou qual

número é maior que o outro, pois em relação ao modelo musical, o modelo de inteiros é igualmente espaçado e ordenado.

2.3.1 Alturas, Classes de Altura e seus intervalos

Conforme Ranh (1980, p. 20), na música um ouvido treinado pode realizar com facilidade uma comparação entre duas alturas, ou seja, dizer se é maior, menor ou igual à outra. Se forem utilizados os números inteiros para parametrizar este relacionamento percebe-se que por outro lado as alturas também são igualmente transcritas para qualquer sistema tradicional de 12 alturas por oitava, ou seja, o intervalo entre duas notas adjacentes é uma constante (exatamente igual). Um sistema tradicional de 12 notas segue um ciclo infinito crescente ou decrescente de oitavas e a parametrização de estruturas musicais utilizando números inteiros atende os mesmos aspectos, tornando-os equivalentes para a representação de estruturas musicais. Sendo assim, subentende-se que uma unidade dos (números inteiros) é equivalente a um semitom (sistema de alturas). O resultado entre o relacionamento entre o modelo de inteiros e as alturas é uma perfeita relação entre cadeias de inteiros e cadeias de alturas (escala cromática). Com estas informações é possível numerar as alturas partindo de um numero e uma altura como referência inicial. No piano, por exemplo, pode-se assumir as alturas do inicio ao fim indo de 0 até 88, ou então convencionar a origem como sendo o C4, o DÓ central do piano na 4ª oitava, indo de -44 até o 44, melhorando a representação do relacionamento. O quadro 5 apresenta o relacionamento entre os números inteiros e as alturas.

...	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	...
...	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	...

Fonte: Rahn (1980, p. 20).

Quadro 5 – Relação entre notas e números

A distância em número de semitons entre duas alturas pode ser obtido subtraindo-se o nome de uma do nome da outra. Nota-se, no Quadro 6, que esta notação distingue intervalos crescentes e decrescentes, onde o intervalo de altura ip<x, y> define um intervalo ordenado entre duas alturas e conseqüentemente o seu sentido (RAHN, 1980, p. 21).

ip<x,y> = y – x

C4 para F4 = A<0,5> = 5 – 0 = 5

F4 para C4 = A<5,0> = 0 – 5 = -5

Fonte: Rahn (1980, p. 21).

Quadro 6 – Notação para distinção da distância e do sentido entre intervalos

Segundo Rahn (2002, p. 14), os resultados positivos significam que é um intervalo ascendente, enquanto os resultados negativos são descendentes. Mas quando é necessário definir-se uma medida de distância independente do seu sentido, ou seja, considerar apenas o deslocamento entre duas alturas, aplica-se o valor absoluto no resultado da subtração entre as duas alturas x e y, conforme o Quadro 7, definindo ip(x, y) como o intervalo desordenado entre duas alturas x e y. Lembrando que o valor absoluto transforma a diferença negativa em positiva.

ip(x,y) = |y - x|

ip(15,4) = |4 - 15| = |-11| = 11

ip(-8,-14) = |-14 - (-8)| = |-14 + 8| = |-6| = 6

ip(-13,-4) = |-4 - (-13)| = |-4 + 13| = |9| = 9

Fonte: Rahn (1980, p. 22).

Quadro 7 – Distância absoluta

Rahn (1980, p. 22) afirma que muito da teoria da música não explica sobre altura, mas sobre classes de altura. Uma classe de altura é a equivalência de altura que todas as alturas oitavadas possuem. Para ter-se uma maior precisão deste conceito, utiliza-se o modelo inteiro de altura. Assim pode-se definir que duas alturas b e c são da mesma classe de equivalência de altura se e somente se por algum inteiro n, b = 12.n + c, ou equivalente (para algum inteiro não negativo n) ip(b,c) = 12.n, conforme o exemplo do Quadro 8.

IA(1,13) = 12

IA(-4, 20) = |20 – (-4)| = 20 + 4 = 24 = 2.12

IA(46,-2) = |(-2) - 46| = |-48| = 48 = 4.12

Fonte: Rahn (1980, p. 22).

Quadro 8 – Exemplo de aplicação da fórmula de detecção de alturas na mesma classe de equivalência de altura

De acordo com o Quadro 8 acima, as alturas 1 e 13 estão na mesma classe de equivalência de altura, -4 e 20 estão em uma classe diferente, e as alturas 46 e –2 estão juntas mas em uma terceira classe de equivalência. Ranh (1980, p. 23) afirma que as alturas na mesma classe de altura são múltiplos com um deslocamento entre o anterior e o próximo de

12 semitons, ou seja, o intervalo de altura desordenado entre eles é 0, 12, 24, 36, etc. Pode-se simplificar definindo a classe de altura como um conjunto de todas essas alturas. Sendo assim pode-se afirmar que existem 12 classes de altura diferentes, e pode-se chamá-las pelo seu menor membro não negativo conforme mostrado no Quadro 9 a seguir.

0 =	(...	-36,	-24,	-12,	0,	12,	24,	36,	48 ...)
1 =	(...	-35,	-23,	-11,	1,	13,	25,	37,	49 ...)
2 =	(...	-34,	-22,	-10,	2,	14,	26,	38,	50 ...)
3 =	(...	-33,	-21,	-9,	3,	15,	27,	39,	51 ...)
4 =	(...	-32,	-20,	-8,	4,	16,	28,	40,	52 ...)
5 =	(...	-31,	-19,	-7,	5,	17,	29,	41,	53 ...)
6 =	(...	-30,	-18,	-6,	6,	18,	30,	42,	54 ...)
7 =	(...	-29,	-17,	-5,	7,	19,	31,	43,	55 ...)
8 =	(...	-28,	-16,	-4,	8,	20,	32,	44,	56 ...)
9 =	(...	-27,	-15,	-3,	9,	21,	33,	45,	57 ...)
10 =	(...	-26,	-14,	-2,	10,	22,	34,	46,	58 ...)
11 =	(...	-25,	-13,	-1,	11,	23,	35,	47,	59 ...)

Fonte: Rahn (1980, p. 23).

Quadro 9 – As 12 classes de altura

Segundo Rahn (1980, p. 23), com o objetivo de alcançar um critério que reduzisse o processo de verificação da equivalência de classes de altura, matemáticos desenvolveram um sistema aritmético de módulo, onde a palavra módulo é abreviada para mod. A matemática define esta redução de processo através do módulo da seguinte forma, dois inteiros b e c são equivalentes mod 12 se e somente se b = 12.n + c para qualquer inteiro n. Esta foi exatamente a fórmula encontrada nas definições prévias de classes de intervalo. Convencionando-se o Dó central do piano como sendo a altura 0, obtêm-se as alturas para as classes de altura no Quadro 9 aplicando-se a fórmula mod 12 alterando ascendentemente ou descendentemente ao infinito o número inteiro n, definindo desta forma as classes de intervalo.

2.3.2 Intervalo de classe de altura ordenado e desordenado

Segundo Ranh (1980, p. 25), o intervalo ordenado entre duas classes de altura é uma sequência crescente ou decrescente. Tomando-se duas classes de altura, uma das duas terá uma altura que é maior do que qualquer altura da outra classe. Por exemplo, pegando-se um DÓ, algum Dó sustenido será maior. A definição de intervalo de classe de altura ordenado pode ser similar a definição de intervalo de altura ordenada, onde defini-se que para quaisquer duas classes de altura a e b, o intervalo de classe de altura ordenado entre x e y é igual a y – x

mod 12, originando a fórmula i<a,b> = (b – a) mod 12. Por exemplo, i<5,1> = 1 – 5 = (-4) = 8, mas i<1,5> = 5 – 1 = 4. O Quadro 10 a seguir apresenta uma ajuda no cálculo de intervalos convertidos para seus menores equivalentes não negativos, mod 12, e o Quadro 11 apresenta exemplos de aplicação da fórmula para intervalos de classe de altura ordenados convencionando que a nota Dó é origem zero (0).

-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
11	10	9	8	7	6	5	4	3	2	1

Fonte: Rahn (1980, p. 25).

Quadro 10 - Menores equivalentes não negativos mod 12

C para B = i<C,A> = i<0,9> = 9 – 0 = 9

A# para C = i<B,C> = i<10,0> = 0 – 10 = -10 = 2

E para G# = i<E,G#> = i<4,8> = 2 – 10 = -8 = 4

F# para D# = i<F#,D#> = i<6,3> = 3 – 6 = -3 = 9

Fonte: Ranh (1980, p. 25).

Quadro 11 – Alguns exemplos de menores equivalentes não negativos mod 12

Num conjunto com classe de altura {0,7}, por exemplo, não é possível determinar se a altura 0 vem primeiro que a altura 7 ou vice-versa. Neste caso, os dois intervalos ordenados possíveis são <0,7> = 7 e <7,0> = 5. Assim a definição de intervalo de classe de altura desordenado ou i(a,b) é necessária para não tornar o intervalo entre classes de altura em um conjunto ambivalente. Pode-se definir intervalo de classe de altura desordenado i(x,y) como sendo igual ao menor resultado entre i<x,y> e i<y,x> (RAHN, 1980, p. 28).

