UNIVERSIDADE ESTADUAL DO PIAUÍ
CAMPUS PROF. ALEXANDRE ALVES DE OLIVEIRA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOSÉ FRANCISCO CARDOSO NETO

SOLFEJANDO:
PROTÓTIPO DE UM DICIONÁRIO DE ACORDES PARA ESTUDANTES DE MÚSICA

PARNAÍBA
2012

JOSÉ FRANCISCO CARDOSO NETO

SOLFEJANDO:
PROTÓTIPO DE UM DICIONÁRIO DE ACORDES PARA ESTUDANTES DE MÚSICA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual do Piauí – UESPI, Campus Prof. Alexandre Alves de Oliveira, como parte das exigências da disciplina de Estágio Supervisionado, requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Orientador: Esp. Mayllon Veras da Silva

PARNAÍBA

2012

FICHA CATALOGRÁFICA ELABORADA PELO
BIBLIOTECÁRIO HERNANDES ANDRADE SILVA CRB-3/936

C268s Cardoso Neto, José Francisco
Solfejando: protótipo de um dicionário de acordes para estudantes de música / José Francisco Cardoso Neto. -- Parnaíba: 2012.
48f. ; il.

Monografia submetida como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, Universidade Estadual do Piauí, Parnaíba-2012.
Orientador: Esp. Mayllon Veras da Silva.

1. Programação (Computadores). 2. Engenharia de Software. 3. Dispositivos Móveis - Dicionário de Acordes. I. Título.

CDD – 005.1

JOSÉ FRANCISCO CARDOSO NETO

SOLFEJANDO: PROTÓTIPO DE UM DICIONÁRIO DE ACORDES PARA ESTUDANTES DE MÚSICA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual do Piauí – UESPI, Campus Prof. Alexandre Alves de Oliveira, como parte das exigências da disciplina de Estágio Supervisionado, requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Esp. Mayllon Veras da Silva

Monografia Aprovada em: **24 de agosto de 2012**.

Banca Examinadora:

Prof. Esp. Mayllon Veras da Silva
UESPI/Parnaíba – Orientador

Prof. M.Sc. Francisco das Chagas Rocha
UESPI/Parnaíba – Avaliador Interno

Prof. M.Sc. José Flávio Gomes Barros
FAP/Parnaíba – Avaliador Externo

À Cleidiana Alves dos Santos

AGRADECIMENTO

Agradeço em primeiro lugar a Deus por ter me concedido o dom da inteligência.

Aos meus familiares e em especial à minha mãe, Joana Cardoso, por ter sido pai e mãe quando necessário.

À Cleidiana Alves que sempre me incentivou e durante meu período de estudo aceitou manter um namoro "semipresencial".

Aos meus amigos que acreditaram, ou não, que um dia eu chegaria lá.

Ao meu orientador, Mayllon Veras, que me ajudou a colocar em prática minha idéia.

"E por que caímos, senhor?
Para aprender a levantar, ... senhor!"
(Alfred, no filme: Batman Begins)

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para o auxílio de estudantes de música. O objetivo principal é proporcionar aos usuários um dicionário de acorde voltado para a teoria musical. Na primeira parte é feito um estudo sobre a teoria musical em seus conceitos e relações com a tecnologia, são estudados os dispositivos móveis com as tecnologias voltadas para esses aparelhos e o uso de sons nestes aparelhos. São explanados os estudos de caso, diagramas e métodos utilizados para o desenvolvimento do mesmo. Na segunda parte é feito o desenvolvimento do protótipo. O protótipo foi feito usando MIDI embasado na teoria musical dos acordes, escalas e intervalos e é voltado para uso em dispositivos móveis. Para isso foi utilizado o J2ME que tem uma larga biblioteca para esses dispositivos.

Palavras-chave: Teoria Musical. MIDI. J2ME. Dispositivos Móveis. Escalas. Acordes.

ABSTRACT

This work presents the development of a prototype to help the students about music. The principal objective is to provide to the users a chord dictionary turned to the musical theory. In the first part is done a study about the musical theory in its concepts and relation to the technology, mobile devices are studied with the technologies turned to these devices and the use of sounds in these devices. There are explanations of the case studies, diagrams and methods utilized to the development the same. In the second part is done the development of the prototype. The prototype was made using MIDI grounded in the music theory of the chords, scales and intervals and it is turned to the use in mobile devices. For this was used J2ME having a large library for these devices.

Key-words: Musical Theory. MIDI. J2ME. Mobile Devices. Scales. Chords.

LISTA DE SIGLAS

AMEI	Association Musical Eletronics Industry
AMS	Application Management Software
API	Application Programming Interface
CDC	Connected Device Configuration
CDDL	Common Development and Distribution License
CLDC	Connected, Limited Device Configuration
GNU	General Public License
J2ME	Java 2 Micro Edition
KVM	Kilo Virtual Machine
MIDI	Musical Instruments Digital Interface
MIDP	Mobile Information Device Profile
MMA	Midi Manufactures Association
OHA	Open Handset Alliance
PDA	Personal Digital Assistant
PyS60	Python for Series 60
S60	Series 60
UML	Unifield Modeling Language

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Em todas as atividades desenvolvidas no dia a dia, seja no nível profissional ou não, é imprescindível ter uma fonte de consulta onde se possa esclarecer algumas dúvidas. No universo da música não é diferente. Muitas vezes o músico se depara com situações de dúvidas onde por vezes tem que recorrer a alguém que possa lhe auxiliar.

As principais dificuldades encontradas entre os músicos são: manter a afinação adequada do seu instrumento, decorar todas as posições do acordes e saber quais notas compõem o acorde a ser tocado.

Mundialmente, o desenvolvimento de aplicativos para celulares tem tido um grande impulso devido ao aumento do número de aparelhos, bem como o acréscimo de novas funcionalidades, serviços e inovações. A partir da chegada dos *smartphones* um salto qualitativo e quantitativo foi sentido principalmente pelos usuários que, agora, já podem desfrutar dos serviços de um *Personal Digital Assistant* (PDA) e de um celular no mesmo aparelho. Esses aparelhos permitem a execução de programas de diversas funcionalidades, como também têm um grande potencial para o desenvolvimento de *softwares* especializados, como aplicações de pesquisa, coleta de dados, jogos e por que não a música também? Aliado a isto, estes aparelhos estão tornando-se mais robustos, com um *hardware* mais potente e com várias facilidades para o desenvolvimento de aplicações.