Para Rahn (1980, p. 28), convenciona-se o menor valor possível da classe de altura ordenada como sendo o resultado da classe de altura desordenada, onde os valores possíveis para estas classes de altura desordenada são 0, 1, 2, 3, 4, 5 e 6. Se dois números quaisquer forem adicionados e resultarem em 12, ou 0 mod 12, pode-se chama-los de complementares mod 12. Visto o Quadro 12 que mostra o relacionamento entre os números mod 12 e os complementares mod 12, percebe-se que o menor resultado entre i<a,b> e i<b,a> é sempre o menor de dois complementos mod 12, como mostrado no quadro 13.

Inteiros mod 12:	0	1	2	3	4	5	6	7	8	9	10	11
Complementos de mod 12:	0	11	10	9	8	7	6	5	4	3	2	1

Fonte: Rahn (1980, p. 28).

Quadro 12 – Complementares Mod 12

		(para x <= y)
0 1 2 3 4 5 6	i(x,y) = i(y,x)	i<x,y>
0 11 10 9 8 7 6		i<y,x>

Fonte: Rahn (1980, p. 28).

Quadro 13 – Menor Complementar Mod 12

Rahn (1980, p.28) ainda afirma que o intervalo complementar mod 12 é inversamente relacionado, onde 11 (um intervalo sétima maior) é relacionado através de i(x,y) com a inversão de uma sétima menor, 1 (uma segunda menor); a classe de altura 10 é relacionada a 2; 9 ao 3; e assim por diante.

2.4 MIDI

MIDI é uma tecnologia padronizada de comunicação entre equipamentos eletrônicos e instrumentos musicais (teclados, guitarras, computadores, sintetizadores, e outros), tornando possível que a composição musical seja executada, transmitida ou manipulada por qualquer dispositivo que reconheça este padrão.

Outros formatos, como o formato *MPEG-1/2 Audio Layer 3* (MP3) e WAVE*form audio format* (WAV), diferenciam-se no fato de que um arquivo MIDI não contém o áudio propriamente dito, e sim as instruções para produzi-lo, ou seja, é praticamente uma partitura digitalizada.

> [...] a tecnologia MIDI permite que dispositivos eletrônicos (usualmente sintetizadores, mas também computadores, gravadores multipista, e até mesmo controladores de luzes para shows, videocassetes etc.) interajam e trabalhem em sincronia com outros dispositivos compatíveis com MIDI. Usando um controlador mestre, como um teclado, um instrumento pode reproduzir ou enviar sons para outro instrumento conectado remotamente. Isto elimina a necessidade de um tecladista ter nove ou dez teclados à sua volta. Ele pode reproduzir o som de todos os teclados usando um só teclado, simplesmente conectando-os via MIDI. Os outros teclados não precisam nem estar próximos, e o músico não precisa nem tocá-los com as próprias mãos, apesar de poder interagir com eles. (GONTIJO, 1998 apud TOMEDI, 2002, p. 22)

Tomedi (2002, p. 22) também diz que MIDI é um protocolo de comunicação, composto de um grande conjunto de comandos musicais, com os quais instrumentos eletrônicos controlam uns aos outros.

2.5 JME

Com o intuito de focar um segmento de tecnologia para cada mercado específico, a Sun (SUN MICROSYSTEMS, 2008) reagrupou as suas tecnologias Java em três edições: *Java Enterprise Edition* (JEE), *Java Standard Edition* (JSE) e JME.

Segundo Maciel e Pitoni (2001), JME é uma tecnologia que possibilita o desenvolvimento de software para sistemas e aplicações embarcadas, ou seja, toda aquela que roda em um dispositivo de propósito específico, desempenhando alguma tarefa que seja útil para o dispositivo. JME é a plataforma Java para dispositivos compactos, como celulares, *Personal Digital Assistants* (PDAs) e uma outra gama de dispositivos, que consiste em uma coleção de *Aplication Programming Interface* (APIs) do Java definidas através da *Java Community Proccess* (JCP).

O JME tem como objetivo dois grupos diferentes de produtos:

a) dispositivos compartilhados, fixos e conectados à informação (*information connected*): exemplos típicos são as internet TVs, telefones com internet, sistemas de navegação de carros e comunicadores *high-end*;
b) dispositivos pessoais, móveis e conectados à informação (*information connected*): *pagers*, telefones celulares e PDAs são os melhores exemplos para esta classe.

Mesmo tendo muitas coisas em comum, todos estes dispositivos também diferem na forma, função e características. Para isso existe a especificação de uma configuração mínima em termos de hardware e bibliotecas padrão para o dispositivo, onde a *Connected Device Configuration* (CDC) é prevista para a primeira categoria de produtos e a *Connected Limited Device* Configuration (CLDC) é prevista para a segunda categoria de dispositivos.

Há ainda o conceito de perfil (*profile*). Um perfil é um conjunto de bibliotecas que são muito mais específicas a uma categoria de dispositivos do que as bibliotecas disponíveis pela configuração. Perfis são implementados em conformidade com uma configuração (MACIEL; PITONI, 2001).

2.6 MIDP 2.0

O *Mobile Information Device Profile* (MIDP) é um perfil que foi adicionado à *Java*

Specifications Request (JSR) 118 e que consiste em um conjunto de classes que possibilita os desenvolvedores de software implementarem as aplicações de acordo com as características das aplicações de dispositivos computacionais simples e de pequena capacidade de processamento, como os telefones celulares. O MIDP fornece muitas funcionalidades como suporte aos protocolos de rede, reprodução de multimídia, definição de formulários e itens, APIs para jogos, suporte ao sistema de cores RGB e validação de permissões de segurança e assinaturas virtuais (JOHNSON, 2008, p. 33).

O MIDP 2.0 aprimorou as capacidades de produtos que utilizavam do perfil MIDP 1.0, adicionando várias características às APIs originais, acrescentando novas APIs de rede suportando *Transmissiont Control Protocol* (TCP) *sockets*, *User Datagram Protocol* (UDP) datagrama, serialização, iniciliazação de envio (*push-initialed*) e conexões seguras. Uma robusta API de segurança e gerenciamento e APIs para som e eventos de jogos estão sendo adicionadas também (SUN DEVELOPER NETWORK, 2008b).

Para reprodução de som, utiliza-se a MIDP 2.0 Media API que deriva da Mobile Media API (acrescentada na JSR 135). A MIDP 2.0's Media API removeu a reprodução de vídeo e outros recursos para se focar em sons básicos, incluindo MIDI e geração de tons (BARBAGALLO, 2004, p. 182). Mais especificamente utiliza-se o método `playTone` da classe `Manager` para reproduzir o som de uma nota musical, onde são passados três parâmetros:

a) o primeiro parâmetro é um número inteiro entre 0 e 127 da nota musical que se deseja tocar;
b) o segundo parâmetro é um número inteiro que representa os milissegundos para a duração da nota musical;
c) e o terceiro parâmetro é um número inteiro entre 0 e 100 para o volume da nota musical.

2.7 TRABALHOS CORRELATOS

A seguir são apresentados dois trabalhos correlatos ao trabalho proposto, sendo eles: um protótipo de um sistema para auxílio ao treinamento da percepção musical feito em Delphi para computadores *desktop* (TOMEDI, 2002); e o EARMASTER (2009), um software de treinamento auditivo de músicos já presente no mercado.

2.7.1 Protótipo de um sistema para auxílio ao treinamento da percepção musical

O trabalho desenvolvido por Tomedi (2002) foi a criação de um protótipo de um sistema para auxílio ao treinamento da percepção musical em computadores *desktop*. O seu trabalho teve por finalidade o desenvolvimento e a construção de um protótipo de sistema capaz de gerar exercícios interativos para o treinamento da percepção musical. No início, é mostrado um estudo sobre a teoria musical e seus conceitos, em particular o modelo de inteiros para classes de alturas musicais, além do estudo da tecnologia MIDI. O protótipo implementado no trabalho de Tomedi (2002) baseia-se apenas na parametrização das regras da teoria musical, aplicada à tecnologia MIDI para geração aleatória de exercícios de intervalos, escalas e acordes.

A utilização de relações pré-definidas baseadas na teoria musical para a geração de sistemas musicais é possível, pois pode-se encarar a música, de certa forma, como uma ciência exata, com atributos lógicos e aritméticos, permitindo a aplicação de relações matemáticas sobre as relações existentes na teoria musical contemporânea.

2.7.2 EARMASTER

O EARMASTER (2009) é um software que já atua no mercado desde 1994, quando a primeira versão do software de treinamento auditivo foi desenvolvida, feita para MS-DOS. Segundo a EARMASTER (2009), o treinamento auditivo é essencial para entender a música ouvida e tocada. Com o EarMaster é possível aprender a identificar, transcrever e tocar intervalos, acordes e ritmos com várias lições trazidas pelo software, desde música clássica ao *rock*, incluindo acordes e ritmos diferenciados, e ainda um modo de prática de acordes para o que você precisar.

Em 2009 foi lançada a versão mais recente de um dos seus produtos, o EarMaster Essencial 5. A EARMASTER (2009) possui três produtos no mercado: o EarMaster Essencial 5, o EarMaster Pro 5 e o EarMaster School 5. Onde o EarMaster School 5 é o produto mais completo entre eles.

3 DESENVOLVIMENTO

Este capítulo detalha as etapas do desenvolvimento do protótipo. São ilustrados os principais requisitos, a especificação, a implementação (mencionando técnicas e ferramentas utilizadas, bem como a operacionalidade do protótipo) e por fim são listados resultados e discussão.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo de aplicação proposto deverá:

a) apresentar uma interface adequada às configurações de hardware de um dispositivo móvel e também amigável ao usuário (Requisito Não Funcional – RNF);
b) gerar perguntas/questões relacionadas a teoria musical com a ajuda do som (Requisito Funcional – RF);
c) receber resposta do usuário via teclado (RNF);
d) indicar se a resposta do usuário está correta (RF);
e) ao fim de cada questão respondida, classificar o nível de percepção musical do usuário (RF);
f) ser implementado utilizando a linguagem de programação Java (RNF);
g) ser implementado utilizando a especificação MIDP 2.0 do Java para dispositivos móveis (RNF).

3.2 ESPECIFICAÇÃO

A especificação do presente trabalho foi desenvolvida utilizando a notação UML (OMG, 2009) em conjunto com a ferramenta Enterprise Architect (SPARX SYSTEMS, 2009). São explanados diagramas de casos de uso, atividades, seqüência, classes e navegabilidade. Alguns diagramas estão em sua forma resumida para melhor visualização.

A ferramenta recebeu o nome de Ear Trainer Mobile devido às suas características. A seguir são apresentados os três casos de uso da aplicação (Figura 3).

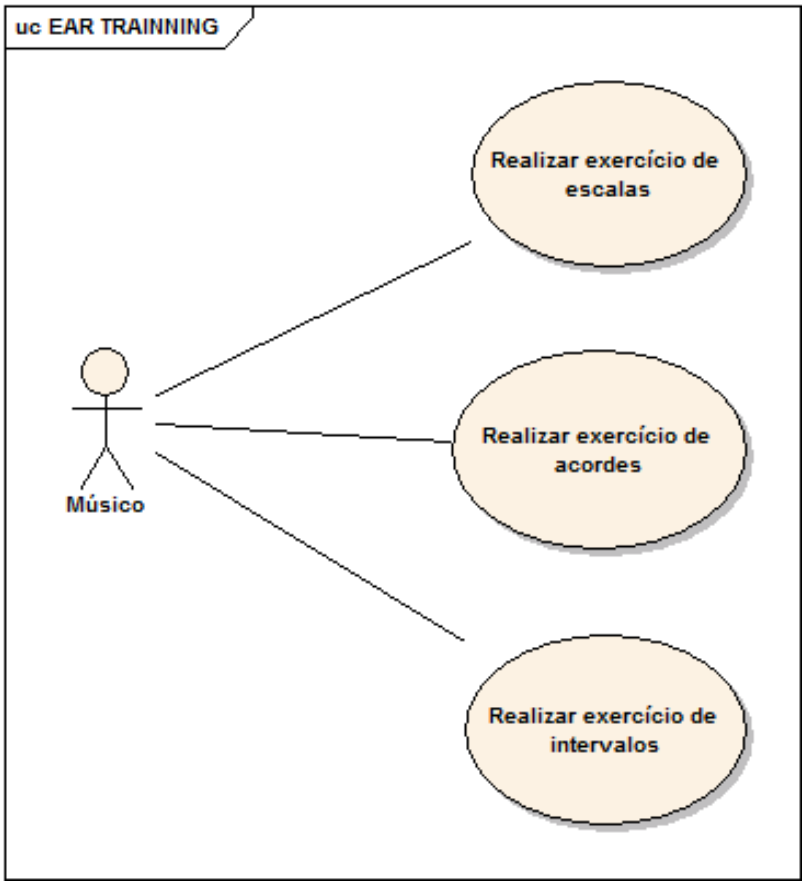


Figura 3 – Diagrama de Casos e uso

Os três casos de uso (`Realizar exercício de escalas` (Quadro 14), `Realizar exercício de acordes` (Quadro 15) e `Realizar exercício de intervalos` (Quadro 16)) apresentados no diagrama de casos de uso (Figura 3) descrevem o que o usuário faz para realizar os exercícios. Cada caso de uso possui, além do cenário principal, três fluxos alternativos responsáveis por informar o que ocorrerá caso o usuário opte por não prosseguir com o exercício e dois fluxos alternativos responsáveis por informar o que ocorrerá caso o usuário opte por tocar a estrutura musical atual novamente.

Realizar exercício de escalas: possibilita ao usuário realizar o questionário de teoria musical sobre escalas.	
Pré-condição	O usuário deve iniciar o exercício escolhendo a opção iniciar no menu inicial.
Cenário principal	1) O usuário seleciona a opção de exercício `Escalas`. 2) O sistema apresenta a tela de opções de dificuldade (Duas escalas, Três escalas e Quatro escalas). 3) O usuário seleciona a opção de dificuldade desejada. 4) O sistema apresenta a pergunta com as possíveis respostas sobre o exercício e toca o som da estrutura musical correspondente à pergunta. 5) O usuário seleciona a opção desejada. 6) O sistema apresenta uma mensagem para o resultado da questão.
Fluxo Alternativo 01	1) No passo 1, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por voltar, o sistema apresenta a tela de exercícios.
Fluxo Alternativo 03	No passo 5, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 04	No passo 5, caso o usuário opte por tocar a estrutura musical novamente, o sistema permanece na tela de pergunta atual.
Fluxo Alternativo 05	No passo 6, caso a resposta dada pelo usuário estiver errada e o usuário opte por tocar a estrutura musical novamente, o sistema reproduz o som da estrutura musical atual e permanece na tela de resultado para a pergunta atual.
Pós-condição	O sistema apresenta o número de respostas certas e erradas, e apresenta uma mensagem de como está a audição do músico para os exercícios de escalas.

Quadro 14 - Caso de uso `realizar exercício de escalas`

Realizar exercício de acordes: possibilita ao usuário realizar o questionário de teoria musical sobre acordes.	
Pré-condição	O usuário deve iniciar o exercício escolhendo a opção iniciar no menu inicial.
Cenário principal	1) O usuário seleciona a opção de exercício `Acordes`. 2) O sistema apresenta a tela de opções de dificuldade (Acordes maior/menor, Acordes simples e Acordes com inversão). 3) O usuário seleciona a opção de dificuldade desejada. 4) O sistema apresenta a pergunta com as possíveis respostas sobre o exercício e toca o som da estrutura musical correspondente à pergunta. 5) O usuário seleciona a opção desejada. 6) O sistema apresenta uma mensagem para o resultado da questão.
Fluxo Alternativo 01	No passo 1, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por voltar, o sistema apresenta a tela de exercícios.
Fluxo Alternativo 03	No passo 5, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 04	No passo 5, caso o usuário opte por tocar a estrutura musical novamente, o sistema permanece na tela de pergunta atual.
Fluxo Alternativo 05	No passo 6, caso a resposta dada pelo usuário estiver errada e o usuário opte por tocar a estrutura musical novamente, o sistema reproduz o som da estrutura musical atual e permanece na tela de resultado para a pergunta atual.
Pós-condição	O sistema apresenta o número de respostas certas e erradas, e apresenta uma mensagem de como está a audição do músico para o exercício de acordes.

Quadro 15 - Caso de uso `realizar exercício de acordes`

Realizar exercício de intervalos: possibilita ao usuário realizar o questionário de teoria musical sobre intervalos.	
Pré-condição	O usuário deve iniciar o exercício escolhendo a opção iniciar no menu inicial.
Cenário principal	1) O usuário seleciona a opção de exercício `Intervalos`. 2) O sistema apresenta a tela de opções de dificuldade (Crescente, Decrescente e Crescente/Decrescente). 3) O usuário seleciona a opção de dificuldade desejada. 4) O sistema apresenta a pergunta com as possíveis respostas sobre o exercício e toca o som da estrutura musical correspondente à pergunta. 5) O usuário seleciona a opção desejada. 6) O sistema apresenta uma mensagem para o resultado da questão.
Fluxo Alternativo 01	No passo 1, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por voltar, o sistema apresenta a tela de exercícios.
Fluxo Alternativo 03	No passo 5, caso o usuário opte por voltar, o sistema apresenta a tela de menu inicial.
Fluxo Alternativo 04	No passo 5, caso o usuário opte por tocar a estrutura musical novamente, o sistema permanece na tela de pergunta atual.
Fluxo Alternativo 05	No passo 6, caso a resposta dada pelo usuário estiver errada e o usuário opte por tocar a estrutura musical novamente, o sistema reproduz o som da estrutura musical atual e permanece na tela de resultado para a pergunta atual.
Pós-condição	O sistema apresenta o número de respostas certas e erradas, e apresenta uma mensagem de como está a audição do músico para o exercício de intervalos.

Quadro 16 - Caso de uso `realizar exercício de intervalos`

O diagrama de atividades apresentado na Figura 4, descreve os passos da execução dos exercícios da aplicação. Este diagrama de atividades indica que inicia-se exibindo a tela de opções de exercícios, e após isso, apresenta-se a tela de dificuldades. Finalmente, apresenta

uma tela com uma pergunta ao usuário, onde para cada uma destas telas o usuário deverá escolher uma das opções para ir para a tela seguinte. Após a escolha da resposta, se a mesma estiver correta, então a aplicação mostra uma tela de congratulação, e se estiver errada, revela a resposta correta. Após o resultado de cada pergunta, a aplicação vai para a atividade de teste `Ainda há perguntas?`, e se a pergunta atual for menor ou igual a 10 então é solicitada uma nova pergunta, senão apresenta uma tela com uma mensagem contendo o nível de percepção musical, e também o número de respostas certas e o número de respostas erradas, e após esta tela, finaliza o exercício.