O grande diferencial desses aparelhos é a presença de um sistema operacional como *Windows Mobile, Symbian, Android,* dentre outros que possibilitam ao usuário adequar o aparelho às suas condições de trabalho, já que é muito grande a gama de aplicativos desenvolvidos para eles. Também estão disponíveis várias linguagens de programação para cada um deles. Hoje, o programador tem a seu dispor várias possibilidades de desenvolvimento para dispositivos portáteis. Sistemas como o *Symbian* suportam C++, J2ME, Qt e uma linguagem desenvolvida pela Nokia especialmente para esse sistema: o *Python for Symbian S60* (PyS60). O *Android* tem suporte a *Java, Python* e outras linguagens, sem falar no *Windows Mobile* que promete dar grande suporte à diversas linguagens de programação.

Pensando nisso é que foi proposto o desenvolvimento deste dicionário de acordes para ser usado em celulares que, devido a sua portabilidade, será de grande auxílio aos músicos no desenvolvimento das suas atividades seja como estudo ou como trabalho. Pois o mesmo poderá ser usado tanto para conhecer a formação dos acordes como ouvir o som correto dos mesmos, além do conhecimento das escalas das diversas tonalidades e prática de exercícios de intervalo.

O presente trabalho está dividido em quatro capítulos incluindo esta introdução.

No segundo capítulo é feito um estudo da música e os dispositivos móveis. É explanado a origem da música com os seus conceitos primordiais como intervalo, escalas e acordes, os diversos tipos de amostragens de som e o padrão MIDI. É feito, ainda, um estudo dos dispositivos móveis com as tecnologias adequadas a cada um deles. São estudados os sistemas operacionais predominantes no mercado hoje, bem como o J2ME como alternativa de desenvolvimento para dispositivos móveis.

No terceiro capítulo são mostrados os passos no desenvolvimento da aplicação. São esplanadas as especificações do protótipo, seus requisitos funcionais e não-funcionais, os diagramas e é mostrado o processo da implementação. Neste capítulo também é feita a demonstração da aplicação seqüenciando as telas passo a passo da aplicação.

E, por fim, no quinto capítulo são.feitas as considerações finais com propostas de trabalhos futuros.

2 MÚSICA E DISPOSITIVOS MÓVEIS

2.1 Teoria musical

Desde o surgimento do homem na terra que o mesmo tem se manifestado como um ser artístico. Dentre essas manifestações as que mais se evidenciam são a pintura e a música.

Segundo Tomedi (2002, p. 4) "o homem das cavernas dava à sua música um sentido religioso. Ele considerava-a um presente dos deuses e atribuía-lhe funções mágicas." Embora não contasse com a complexidade de hoje ela já tinha ritmo e uma linha melódica. Primeiramente estes sons eram produzidos apenas por instrumentos de percussão como tambores feitos com madeira e peles de animais. Mas com o passar do tempo o homem percebeu que soprando em chifres ou em folhas de árvores ele conseguia produzir alguns sons: surgiram os instrumentos de sopro, feitos com madeira ou metais e que hoje são uma grande variedade, dentre eles temos o saxofone, flauta, clarinete, entre outros. Mais tarde surgiram os instrumentos de corda como violino, alaúde, cravo, piano. Juntando os instrumentos de percussão, os metais, madeiras e os de cordas tem-se o conjunto de instrumentos acústicos, ou seja, que produzem som sem o auxílio de nenhum aparelho elétrico amplificador.

Os sons que ouvimos diariamente nos permitem afirmar que existe uma infinidade de sons. Atualmente, segundo Tomedi (2008) na escala ocidental[1], podemos representar os sons com sete notas[2] naturais: Dó (C), Ré (D), Mi (E), Fá (F), Sol (G), Lá (A), Si (B). Essa representação chama-se de escala ocidental temperada. Segundo Carvalho (1997) temperamento é um método de ajuste das distâncias de altura entre os sons, que permite limitar os intervalos determinados no interior da oitava, por identificação das notas muito próximas.

De acordo com Kestring (2009) essas notas admitem algumas alterações de modo ascendente ou descendente. Alterações são simbolizadas pelo sustenido (#) quando ascendente e pelo bemol (b) quando descendente[3]. Essas modificações recebem o nome de acidentes e

---

[1] "É sabido que outras culturas (ameríndios, africanos, etc.) fazem música em sistema sonoro de um quarto de tom (1/4), outros (indus) em sistema sonoro de um oitavo de tom (1/8), e outros ainda (chineses) em sistema sonoro de um dezesseis avos de tom (1/16)" (CARVALHO 1997, 126) seguindo assim outras escalas.

[2] "Nota ou solfa são designativos específicos da fixidade de alteza *relativa* de cada lance sonoro regular afinado, código de intonação (afinação fixa) na música ocidental" (CARVALHO 1997, p. 53).

[3] Além dessas duas alterações existe ainda o bequadro (bq) que anula todos os acidentes, ou seja, ele faz com que as notas voltem ao acorde natural desfazendo o efeito dos sustenidos e bemóis na pauta musical.

deixam estruturada a escala ocidental conforme o quadro 1.

Quadro 1 – Estrutura da Escala Ocidental

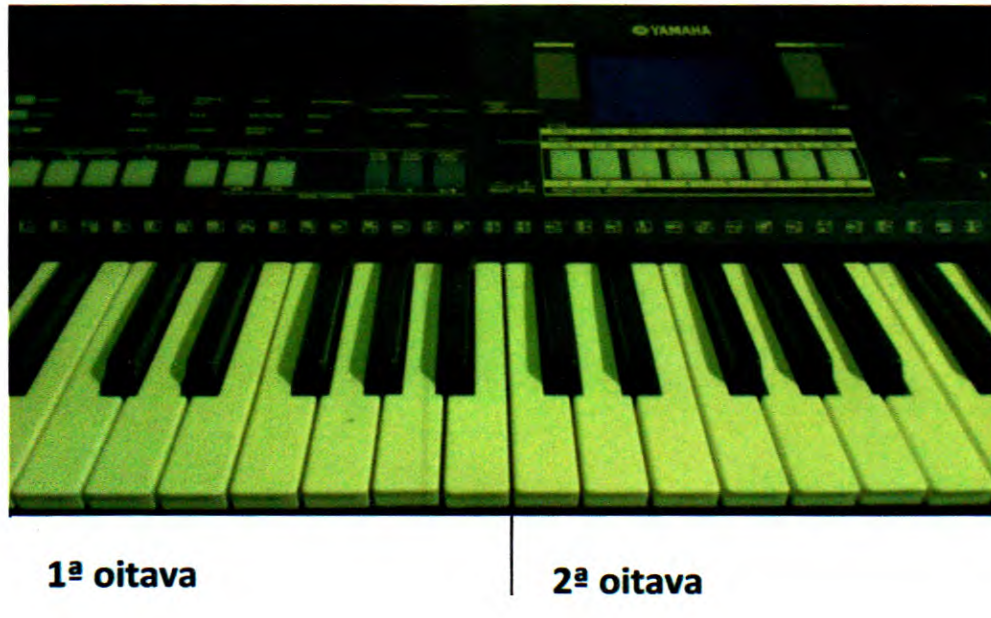
C	C#/ Db	D	D#/ Eb	E	F	F#/ Gb	G	G#/ Ab	A	A#/ Ab	B

Observando essa estrutura verifica-se que todas as notas podem ser alteradas e que uma nota com sustenido corresponde à próxima nota com bemol, por exemplo: Dó sustenido (C#) tem o mesmo som de Ré bemol (Db). Verifica-se também, de acordo com Kestring (2009), que alterando as notas E e B em escala ascendente e F e C em escala descendente encontramos as notas naturais F e C e E e B, respectivamente, por isso dizemos que essas notas não tem sustenido e nem bemol. Essas noções são importantes para compor os intervalos e ter bem definido cada nota que está naquele espaço. Todo esse conjunto de notas forma o sistema temperado que é "baseado na sucessão de sons intervalados por semitons ou meio-tons matematicamente iguais" (CARVALHO 1997, p. 136).

2.1.1 Intervalos e escalas

Como mostrado no quadro 1, a escala ocidental temperada é composta por sete notas musicais naturais e mais os acidentes. Sendo assim, toda e qualquer música composta que obedeça a esse padrão será escrita utilizando essas notas que se repetem no teclado, nas chamadas oitavas (intervalo ente um dó e outro) como mostra a figura 1.

Figura 1 – Sucessão de oitavas no teclado Yamaha PSR-S550b



Intervalo pode ser definido como a "razão (relacionamento, distância, diferença) entre duas freqüências (alturas)" (CARVALHO 1997, p. 101). Cada intervalo recebe um nome que é dado de acordo com a tônica do acorde (primeira nota do acorde) e a distância em semitons entre elas. Assim, segundo Kestring (2009) temos os intervalos maiores, menores, justos, aumentados e diminutos. A distância entre os intervalos e a nota fundamental do acorde chama-se grau e cada grau recebe uma denominação. O quadro 2 mostra esses graus, intervalos e as notas de cada intervalo, tendo como base o início da escala em C (Dó) seguindo a classificação proposta por Carvalho (1997).

Quadro 2 – Correspondência entre grau, intervalo e notas.

GRAU	INTERVALO	NOTA
1° - Tônica	Nota fundamental	C
2° - Sobretônica	Segunda maior	D
3° - Mediante	Terceira maior	E
4° - Sobdominante	Quarta justa	F
5° - Dominante	Quinta justa	G
6° - Sobredominante	Sexta maior	A
7° - Sobtônica ou sensível	Sétima maior ou menor	B ou Bb
8° - Tônica	Repetição do 1° grau uma oitava acima	C

Fonte: Carvalho, 1997.

Todas as notas desse intervalo fazem parte do que é denominado na música de escala cromática ocidental, que é a representação de todos os intervalos musicais usados na música contemporânea no ocidente.

Escala ou gama é uma "progressão ascensional ou descensional de notas estritamente melódicas ordenadas (...) dentro do limite de uma oitava (...) tendo a primeira nota (e a oitava como reiteração da primeira)" (CARVALHO 1997, 160)

As escalas podem ser maior ou menor, de acordo com os intervalos, sendo que cada escala maior tem a sua relativa na escala menor. Por exemplo: a escala de Dó Maior tem sua relativa em Lá Menor. Segundo Carvalho (1997), elas derivam de dois modelos estabelecidos e herdados dos antigos modos grego-latino-eclesiásticos que se tornaram escalas-padrões:

Escala padrão de Dó (C) = C D E F G A B = intervalos = T T ST T T T ST

Escala padrão de Lá (A) = A B C D E F G = intervalos = T ST T T ST T T

Onde T significa um tom e ST semitom.

Seguindo este padrão, podemos montar a escala-base de qualquer acorde variando as notas, mas mantendo o mesmo sistema. Por exemplo, quando montamos a escala de D (Ré Maior) utilizamos o modelo da Escala de C (Dó Maior), sendo assim, ocorre apenas uma transposição[4] das notas obedecendo os mesmos intervalos como mostrado abaixo:

Escala de Dó (C) = C D E F G A B

Escala de Ré (D) = D E F# G A B C#

Observa-se que embora sejam compostas por notas diferentes, as duas escalas seguem o mesmo esquema de intervalos, ou seja, T T ST T T T ST. Os aparelhos modernos já têm implementada uma função chamada de *transpose* que faz a transposição automática da linha melódica executada.

2.1.2 Arpejo e acorde

Segundo Carvalho (1997) acorde é um conjunto de três, quatro ou mais notas de nomes diferentes tocados de forma harmônica e síncrona. Carvalho (1997) ainda classifica os acordes como tríade quando formado por três notas; por quatro, tétrade; por cinco notas, pêntade; e assim por diante. Mas "a harmonia clássica só admite acordes até quatro notas" (CARVALHO 1997, p. 268).

Arpejo consiste em um "acorde em que as notas são executadas (e mantidas) em seqüência (uma depois da outra)" (CARVALHO 1997, p. 289), de modo a soarem como se fossem tocadas por uma harpa. A palavra arpejo deriva do italiano *arpeggio* que quer dizer: à maneira de harpa. Percebe-se que tanto acorde como arpejo são a execução de notas, mas tocadas de um modo diferente, "onde tanto o acorde quanto o arpejo, formam um relacionamento harmônico específico entre si" (KESTRING 2009, p. 18).

Um acorde "é basicamente constituído de notas em alternação intercalar por terça, quarta ou quinta" (CARVALHO 1997, p. 205). A nota fundamental é aquela que inicia o acorde e que dá nome ao mesmo e os intervalos intercalares para compor o restante do acorde são dados sempre em relação a essa nota fundamental.

Para a formação dos acordes utiliza-se uma seqüência de intervalos que fazem com que o acorde seja denominado como maior, menor, diminuto, com sétima, quarta e os outros. Segundo Carvalho (1997) os principais acordes formados por três notas são:

- acorde maior: fundamental, terça maior e quinta justa;

[4] Substituição de um tom por outro. (Carvalho 1997, 248).

- acorde menor: fundamental, terça menor e quinta justa;
- acorde com quarta: fundamental, quarta justa e quinta justa.