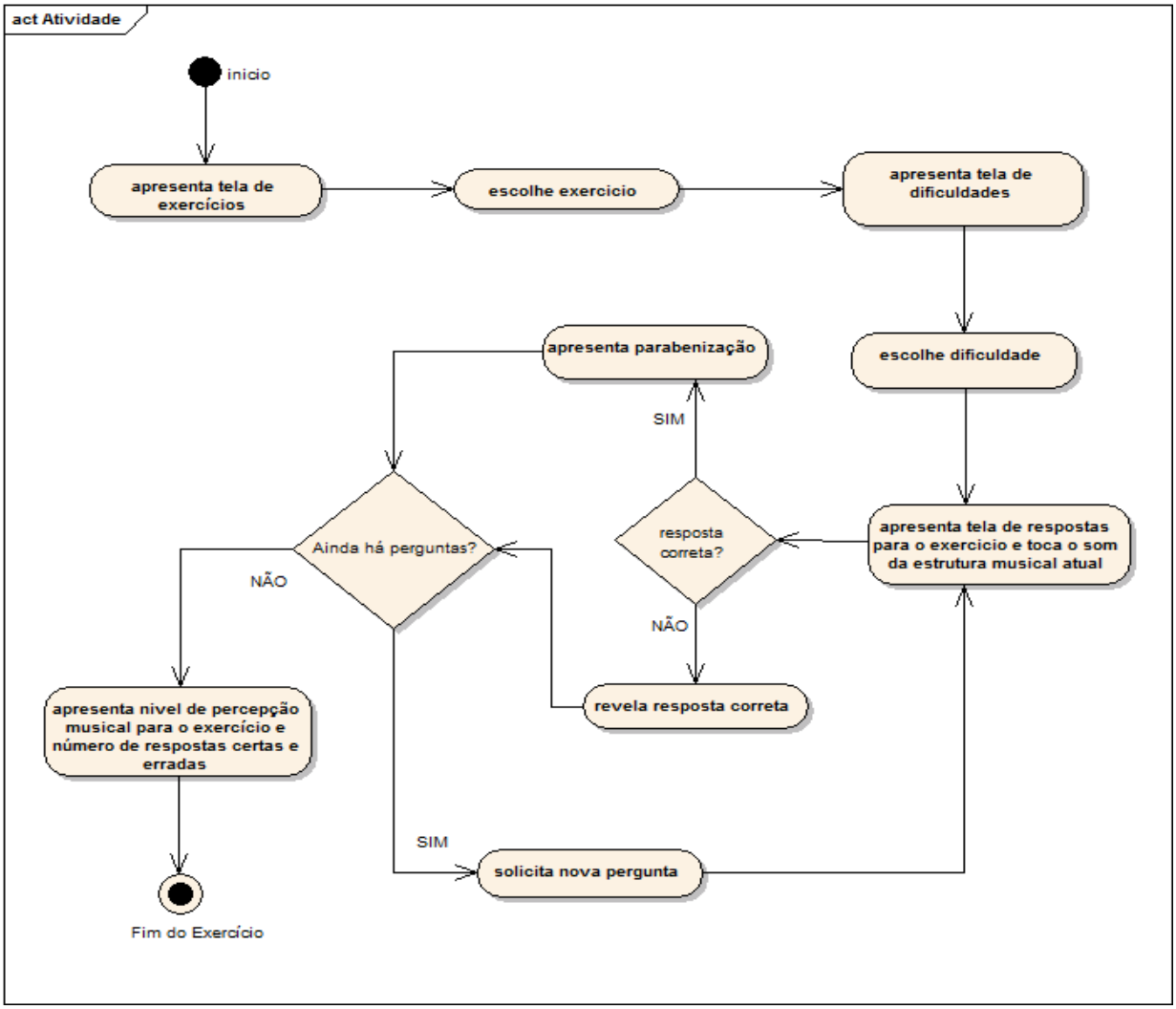


Figura 4 – Diagrama de Atividades dos exercícios da aplicação

A execução dos exercícios da aplicação pode ser vista através do diagrama de seqüência apresentado na Figura 5. Nesta figura percebe-se que a rotina de perguntas está dentro do retângulo `loop`, prosseguindo para a `telaFim()` somente se não atender mais a condição descrita no retângulo, ou seja, se o número da pergunta atual é menor ou igual à 10.

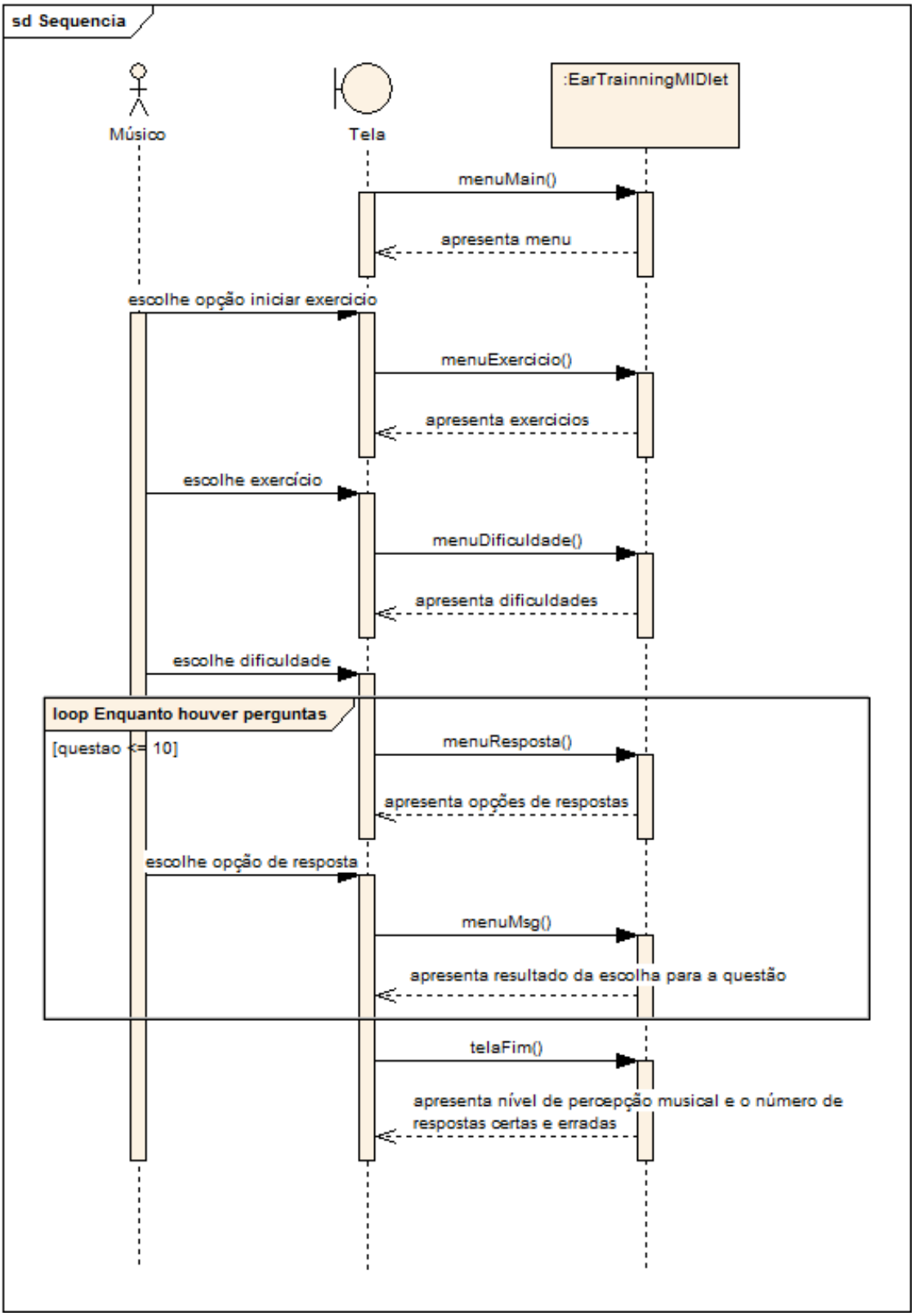


Figura 5 – Diagrama de Seqüência da execução dos exercícios na aplicação

O detalhamento da classe única utilizada para o desenvolvimento da aplicação é apresentado na Figura 6. Os atributos `opcoesExercicio`, `opcoesDificuldade`, `opcoesResposta` e `opcoesInicio` são os componentes gráficos de tela que listam as opções da aplicação conforme o próprio nome já indica. Os atributos `resultadoPergunta` e

`resultadoExercicio` são os componentes gráficos de tela que apresentam os resultados de cada pergunta e exercício da aplicação conforme o próprio nome já indica. O atributo `notaOrigem` guarda o número inteiro de origem 60 para as notas musicais. O atributo `notasMusicais` guarda a descrição das notas musicais (DÓ, DO#, RÉ,...), e o atributo `reprNotas` guarda a descrição das notas musicais em uma representação diferente (C, C#, D, D#,...). Com relação a intervalos tem-se os atributos `escalaCromatica` e `tipoIntervalo` que guardam os intervalos existentes em representação musical (1, 2-, 2, 3m,....) e os tipos de exercício referentes a intervalos respectivamente. Por fim, o atributo `nomeEscalas` guarda a descrição dos tipos de escala que esta aplicação engloba e o atributo `nivelPercepcaoMusical` guarda a descrição dos níveis possíveis nesta aplicação para a percepção musical do músico.

Com relação aos métodos do diagrama de classes da Figura 6, temos:

a) `playNote()`: toca o som da estrutura musical atual;
b) `montaAcorde()`, `montaIntervalo()` e `montaEscala()`: são responsáveis por montar as estruturas musicais;
c) `random()`: gera um número inteiro aleatório, recebendo um número inteiro para o *range* como parâmetro;
d) `getGrandeza()`: pega a descrição maior ou menor para um acorde;
e) `menuMain()`, `menuExercicio()`, `menuDificuldade()` e `menuResposta()`: são os métodos de construção das telas de opções inicial, exercício, dificuldade e resposta respectivamente;
f) `menuMsg()` e `telaFim()`: são os métodos que apresentam as telas de resultados de questões e de exercícios respectivamente;
g) `inicializaQuestionario()`: inicializa os atributos da aplicação.

Figura 6 – Diagrama de classes

O diagrama de classes na Figura 7 apresenta a navegabilidade da aplicação através de

seus componentes gráficos de tela `List` e `Alert` e seus `commands` e `select_commands`. O fluxo do diagrama inicia-se na tela `menuInicio` que inicia os exercícios com a seleção do comando `Iniciar` na lista de comandos. Em seguida apresenta a tela `menuExercicio` que tem a opção de comando `Voltar` indo para a tela `menuInicio`, e também as opções de exercício como comandos de seleção na lista de comandos. Após isto, apresenta a tela `menuDificuldade` que tem a opção de comando `Voltar` indo para a tela `menuExercicio`, e também as opções de dificuldade, que se modificam dependendo do exercício, como comandos de seleção na lista de comandos. Depois disto, apresenta a tela `menuResposta` que tem as opções:

a) `Voltar`: é um comando que faz retornar a tela `menuInicio`;
b) `Play`: é um comando que permanece na mesma tela e reproduz o som da estrutura musical atual;
c) e também as opções de resposta, que se modificam dependendo do exercício, como comandos de seleção na lista de comandos.

Finalmente, apresenta as telas de resultados para a pergunta na tela `resultadoPergunta` e para o exercício na tela `resultadoExercício`. Onde para a pergunta há os comandos:

a) `Confirmar` tanto para a resposta certa quanto para a resposta errada, indo com este comando para a tela de perguntas apenas se ainda não forem respondidas as dez (10) perguntas, senão vai para a tela de resultado para o exercício;
b) e o comando `Play`, para a reprodução do som da estrutura musical atual, apenas para a resposta errada.

Para o exercício há apenas o comando `Confirmar` que remete o usuário à tela `menuInicio`, encerrando o exercício.

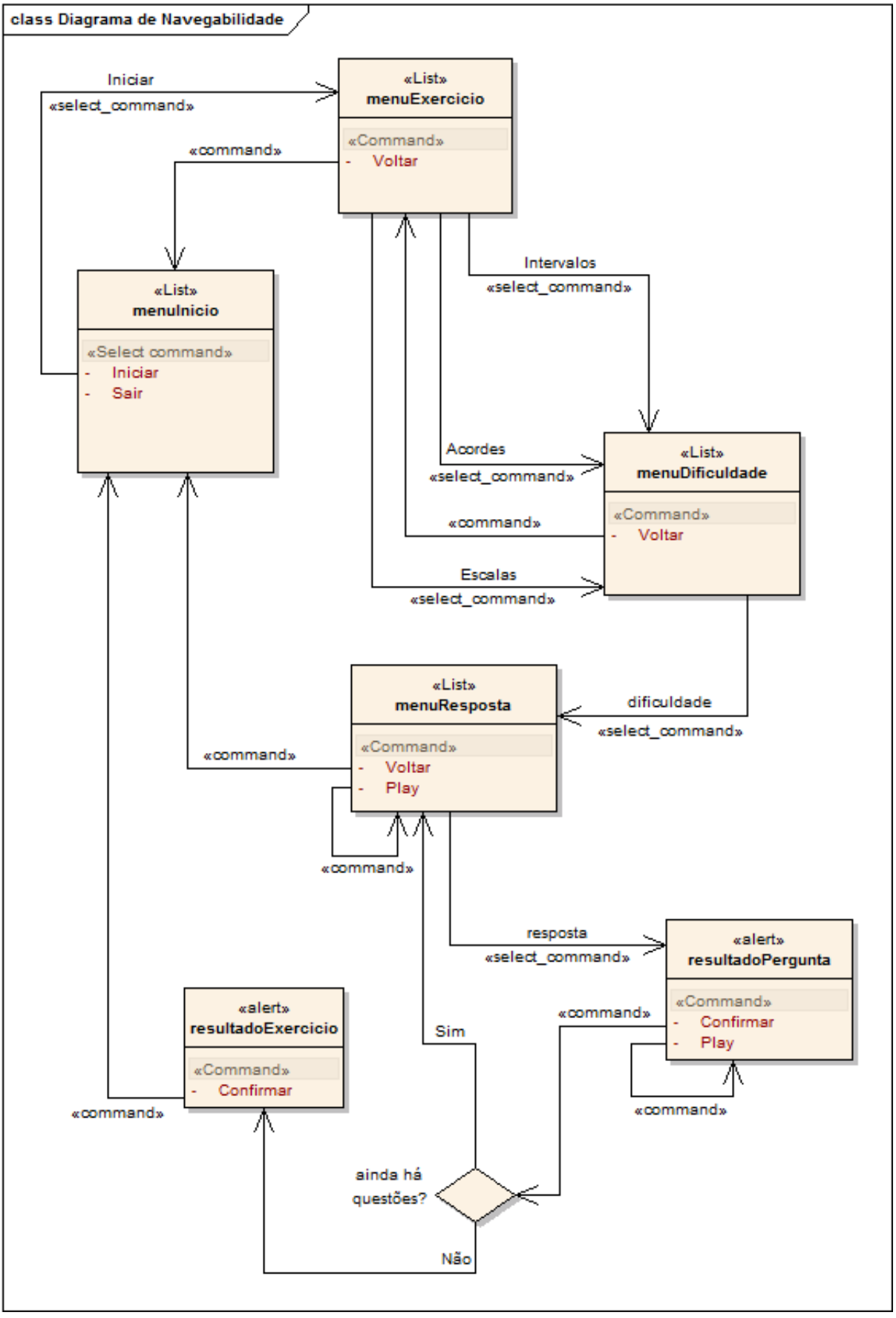


Figura 7 – Diagrama de navegabilidade

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação da aplicação.

3.3.1 Técnicas e ferramentas utilizadas

Para a implementação da aplicação foi utilizada a linguagem de programação Java, contando com a tecnologia JME. O ambiente de desenvolvimento escolhido foi o Eclipse (ECLIPSE, 2009). Para o desenvolvimento da aplicação utilizou-se o plugin EclipseME (SOURCEFORGE, 2009) para o Eclipse.

3.3.1.1 Exercício de Intervalos

No Quadro 17 é possível analisar o um trecho do código fonte do método `montaIntervalo()`, que é responsável pela montagem das estruturas musicais para o exercício de intervalo. Inicialmente ele sorteia a nota fundamental (`nota_tom`) para depois sortear o intervalo em relação a ela, e sorteia também qual das opções na tela vai ser a resposta. O sorteio das notas a serem atribuídas à estrutura musical que irá ser tocada é feita de acordo com as opções de dificuldade, escolhidas pelo usuário, contidas no atributo `opcaoDificuldade`. É sorteado entre intervalos crescentes ou decrescentes apenas se o usuário escolher esta dificuldade, sendo considerada a variável `randon_op` apenas neste caso, caso contrário a própria dificuldade já indica se vai ser crescente ou decrescente. Se o exercício de intervalos for de dificuldade crescente então adiciona um intervalos acima da nota fundamenta, caso contrário adiciona um intervalo abaixo da nota fundamental. No fim são atribuídas as variáveis `nota_tom` (nota fundamental) e `inter` (intervalo em relação a nota fundamental) para a estrutura musical representada pelo atributo notas.

```
private void montaIntervalo() {
	int nota_tom = this.random(12) + notaOrigem;
	this.resp = this.random(3);
	int inter = 0; //número da nota
	int randon_op = 0; //sorteia intervalo crescente ou decrescente
	if (opcoesDificuldade.getSelectedIndex() == 2) {
		randon_op = this.random(2);
	} else {
		randon_op = -1;
	}

	if (opcoesDificuldade.getSelectedIndex() == 0 || randon_op == 0) {
		inter = this.random(12) + nota_tom; //intervalo ascendente
	} else {
		inter = nota_tom - this.random(12);
	}
	int inter_aux = 0;
	this.notas = new int[2];
	this.notas[0] = nota_tom; // primeira
	this.notas[1] = inter; // intervalo
```

Quadro 17 – Trecho do código fonte do método `montaIntervalo()` que é responsável pela montagem das estruturas musicais para o exercício de intervalos