Ainda segundo Carvalho (1997) os principais acordes com quatro notas são:

- acorde com sétima: fundamental, terça menor, quinta justa e sétima menor;
- acorde menor com sétima: fundamental, terça menor, quinta justa e sétima menor;
- acorde com sétima maior: fundamental, terça maior, quinta justa e sétima maior.

2.1.3 Computação musical

Com o avanço tecnológico, e principalmente a invenção do telefone por Graham Bell em 1876, percebeu-se que o som podia ser convertido em sinal elétrico e vice-versa. Já o gramofone deu a possibilidade de armazenamento e alteração do som.

De acordo com Miletto et al (2004) o início da música eletrônica se deu quando em 1906 Thaddeus Cahill inventou um instrumento que produzia sons ligando a saída de bancos de dínamos ao auto-falante, ou cápsula receptora de um telefone. Baseado nesses avanços e vários anos de estudo, em 1930 os físicos Coupleux e Givelet inventaram um órgão com centenas de válvulas osciladoras para produzir o som das notas. Eles utilizavam uma válvula para cada nota.

Segundo Ueda (2003) desde a criação dos computadores que a música ficou atrelada a eles. Foi já na década de 50, que Lejaren Hiller juntamente com Leonard Isaacson criaram a primeira composição musical gerada por um computador utilizando o Illiac da Universidade de Illions. Eles introduziram algoritmos para notação musical tradicional e o computador gerou a partitura, uma peça para quarteto de cordas chamada *Illiac Suite*.

Por volta de 1970, a Bell Labs desenvolveu um modo de amostrar digitalmente o som, tornando o que é um evento fundamentalmente analógico e contínuo em um sinal digital e discreto, representado por uma série de sinais binários (1 e 0), que podem ser armazenados, manipulados e reproduzidos.

Atualmente, existem várias formas de codificação do som para que sejam processados pelo computador. Eles variam entre si, quanto ao tipo de compressão, com ou sem tratamento. De acordo com Miletto et al (2004) os principais formatos no mercado são:

- Wave, da Microsoft (extensão .wav)

- AIFF, da Apple (extensão .aiff ou .aif)
- Sun Audio, da Sun (extensão .au)
- Real Audio, da RealNetworks (extensão .ra)
- MPEG Layer 3 (extensão .mp3)
- Windows Media Audio, da Microsoft (extensão .wma)

Além desses formatos para reprodução de som pelos computadores existem aqueles que são criados especificamente para os sintetizadores que são dispositivos capazes de gerar som artificialmente através de parâmetros de edição de timbres. Esses sons, armazenados e produzidos pelos sintetizadores, possibilitaram ao músico um grande avanço na execução de canções. Desde a década de 80, várias pesquisas têm sido feitas no ramo da música eletrônica e fabricantes como Yamaha, Roland e Korg, a cada ano se superam no requisito da geração de sons mais puros e perfeitos com "vários tipos de síntese como FM, auditiva, subtrativa, linear e modelagem física" (MILETTO et al 2004, p. 03). O avanço da ciência na área da música é tão marcante a ponto de utilizar inteligência artificial, redes neurais e realidade virtual na construção de sistemas inovadores e que permitem uma grande flexibilidade ao usuário.

Assim, surgiu a Computação Musical que, de acordo com Miletto et al (2004, p. 01) "[...] investiga métodos, técnicas e algoritmos para processamento e geração de som e música, representações digitais e armazenamento de informação sônica e musical".

2.1.3.1 MIDI

De acordo com Tomedi (2002) o *Musical Instruments Digital Interface* (MIDI) é um protocolo de comunicação, um conjunto de comandos entre dispositivos, dando ordens aos mesmos com respeito ao que devem fazer. O que soa são os aparatos, os instrumentos, "o MIDI" que, ademais, tem outras funções fora a de controlar a execução de sons.

Segundo Blanco (2011) o protocolo MIDI foi concebido devido à necessidade dos músicos de controlar vários equipamentos com seus comandos e criar *layers*[5] com vários sons entre eles. Os primeiros resultados desta nova tecnologia foram mostrados no *North American Music Manufacturers Show* de 1983 em Los Angeles. Durante este evento, foram interligados dois sintetizadores através de cabos MIDI, onde ao tocar uma nota em um dos sintetizadores o público pode ver que o outro instrumento também tocava junto. O cabo MIDI funciona entre

[5] Camadas de som que se sobrepõem para formar novos sons.

os instrumentos musicas como os modens funcionam nos computadores.

> A informação MIDI possui caráter notadamente musical: se refere a comandos *play-stop*, ativação de nota, tempo, volume, etc, mesmo que seu uso avançado possibilite muito mais coisas. (BLANCO, 2011)

Dependendo da configuração de hardware de quem manda ou recebe a mensagem MIDI, algumas instruções são ignoradas pelo receptor já que o mesmo só pode executar aquilo que seu hardware permite.

O padrão MIDI é administrado por duas organizações: a *Midi Manufacturers Association*[6] (MMA) e *Association Musical Eletronics Industry*[7] (AMEI), e se constitui em um dos únicos exemplos de um acordo entre diversos fabricantes de equipamentos variados que deu certo.

> [...] a tecnologia MIDI permite que dispositivos eletrônicos (...) 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. (GONTIJO, 1998 apud TOMEDI, 2002, p. 22)

Instrumentos MIDI necessitam de duas portas para MIDI uma IN para receber informações e uma OUT para enviar, mas alguns aparelhos como o teclado Yamaha PSR-550 apresentam também uma porta THRU que funciona como entrada e saída de MIDI.

Algum tempo atrás os sintetizadores só utilizavam cabos específicos de MIDI com cinco pinos que conseguiam reproduzir 16 canais de sons. Antes das entradas USB os cabos MIDI para se comunicarem com os computadores utilizavam uma interface MIDI / Serial. Hoje os aparelhos mais modernos saem de fábrica com uma porta USB, através da qual é possível conectá-los a computadores e enviar ou receber instruções MIDI reproduzindo ainda 16 canais. Na ligação sintetizador-sintetizador o único modo continua sendo o cabo MIDI padrão que pode ser ligado nas portas OUT e IN ou somente uma das duas, de acordo com a necessidade do usuário. Os instrumentos ligados ficam na condição de senhor e escravo, onde o senhor que é quem dá a ordem é o controlador MIDI e o escravo que é quem executa a ordem é um *sampler*.

Controladores MIDI são instrumentos capazes de emitir e interpretar arquivos MIDI. Esses controladores geralmente têm o formato de um teclado, mas existem dos mais diversos tipos como flauta, acordeon e baterias conforme mostra a figura 2.