3.3.1.2 Exercício de Acordes

No Quadro 18 é possível analisar um trecho do código fonte do método `montaAcorde()`, que é responsável pela montagem das estruturas musicais para o exercício de acordes. Inicialmente este método sorteia a nota fundamental do acorde, atribuindo esta nota à variável `nota_tom`, para depois sortear as outras notas em relação ao mesmo. Após isso sorteia, baseado na dificuldade escolhida, qual das opções na tela vai ser a resposta, atribuindo a resposta ao atributo `resp`, e designa, também baseado na dificuldade escolhida, quantas opções de resposta haverá na tela, atribuindo ao atributo `opcao`. As notas à serem atribuídas a estrutura musical que irá ser tocada são designadas de acordo com a resposta sorteada e as opções de dificuldade escolhidas pelo usuário, contidas no atributo `opcaoDificuldade`. No fim da construção da estrutura do acorde, se a dificuldade for de acordes invertidos, é trocada a ordem das notas.

```
private void montaAcorde() {
int nota_tom = this.random(12) + notaOrigem; //primeira ou fundamental
if (opcoesDificuldade.getSelectedIndex() == 0) {
       this.opcao = new String[2];
       this.resp = this.random(2);
} else {
       this.resp = this.random(4);
       this.opcao = new String[4];
}
int inverte_aux = 0;
this.notas = new int[5];
if (opcoesDificuldade.getSelectedIndex() == 0) { // acordes maiores/menores
       this.notas[0] = nota_tom; //1a
       if (this.resp == 0) { // acorde maior
              this.notas[1] = nota_tom + 4; //3M
       } else {
              this.notas[1] = nota_tom + 3; //3m
       }
       this.notas[2] = nota_tom + 7; //5 justa/perfeita/generica
       this.notas[3] = nota_tom + 0 + 12; //1a - uma oitava acima
       this.notas[4] = nota_tom + 7 + 12; //5a perfeita - uma oitava acima
} else {
       this.notas[0] = nota_tom; //1a
       if (this.resp == 0 || this.resp == 2) { // acorde maior ou aumentado
              this.notas[1] = nota_tom + 4; //3M
       } else if (this.resp == 1 || this.resp == 3) { // acorde menor ou dim
              this.notas[1] = nota_tom + 3; //3m
       }
       if (this.resp == 0 || this.resp == 1) { // acorde maior ou aumentado
              this.notas[2] = nota_tom + 7; //5 justa/perfeita/generica
              this.notas[4] = nota_tom + 7 + 12; //5a perfeita - uma 8va acima
       } else if (this.resp == 2) { // se for acorde aumentado
              this.notas[2] = nota_tom + 8; //5 aumentada
              this.notas[4] = nota_tom + 8 + 12; //5a perfeita - uma 8va acima
       } else { // se for acorde diminuto
              this.notas[2] = nota_tom + 6; //5 diminuta
              this.notas[4] = nota_tom + 6 + 12; //5a perfeita - uma 8va acima
       }
       this.notas[3] = nota_tom + 0 + 12; //1a - uma oitava acima
       if (opcoesDificuldade.getSelectedIndex() == 2) { //se for um acorde
invertido
              inverte_aux = this.notas[1];
              this.notas[1] = this.notas[0];
              this.notas[0] = inverte_aux;
       }
```

Quadro 18 – Trecho do código fonte do método `montaAcorde()` que é responsável pela montagem das estruturas musicais para o exercício de acordes

3.3.1.3 Exercício de Escalas

Pelo Quadro 19 é possível analisar um trecho do código fonte do método `montaEscala()`, que é responsável pela montagem das estruturas musicais para o exercício de escalas. Inicialmente este método sorteia a nota fundamental da escala para depois sortear as outras notas em relação à mesma. Após isso sorteia, baseado na dificuldade escolhida, qual das opções na tela vai ser a resposta, atribuindo a resposta ao atributo `resp`, e designa,

também baseado na dificuldade escolhida, quantas opções de resposta haverá na tela, atribuindo ao atributo `opcao`. As notas à serem atribuídas a estrutura musical de escalas que irá ser tocada é feita de acordo com as opções de dificuldade escolhidas pelo usuário, contidas no atributo `opcaoDificuldade`.

```
private void montaEscala() {
	int nota_tom = this.random(12) + notaOrigem;
	if (opcoesDificuldade.getSelectedIndex() == 0) { //2 escalas
		this.opcao = new String[2];
		resp = this.random(2);
	} else if (opcoesDificuldade.getSelectedIndex() == 1) { //3 escalas
		this.opcao = new String[3];
		resp = this.random(3);
	} else { //4 Escalas
		resp = this.random(4);
		this.opcao = new String[4];
	}
	/* -- TONS e Semitons de ESCALAS --
	MAIOR:           T T s T T T   s
	MENOR Natural:   T s T T s T   T
	MENOR Harmônica: T s T T s T+s s
	MENOR Melódica:  T s T T T T   s  */
	this.notas = new int[8]; // a estrutura musical terá 8 notas
	if (this.resp == 0) { //Maior
		this.notas[0] = nota_tom; // 1a
		this.notas[1] = nota_tom + 2; //Tom
		this.notas[2] = nota_tom + 4; //Tom
		this.notas[3] = nota_tom + 5; //Semitom
		this.notas[4] = nota_tom + 7; //Tom
		this.notas[5] = nota_tom + 9; //Tom
		this.notas[6] = nota_tom + 11; //Tom
		this.notas[7] = nota_tom + 12; //Semitom
	} else if (this.resp == 1) { //Menor Natural
		this.notas[0] = nota_tom; // 1a
		this.notas[1] = nota_tom + 2; //Tom
		this.notas[2] = nota_tom + 3; //Semitom
		this.notas[3] = nota_tom + 5; //Tom
		this.notas[4] = nota_tom + 7; //Tom
		this.notas[5] = nota_tom + 8; //Semitom
		this.notas[6] = nota_tom + 10; //Tom
		this.notas[7] = nota_tom + 12; //Tom
	} else if (this.resp == 2) { // Menor Harmônica
		this.notas[0] = nota_tom; // 1a
		this.notas[1] = nota_tom + 2; //Tom
		this.notas[2] = nota_tom + 3; //Semitom
		this.notas[3] = nota_tom + 5; //Tom
		this.notas[4] = nota_tom + 7; //Tom
		this.notas[5] = nota_tom + 8; //Semitom
		this.notas[6] = nota_tom + 11; //Tom+Semitom
		this.notas[7] = nota_tom + 12; //Tom
	} else { //Menor Melódica
		this.notas[0] = nota_tom; // 1a
		this.notas[1] = nota_tom + 2; //Tom
		this.notas[2] = nota_tom + 3; //Semitom
		this.notas[3] = nota_tom + 5; //Tom
		this.notas[4] = nota_tom + 7; //Tom
		this.notas[5] = nota_tom + 9; //Tom
		this.notas[6] = nota_tom + 11; //Tom
		this.notas[7] = nota_tom + 12; //Semitom
	}
```

Quadro 19 – Trecho do código fonte do método `montaEscala()` que é responsável pela montagem das estruturas musicais para o exercício de escalas

3.3.1.4 Tocar estruturas musicais

O Quadro 20 apresenta o código fonte do método `playNote()`, que é responsável por reproduzir o som das estruturas musicais da aplicação. Este método utiliza para sua iteração o atributo `notas` que contém toda a estrutura musical em números inteiros que irá ser tocada para a questão atual.

```
private void playNote() {
	for (int i = 0; i < this.notas.length; i++) {
		try {
			Manager.playTone((int) notas[i], 200, 100);
		} catch (MediaException me) {
			System.out.println("Erro ao tocar nota: \'" + me.toString()
+ "\'!");
		}
	}
}
```

Quadro 20 - Código fonte do método `playNote()` que é responsável por reproduzir o som das estruturas musicais

3.3.2 Operacionalidade da implementação

Esta seção tem por objetivo mostrar a operacionalidade da implementação em nível de usuário. Nas próximas seções serão abordadas todas as funcionalidades da aplicação.

3.3.2.1 Realizando um exercício de Escalas

Na tela de escolha de exercícios, o usuário tem a opção de exercício `Escalas`. A Figura 8 demonstra esta situação.

Figura 8 – Realizar exercicio de escalas

Após selecionar a opção `Escalas` são exibidas as opções de dificuldade conforme a Figura 9.

Figura 9 – Tela de dificuldades para o exercício de escalas

As opções de dificuldade para o exercício de escalas são:

a) `Duas escalas`: faz perguntas para as escalas Maior e Menor Natural;

b) `Três escalas`: faz perguntas para as escalas Maior, Menor Natural e Menor Harmônica;

c) `Quatro escalas`: faz perguntas para as escalas Maior, Menor Natural, Menor Harmônica e Menor Melódica.

As perguntas referentes a escalas são mostradas após a escolha da dificuldade, conforme a Figura 10. Após isso, toca-se o som da pergunta correspondente. Caso o usuário queira voltar para a tela inicial, há o comando `Voltar` para esta situação, e caso o usuário queira tocar o som da escala novamente, pode usar o comando `Play`.