---

[6] http://www.midi.org

[7] http://www.amei.or.jp/index_e.html

Figura 2 – Bateria Roland TD-6K que utiliza protocolo MIDI

Fonte: http://www.roland.com/products/en/TD-6K

Como o formato MIDI é um protocolo universal os controladores entendem a mensagem, a interpretam e executam de acordo com os recursos disponíveis.

Através do protocolo MIDI, é possível fazer que uma nota tocada em um controlador seja reproduzida em tempo real num computador ou módulo (sintetizador ou *sampler*[8]). Basicamente uma mensagem MIDI é enviada por um controlador (um teclado, por exemplo) e do outro lado processada por outro equipamento (a placa de som de um computador). Na verdade, o que o controlador faz é dar uma "ordem" para que o sintetizador produza uma nota musical. É como se o controlador dissesse: "toque a nota Fá por 8 tempos a um andamento de 100 batidas por minuto (BPM), com volume 80, utilizando timbre de violino". As mensagens MIDI não possuem o timbre em si, o mesmo é produzido no sintetizador. Por esse motivo que uma mesma mensagem midi com certeza soará diferente em seu computador e em um módulo. Nos computadores, a sintetização de um som midi se dá através da placa de som, ou emulados através do sistema operacional. Portanto, quanto melhor a placa, mais qualidade terá o som reproduzido.

2.1.3.2 Formato de arquivo MIDI

Como já foi especificado, os arquivos MIDI não transmitem a música propriamente dita, mas apenas mostram como ela deve ser executada. Para efeito de comparação ilustramos duas formas de entrada de MIDI para poder distinguir a diferença

[8] Equipamento ou programa capaz de armazenar sons em formato Wave e reproduzi-los posteriormente em sincronia com comandos MIDI.

entre elas.

Figura 3 – Pauta de partitura

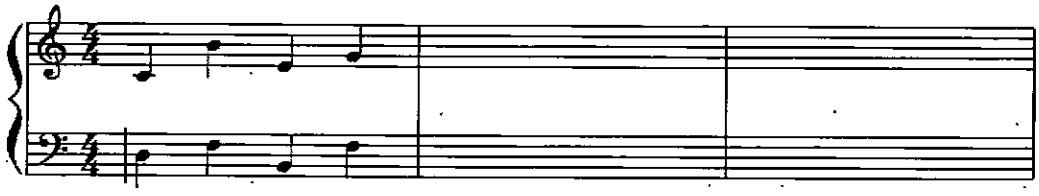


Figura 4 – Tabela de eventos MIDI

Event List - Track 1

Trk	Hr:Mn:Sc:Fr	Meas:Beat:Tick	Chn	Kind	Values		
1	00:00:00:00	1:1:000	1	Note	C 5	83	119
1	00:00:00:18	1:2:000	1	Note	E 5	95	119
1	00:00:01:06	1:3:000	1	Note	E 5	86	2:000
1	00:00:01:06	1:3:000	1	Note	C 5	101	2:000
1	00:00:01:06	1:3:000	1	Note	G 5	101	2:000

Fonte: Theophilo Augusto Pinto, 2011

A figura 3 apresenta uma partitura, usada pelos músicos e que pode ser escrita com qualquer editor de partitura disponível no mercado como o Sibelius ou Encore. Neste caso foi utilizado o Encore 5.0.0 (GVOX Corporation). Esses editores de partitura possibilitam ao músico gerar, a partir da partitura criada, um arquivo MIDI. É aí que aparece a figura 4 que mostra a conversão da partitura em eventos MIDI. Este é o modo como o arquivo MIDI é passado para o instrumento que vai interpretá-lo.

Essa tabela representa uma seqüência de *bytes* que serão lidos e interpretados pelo instrumento que irá gerar o som. A coluna Trk indica a pista onde será usado o som; Hr:Mn:Sc:Fr situa o evento MIDI na linha do tempo de execução da música; Meas:Beat:Tick apresenta o compasso, tempo de compasso e *tick* do evento; Cnh é o canal por onde sai o som (por padrão uma arquivo MIDI suporta até 16 canais e cada canal pode conter mais de uma pista de execução); *Kind* é o tipo de evento a ser tratado, no exemplo acima é passada uma nota; na coluna *Values* é apresentado o nome da nota, sua velocidade e duração.

As informações passadas por esta tabela são entendidas universalmente por todos os aparelhos capazes de gerar som que vão desde os mais potentes sintetizados e *samplers* a computadores *desktop* e celulares.

2.2 Dispositivos Móveis

Com a propagação dos dispositivos móveis e agregação de novos recursos de

hardware aos mesmos, novas possibilidades foram criadas para se trabalhar com esses dispositivos. Informações que antes só eram possíveis com computadores *desktops* hoje estão acessíveis em quase todo lugar nos dispositivos móveis.

Conceitua-se dispositivo móvel como "qualquer equipamento ou periférico que possa ser transportado com conteúdo e esteja acessível em qualquer lugar." (FIEMG, 2007, p. 01). Em comum eles têm o tamanho reduzido, memória limitada, poder de processamento limitado, baixo consumo de energia e conectividade limitada. De acordo com Oracle (2012) pode-se classificar uma série de aparelhos como dispositivos móveis divididos em três níveis principais.

- Terceiro nível: laptops;
- Segundo nível: PDAs;
- Primeiro nível: celulares.

Inicialmente os celulares serviam apenas aos serviços de telefonia móvel. O DynaTAC 800X, fabricado pela Motorola em 1983, "pesava 1 kg, tinha autonomia de 1h de conversação e 8h em *stand by*" (OLIVEIRA 2011, p. 3). Quase trinta anos depois o celular evoluiu e hoje os chamados *smartphones* que são os celulares mais avançados possuem um sistema operacional, recursos de conectividade, recursos de multimídia, câmera, suporte a aplicativos e alguns com *touch-screen.* Hoje *smartphones* como o Nokia 500 dispõem de um processador de 1Ghz, conexão Wifi, Bluetooth, editor de textos, vídeos e imagens, ou seja, superior a um PC do final da década de 90.

2.2.1 Sistemas operacionais para *smartphones*

Dois sistemas operacionais tem se destacado no mercado de *smartphones*: o *Symbian* e o *Android.* O *Symbian* nasceu a partir da união entre as empresas Nokia, Ericsson, Motorola e Psion. Segundo Schleuss (2010), esta última possuía um sistema operacional chamado de *EPOC32*, que era utilizado nos *PDAs* produzidos pela empresa. No decorrer dos anos, novas empresas entraram no consórcio, como Samsung, Matsushita e várias outras. Ele é um sistema multitarefa que possui uma interface gráfica intuitiva, além de suportar várias plataformas para desenvolvimento de aplicativos móveis. De acordo com Caraciolo (2010) esse sistema operacional, ao contrário do *Windows Mobile* que é uma adaptação do *Windows* para aparelhos móveis, foi desenvolvido especialmente para ser usado nos *smartphones*. Além disso, ele possui suporte a várias plataformas de desenvolvimento como JavaME/MIDP, C++, dentre outros.