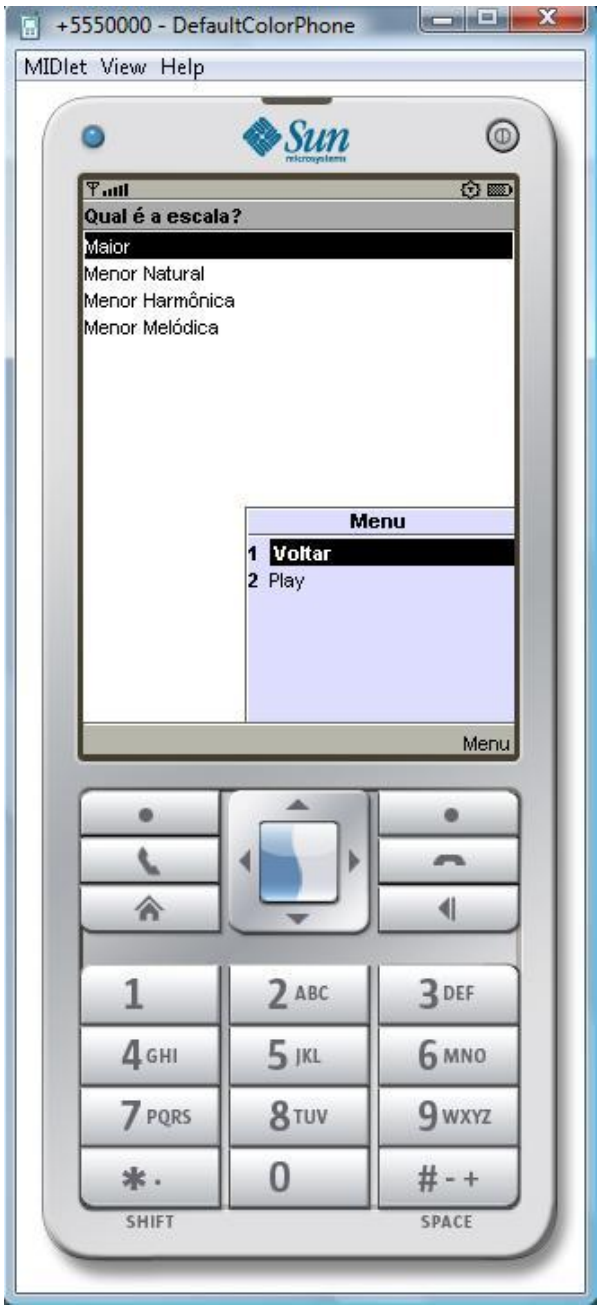


Figura 10 – Tela de questão para o exercício de escalas

3.3.2.2 Realizando um exercício de Acordes

Na tela de escolha de exercícios, o usuário tem a opção de exercício `Acordes`. A Figura 11 demonstra esta situação.

Figura 11 –Realizar exercício de acordes

Após selecionar a opção `Acordes` são exibidas as opções de dificuldade conforme a Figura 12.

Figura 12 – Tela de dificuldades para o exercício de acordes

As opções de dificuldade para o exercício de acordes são:

a) `Acordes maior/menor`: faz perguntas para os acordes Maior e Menor;

b) `Acordes simples`: faz perguntas para os acordes Maior, Menor, Quinta aumentada e Diminuto;

c) `Acordes com inversão`: faz perguntas para os acordes simples, mas com inversão.

As perguntas referentes a acordes são mostradas após a escolha da dificuldade, conforme a Figura 13. Após isso, toca-se o som da pergunta correspondente. Caso o usuário queira voltar para a tela inicial, há o comando `Voltar` para esta situação, e caso o usuário queira tocar o som do acorde novamente, pode usar o comando `Play`.

Figura 13 – Tela de questão para o exercício de acordes

3.3.2.3 Realizando um exercício de Intervalos

Na tela de escolha de exercícios, o usuário tem a opção de exercício de Intervalos. A Figura 14 demonstra esta situação.

Figura 14 – Realizar exercício de intervalos

Após selecionar a opção `Intervalos` são exibidas as opções de dificuldade conforme a Figura 15.

Figura 15 – Tela de dificuldades para o exercício de intervalos

As opções de dificuldade para o exercício de intervalos são:

a) `Crescente`: faz perguntas para intervalos ascendentes à nota fundamental;

b) `Decrescente`: faz perguntas para intervalos descendentes à nota fundamental;

c) `Crescente/Decrescente`: faz perguntas aleatórias entre intervalos ascendentes ou intervalos descendentes à nota fundamental.

As perguntas referentes a intervalos são apresentados após a escolha da dificuldade, conforme a Figura 16. Após isso, toca-se o som da pergunta correspondente. Caso o usuário queira voltar para a tela inicial, há o comando `Voltar` para esta situação, e caso o usuário queira tocar o som do intervalo novamente, pode usar o comando `Play`.

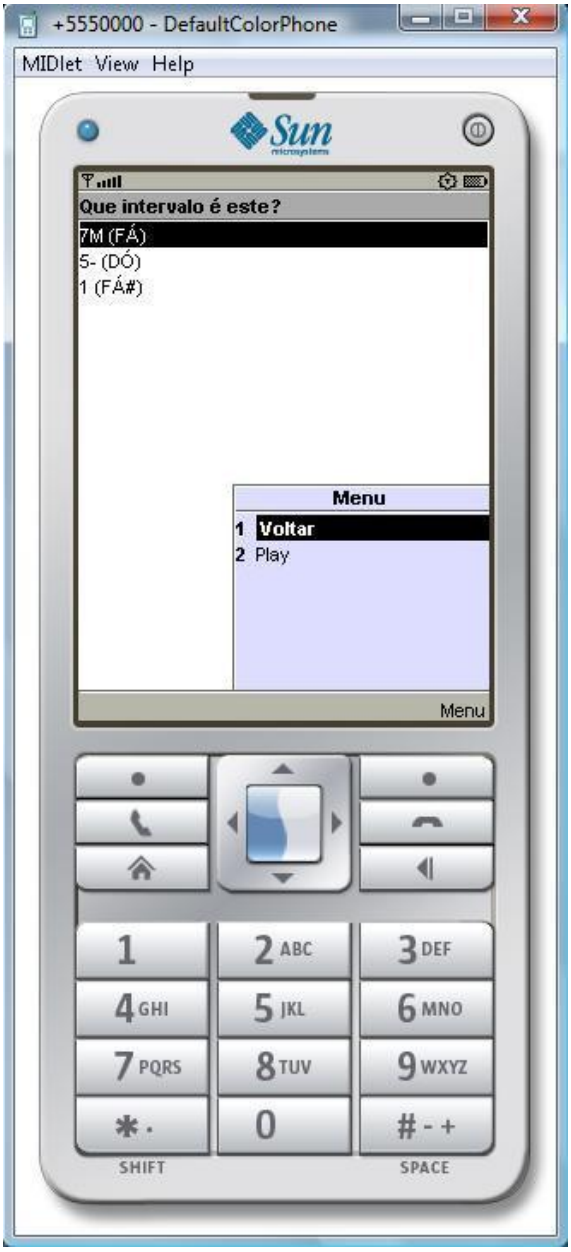


Figura 16 – Tela de questão para o exercício de intervalos

3.3.2.4 Resultados de questões e de fim do exercício

Após cada escolha por parte do usuário para uma resposta, se a mesma estiver correta, a aplicação apresenta uma mensagem informando que a resposta está correta, conforme a Figura 17, senão apresenta uma mensagem informando que a resposta está errada e qual deveria ser a resposta correta, conforme a Figura 18, apresentando um comando a mais no menu de comandos, chamado `Play`, para tocar a estrutura musical novamente, como mostra a Figura 19.

Figura 17 – Tela de resultado positivo da questão atual do exercício

Figura 18 – Tela de resultado negativo da questão atual do exercício

Figura 19 – Tela de resultado negativo da questão atual do exercício com menu de opções

Se a questão atual for de número 10, então após o resultado dela, a aplicação apresenta uma outra tela com uma mensagem informando como está a audição do músico para o exercício realizado, e o número de respostas certas e respostas erradas, conforme a Figura 20.

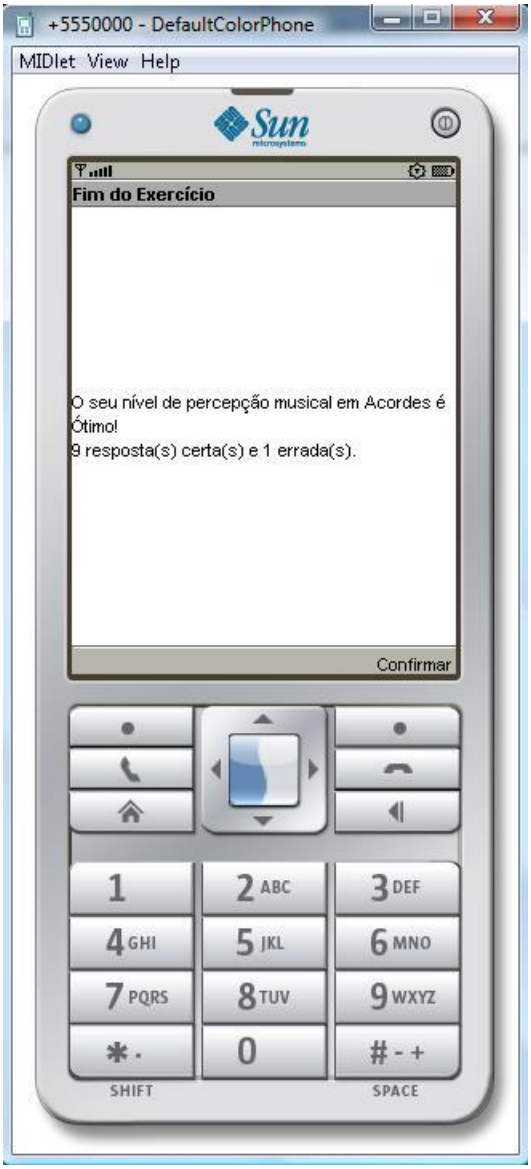


Figura 20 – Tela de resultado do exercício atualmente realizado

3.4 RESULTADOS E DISCUSSÃO

A aplicação desenvolvida demonstra ser muito útil e poderosa em comparação com o trabalho de Tomedi (2002) e sobre o software EARMASTER (2009) pelo fato de ser uma aplicação móvel e de gerar exercícios interativos para a teoria musical. Devido à dificuldade e à própria limitação e complexidade do desenvolvimento de aplicações para dispositivos móveis em JME, espera-se estender futuramente a parte de interface gráfica e qualidade de áudio. Por conseqüência disto, pretende-se estender também os recursos e configurações da parte musical da aplicação desenvolvida, pois nota-se estas características no software EARMASTER (2009). Neste software, por exemplo, o usuário pode construir exercícios para serem disponibilizados para outras pessoas. Quanto ao trabalho de Tomedi (2002), pode-se ter mais opções de exercícios de teoria musical, possibilitando exercitar modos de escalas e outros tipos de acordes não disponíveis na aplicação desenvolvida no presente trabalho. O Quadro 21 abaixo apresenta um comparativo entre a aplicação desenvolvida e os trabalhos correlatos apresentados, caracterizando:

a) mobilidade: se é ou não uma aplicação móvel;
b) qualidade de áudio: é designada como básica quando verifica-se que a reprodução de som é feita apenas com a tecnologia MIDI e é designado como avançada quando verifica-se que há outra tecnologia de reprodução de som;
c) interface gráfica: designa-se como sendo básica quando apresenta uma tela com poucos objetos, e designa-se como sendo avançada quando apresenta uma tela rica e elaborada;
d) Recursos e configurações: designa-se como sendo poucos a baixa quantidade de elementos musicais que engloba e a baixa quantidade de opções de gerenciamento dos seus exercícios, por outro lado designa-se muitos como sendo o oposto de poucos e intermediário como sendo o intermédio entre poucos e muitos.

	Mobilidade	Qualidade de Áudio	Interface Gráfica	Recursos e Configurações
KESTRING	É uma aplicação móvel	Básica	Básica	Poucos
EARMASTER	Não apresenta mobilidade	Avançada	Avançada	Muitos
TOMEDI	Não apresenta mobilidade	Básica	Básica	Intermediário

Quadro 21 – Comparativo entre a aplicação desenvolvida e os trabalhos correlatos

Dentre os problemas que Tomedi (2002) encontrou, o principal, e que também se encontra no presente trabalho, foi o de relacionar a teoria musical de maneira paramétrica, de forma a possibilitar a utilização de um recurso computacional juntamente com uma pré-programação utilizando a tecnologia MIDI, para gerar questões para o treinamento da percepção musical. Este problema foi resolvido na aplicação desenvolvida, da mesma forma que Tomedi (2002) resolveu, utilizando o modelo inteiro de classes de alturas musicais. O trabalho de Tomedi (2002) ajudou a ter uma visão de como é trabalhar com a tecnologia MIDI e como utilizar o modelo inteiro de classes de alturas musicais, para que houvesse uma otimização da aplicação do presente trabalho, decorrente das características dos dispositivos móveis.

Em relação ao tipo de exercício disponibilizado pela aplicação desenvolvida no presente trabalho, limita-se a acordes e arpejos, escalas e intervalos na teoria musical.

4 CONCLUSÕES

O objetivo principal deste trabalho foi desenvolver um protótipo de aplicação baseado na teoria musical, para gerar questões visando o treinamento da percepção musical, em dispositivos móveis.

O problema central e principal encontrado neste trabalho é o de como relacionar a teoria musical de forma paramétrica, e com isso tornar possível a construção de estruturas musicais utilizando recursos computacionais para gerar questões visando o treinamento da percepção musical.

A geração dinâmica de questões utilizando a tecnologia MIDI para treinamento de intervalos, escalas e acordes foi possível através do modelo inteiro de classes de alturas, que relaciona as alturas das notas com os números inteiros. E esta relação permite parametrizar as classes de altura no protótipo de aplicação desenvolvido a fim de gerar automaticamente questões para o treinamento auditivo.

Com este trabalho conclui-se que é possível utilizar relacionamentos pré-definidos entre alturas e números inteiros baseadas na teoria musical para a geração de sistemas musicais, pois a música pode ser entendida, de certa forma, como uma ciência exata, com atributos lógicos e aritméticos, permitindo assim a aplicação de relações matemáticas sobre as relações existentes na teoria musical.

A tecnologia JME pode fornecer com êxito suporte à aplicação desenvolvida, pois a aplicação teve desempenho compatível à arquitetura e a computação musical utilizada nos dispositivos móveis.

Espera-se que o protótipo de aplicação desenvolvida melhore a percepção musical do usuário conforme o que já acontece com o EARMASTER (2009), por ser um software já atuante no mercado.

4.1 EXTENSÕES

Como extensões deste trabalho, sugere-se a implementação de uma interface visual mais rica e elaborada para a aplicação, com mais recursos e configurações, onde poderia ser mostrado ao usuário uma visualização do instrumento (por exemplo, o desenho do braço de uma guitarra ou as teclas de um piano) ao invés de opções de múltipla escolha. Isto melhoraria a afinidade do músico com o instrumento, além do treinamento da percepção musical. Poderia ser exibido também a notação musical padrão, com notas musicais, pauta, armadura de clave, entre outros.

Ao final de cada exercício mostrar um quadro de evolução do usuário para que ele veja se suas habilidades melhoraram.

Com a presente evolução dos dispositivos móveis e suas plataformas de desenvolvimento, vê-se como uma possível extensão acrescentar polifonia para a aplicação, que consiste em tocar várias notas simultaneamente, possibilitando exercícios de acordes e intervalos harmônicos.

Poderia ser implementado também, o treinamento de outros elementos da teoria musical, como por exemplo:

a) ritmos;
b) outros tipos de acordes e escalas;
c) o reconhecimento das notas que compõem um acorde ou escala ou intervalos;
d) o reconhecimento do tom em que se encontra o acorde ou a escala.

Conforme pode-se perceber, existem outros exercícios que podem ser montados para complementar o treinamento da percepção musical de um músico, utilizando-se os recursos apresentados neste trabalho de conclusão de curso.

REFERÊNCIAS BIBLIOGRÁFICAS

BARBAGALLO, R. **Wireles game development in Java with MIDP 2.0**. Texas: Wordware, 2004.

BRANDT, C. **Teoria musical básica.** [S.l], 2007. Disponível em: <http:// www.jazzbossa.com/sabatella/05.01.teoriamusicalbasica.html>. Acesso em: 23 abr. 2009.

DIAS, W. **Teoria musical**: arpejo. Revista teclado e piano. n.º 128, 2007. Disponível em: <http://canaldomusico.uol.com.br/materias_detalhe.asp?ukey=39329173107AFZN72Z>. Acesso em: 30 out. 2008.

EARMASTER. **EARMASTER 5**: interactive ear training software. [S.l], 2009. Disponível em: <http://www.earmaster.com/>. Acesso em: 22 abr. 2009.

ECLIPSE. **Eclipse.org home**. [S.l.], 2009. Disponível em: <http://www.eclipse.org>. Acesso em: 10 maio 2009.

GORDON, E. E. **Teoria de aprendizagem musical**: competências, conteúdos e padrões. Lisboa: Atlas, 2000.

JOHNSON, T. M. **Java para dispositivos móveis**: desenvolvendo aplicações com J2ME. São Paulo: Novatec, 2008.

MACIEL, F. R.; PITONI, R. **Connected, limited device configuration e mobile information device profile**. [Porto Alegre], 2001. Disponível em: <http://www.inf.ufrgs.br/procpar/disc/inf01008/trabalhos/sem01-1/t2/pitoni/#int>. Acesso em: 06 set. 2008.

MICROSOFT DEVELOPER NETWORK. **Mobilidade.** [S.l.], 2008. Disponível em: <http://msdn.microsoft.com/pt-br/library/cc580606.aspx>. Acesso em: 1 nov. 2008.

MORETTI, A. L. **Protótipo de um software para o reconhecimento de notas musicais**. 2003. 66 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: < http://www.inf.furb.br/tcc/index.php?cd=6&tcc=685 >. Acesso em: 07 set. 2008.

OMG. **Object management group**. [S.l], 2009. Disponível em: <http://www.omg.org>. Acesso em: 28 abr. 2009.

RAHN, John. **Basic atonal theory.** New York: Schrimer Books, 1980.

SCLIAR, E. **Elementos de teoria musical.** São Paulo: Atlas, 1994.

SOURCEFORGE. **EclipseME.** [S.l.], 2009. Disponível em: <http://sourceforge.net/projects/eclipseme>. Acesso em: 06 mar. 2009.

SPARX SYSTEMS. **UML tools for software development and modelling**: Enterprise Architect UML modeling tool. [S.l.], 2009. Disponível em: <http://www.sparxsystems.com.au>. Acesso em: 20 maio 2009.

SUN DEVELOPER NETWORK. **Java ME**: the most ubiquitous application platform for mobile devices. [S.l.], 2008a. Disponível em: <java.sun.com/j2me>. Acesso em: 03 set. 2008.

_____. **Mobile Information Device Profile (MIDP).** [S.l.], 2008b. Disponível em: <http://java.sun.com/products/midp/>. Acesso em: 03 set. 2008.

SUN MICROSYSTEMS. **Sun microsystems.** [S.l.], 2008. Disponível em: <http://www.sun.com/>. Acesso em: 03 set. 2008.

TOMEDI, R. A. B. **Protótipo de um sistema para auxílio ao treinamento da percepção musical**. 2002. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.inf.furb.br/tcc/index.php?cd=7&tcc=682>. Acesso em: 28 agos. 2008.