O sistema operacional foi criado levando em consideração as seguintes premissas (MORRIS, 2007):

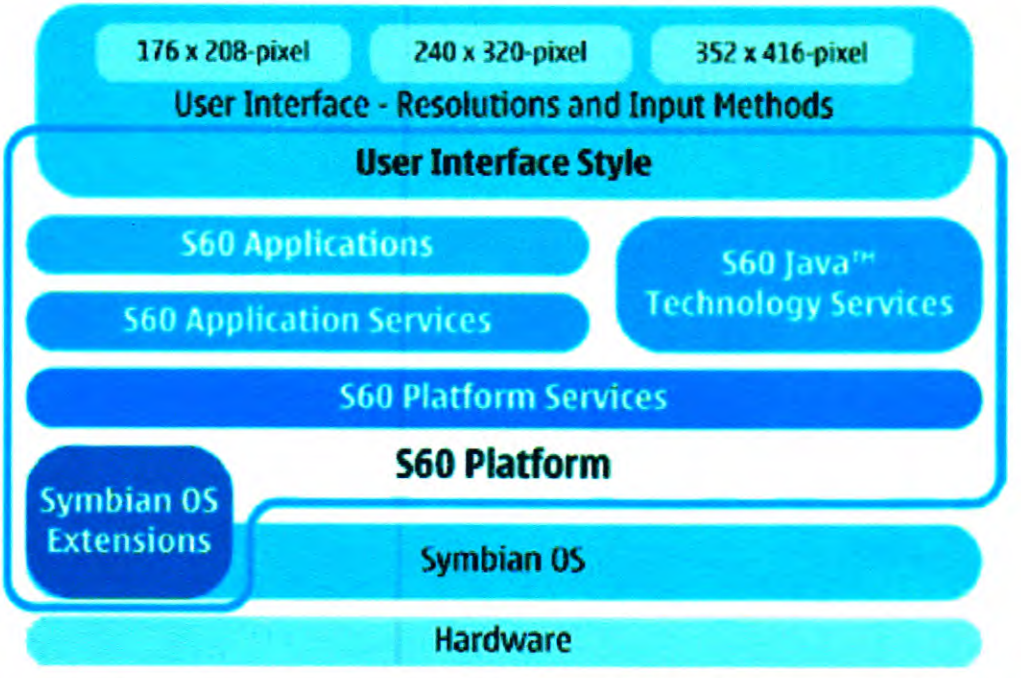
- Os dados do usuário são sagrados;
- O tempo do usuário é precioso;
- Todos os recursos são escassos.

Para atender a estas premissas, o *Symbian* apresenta algumas características:

- Todos os recursos são disponibilizados por intermédio de servidores;
- Todos os serviços são assíncronos, sendo necessário o uso de um modelo de *callback* para invocação de funções;
- Há uma separação rigorosa entre interface do usuário e os serviços que fornecem dados para as interfaces;
- Concepção de funcionalidades sempre pensando em reuso;
- Funcionamento em tempo real, atendendo prontamente a todas as solicitações;
- Concepção para ser extensível e de fácil modificação.

A plataforma S60 garante aos desenvolvedores que determinados elementos estarão presentes em qualquer dispositivo de uma mesma edição e no pacote de recursos da plataforma. A na figura 5 mostra a arquitetura do *Symbiam*.

Figura 5 – Arquitetura do Symbiam



Fonte: http://www.developer.nokia.com/Community/Wiki/Plataforma_Series_60

A versão mais recente desse sistema operacional é o *Symbian Belle* que agora se chama apenas *Nokia Belle*. Lançado em 2011 ele dá suporte aos aplicativos desenvolvidos na

parceria entre a Nokia e a Microsoft.

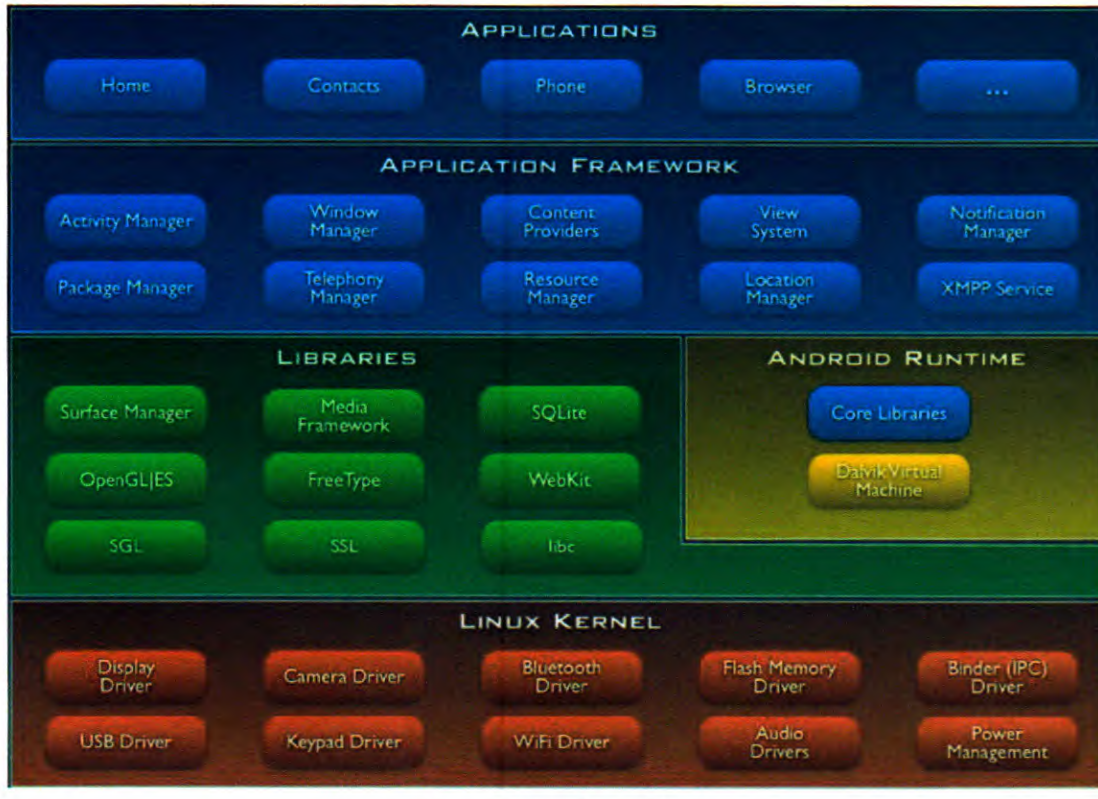
Android é um conjunto de softwares que possui um sistema operacional e um conjunto de aplicações chave. Começou a ser desenvolvido em 2007 por um grupo de empresas denominado *Open Handset Alliance* (OHA) que trabalharam juntas para oferecer uma plataforma aos desenvolvedores, de modo que os mesmos pudessem implementar suas aplicações aos seus dispositivos móveis. Atualmente a Google lidera o chamado *Android Open Source Project* (AOSP) que é responsável pelo desenvolvimento e manutenção do *Android.*

O *Android* hoje é baseado no *kernel* do *Linux* na versão 2.6 compartilhando com este algumas pastas do sistema de arquivos (como pode ser visto na figura 7), mas com outras que não se encontram no *Linux*, como a pasta */cache* que é um tipo de arquivo temporário que pode ser lido rapidamente pelo sistema.

Segundo Gomes (2011, p. 101), "a plataforma Google Android é a primeira plataforma aberta, completa e livre para dispositivos móveis".

A figura 6 mostra os principais componentes deste sistema operacional divididos nos seus quatro níveis.

Figura 6 - Arquitetura do sistema Android



Fonte: Google, 2011.

Na base do *Android* está o *Linux Kernel* que tem os serviços essenciais do sistema

e faz a abstração entre o hardware e os demais níveis do software. O nível acima é onde estão as bibliotecas básicas do *Android* e a máquina virtual *Dalvik* que roda as aplicações. O *Application Framework* é o gerenciador das funções básicas do dispositivo. E por fim estão as aplicações que é o que o usuário comum vê. As camadas inferiores somente os fabricantes e desenvolvedores de softwares é que têm acesso.

2.2.2 JAVA e J2ME

JAVA é uma linguagem de programação que foi lançada nos anos 90 pela *Sun Microsystems.* Inicialmente ela foi desenvolvida tendo em vista os progressos da intemet já que a mesma estava se popularizando muito. Segundo Oliveira (2011) ela não é apenas uma linguagem de programação, mas todo um conjunto de elementos que está em todo o universo da computação com duas características principais:

- O código-fonte da linguagem é interpretado pela *Java Virtual Machine* (JVM) o que toma o código independente de plataforma.
- Todas as tecnologias abrangem um conjunto amplo de *Application Programming Interface* (API) básica da linguagem distribuídas em pacotes principalmente no *java.lang* e *java.io.*

A *Sun,* vendo a necessidade dos mais diversos usuários de Java criou versões distintas da sua ferramenta e a dividiu em três edições específicas com características voltadas para cada grupo em particular.

- ***Java 2 Standar Edition* (J2SE):** com tecnologias voltadas para *desktops.* Contém o conjunto de ferramentas básicas para se desenvolver aplicativos Java assim como as APIs para interface gráfica, multimídia, redes, etc.
- ***Java 2 Micro Edition* (J2ME):** focada no desenvolvimento para dispositivos móveis. Esta edição se diferencia das demais pelo uso de uma máquina virtual chamada *Kilo Virtual Machine* (KVM) ao invés da JVM clássica. J2ME engloba grande parte das APIs da J2SE.
- ***Java 2 Enterprise Edition* (J2EE):** para aplicações empresariais. Esta edição está voltada para o mercado corporativo. Suporta aplicações que necessitem de autenticação, segurança, tráfico de dados intenso. Comporta *JavaServer Pages* (JSP), *eXtensible Markup Language* (XML) e *servlets.*

Para o desenvolvimento deste protótipo será utilizado a J2ME, pois é a versão que oferece suporte à programação para dispositivos de pequeno processamento.

Segundo Romeiro (2005) J2ME é a versão da linguagem Java que dá suporte ao desenvolvimento de aplicações para dispositivos com capacidades reduzidas tanto na parte gráfica quanto de memória e processamento. Esta edição nasceu em 1999 e nada mais era do que uma versão da JMV para ser executada em dispositivos Palm.

Para desenvolver em J2ME devemos levar em consideração três componentes principais: configurações, perfis e pacotes opcionais.

a) Configuração (*Configurations*)

Este componente é de extrema importância para o desenvolvedor, mas totalmente transparente ao usuário final. A configuração define uma série de requisitos que o sistema deve suportar para garantir a sua operacionalidade com o Java. De acordo com a Sun, nela encontramos a linguagem do dispositivo, maquina virtual e APIs suportadas pelo mesmo. Está dividida em dois padrões: o *Connected, Limited Device Configuration* (CLDC) e o *Connected Device Configuration* (CDC).

- CLDC: a Configuração de Dispositivos de Conexão Limitada é voltada para aparelhos com poucos recursos, que operam por baterias, considerados de uso pessoal como celulares e *pagers*. Nesses dispositivos a conexão de rede é limitada, eles têm entre 160 e 512 KB de memória ROM e dispõem de pelo menos 122 KB para rodar a máquina virtual.
- CDC: a Configuração de Dispositivos Conectados foi desenvolvida tendo em vista os dispositivos que suportam todas as funcionalidades da máquina Java. Para isso devem contar com 2 MB de memória ROM, 256 KB de memória RAM e conexão de rede. Nesse grupo encontram-se conversores de TV a cabo, alguns aparelhos de GPS e alguns PDAs. A CDC propicia um ambiente mais próximo da JSE.

b) Perfil (*Profiles*)

Entende-se como perfil a estrutura básica necessária para se desenvolver aplicações para dispositivos de uma mesma família, ou seja, ela supre as APIs e estruturas necessárias para dar suporte a dispositivos com as mesmas funcionalidades.

O perfil é uma estrutura acima das configurações. Enquanto as configurações especificam uma base de bibliotecas, o perfil dispõe dessas bibliotecas que são importantes para a criação de aplicações tais como bibliotecas de interface com o usuário, rede, multimídia, etc.

Vários perfis estão sendo desenvolvidos e estes são agrupados de acordo com a configuração para a qual se destinam. Voltados para o CLDC temos o *Mobile Information Device Profile* (MIDP) e o *Information Module Profile* (IMP) e para CDC os perfis *Foundation Profile* (FP), o *Personal Profile* (PP) e o *Personal Basis Profile* (PBP).

2.2.3 MIDP

O Perfil de Informações para Dispositivos Móveis surgiu na *Java Specifications Request* (JSR) 37 e é construída com base na configuração CLDC. É hoje o perfil mais utilizado pelos desenvolvedores para aplicações móveis, pois contém uma série de APIs que atende os dispositivos mais utilizados hoje em dia como celulares e PDAs.

Segundo Carniel e Teixeira (2003) para enquadrar nas especificações do MIDP os dispositivos devem ter no mínimo 128 KB de memória ROM para as APIs do MIDP, 8 KB de memória ROM para dados, 32 KB de memória RAM para a máquina virtual; display de 96 x 64 *pixels*; teclado de uma mão, duas mãos ou *touch screen*; acesso à rede podendo ser com largura limitada; capacidade de reprodução de sons.

Desde o seu lançamento em 2000 o MIDP tem três versões que foram surgindo de acordo com a evolução do hardware dos dispositivos: a) a MIDP 1.0 já está praticamente em desuso, já que só celulares antigos a suportam; b) a MIDP 2.0 lançada em 2002 incorpora APIs de rede, Bluetooth, tratamento de imagens 2D e 3D, segurança, etc; c) a MIDP 3.0 vem com um tratamento melhor de imagens 3D, novas opções para menu, além de permitir a execução de múltiplas MIDlets ao mesmo tempo.

Para o desenvolvimento do protótipo optou-se pela configuração CLDC versão 1.1. e o perfil MIDP na versão 2.0, pois estas versões estão presentes na maioria dos aparelhos em uso atualmente, tanto *smartphones* como os aparelhos mais básicos do mercado.

Um aplicativo desenvolvido com a MIDP recebe o nome de MIDlet. Ele é uma classe que é estendida da classe *javax.microedition.midlet.MIDlet*, e deve ter os métodos *startApp* – quando o aplicativo encontra-se em execução, *pauseApp* – aplicativo colocado em pausa ou a espera de uma requisição do usuário ou necessidade do próprio dispositivo e *destroyApp* – encerra os aplicativos e libera os recursos que estavam sendo utilizados. Estes métodos definem o ciclo de vida da aplicação. O MIDlet deve estar ligado diretamente ao AMS (*Application Management Software*).

A MIDP 2.0 trouxe várias inovações em relação à MIDP 1.0. Dentre as novidades chama a atenção uma API mais eficiente voltada para segurança e gerenciamento e uma API

para tratamento de sons e eventos de jogos.

No tratamento de sons a MIDP 2.0 implementa um *Audio Building Block* (ABB) que de acordo com a Sun deixa o desenvolvedor livre da *Mobile Media API* (MMAPI) para colocar sons em Wave ou simplesmente gerarem tons básicos para suas aplicações. Para trabalhar com sons MIDI a J2ME dispõe do método *playTone()* que faz parte do pacote *javax.microedition.media.Manager*. Este método aceita três parâmetros inteiros: o primeiro para a nota a ser tocada (0 a 127), o segundo para a duração e o terceiro para o volume da nota (0 a 100).

2.2.4 Trabalhos correlatos

São apresentados dois trabalhos correlatos ao trabalho proposto que são: um protótipo de software para treinamento auditivo de músicos em dispositivos móveis utilizando JME (KESTRING, 2009); e um protótipo de um sistema para auxílio da percepção musical (TOMEDI, 2002).

2.2.4.1 Protótipo de software para treinamento auditivo em dispositivos móveis utilizando JME

O sistema desenvolvido por Kestring (2009) é um aplicativo para ser usado em dispositivos móveis com suporte a linguagem Java. No trabalho de Kestring é feito um estudo de teoria musical dando particular atenção ao modelo inteiro das alturas e das tecnologias MIDI e JME.

Este trabalho é voltado para o treinamento da percepção musical do usuário não dando ênfase à parte de aprendizado da teoria musical. O protótipo baseia-se no reconhecimento de sons tocados pelo dispositivo através de exercícios de escalas e acordes.

2.2.4.2 Protótipo de um sistema de auxílio para percepção musical

O protótipo proposto por Tomedi (2002) é implementado em *Delphi* para ser usado em *desktops*. Tomedi também faz um estudo da teoria musical dando ênfase no modelo inteiro das alturas e na teoria MIDI. O sistema é baseado na criação de parâmetros da teoria musical aplicados com o auxílio da tecnologia MIDI. A aplicação da geração de tons MIDI nesses dois trabalhos tem a vantagem de não ser necessário criar um banco de dados para

acordes, escalas ou tons já que tudo pode ser criado dinamicamente.

São aplicadas relações matemáticas no desenvolvimento da aplicação e dos exercícios. O sistema enfatiza o treinamento auditivo do músico que é obtido com exercícios aleatórios de escalas, acordes e intervalos. Assim como Kestring (2009), esse protótipo não dá ênfase à teoria musical.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são mostradas as principais fases para implementação do protótipo. Primeiramente serão mostrados os requisitos da aplicação (funcionais e não funcionais), será detalhada a especificação e por fim será mostrada a fase de implementação com suas técnicas e ferramentas utilizadas.

3.1 Requisitos

Nas fases de desenvolvimento de um produto talvez a mais complexa e de suma importância seja a engenharia de requisitos. Segundo Sommerville (2007) ela está relacionada com a definição propriamente dita do sistema, suas propriedades desejáveis e essenciais, também define suas restrições quanto à operacionalidade. É nessa fase que alguns questionamentos precisam ser respondidos, tais como: o que o cliente deseja? Quais suas reais necessidades? Podemos dizer, portanto, que a engenharia de requisitos é o grande conciliador entre os clientes, os usuários do software e os desenvolvedores do mesmo. Essa fase não é apenas um mero processo técnico, mas é o norteador de todo o desenvolvimento do projeto podendo ser a causa tanto do sucesso quanto do fracasso do empreendimento.

O levantamento dos requisitos para o desenvolvimento do protótipo se deu a partir de vários estudos sobre a aplicação. Foi levado em consideração o tipo de aparelho a que era destinado, com suas limitações e possibilidades, os usuários e as possibilidades de implementação na linguagem escolhida.

a) Requisitos funcionais

Por requisitos funcionais entende-se que "são as declarações de serviços que o sistema deve fornecer" (SOMMERVILLE 2007, p. 80), como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também estabelecer o que o sistema não deve fazer.

O protótipo proposto deverá contemplar os seguintes requisitos funcionais:

- apresentar resposta à requisição do usuário;
- executar as notas que compõem o acorde e a escala;
- gerar questionário sobre intervalos;
- indicar se a resposta do usuário está certa ou errada;

b) Requisitos não-funcionais

São aqueles que de certo modo não estão relacionados diretamente às funções do sistema. A sua ligação com o sistema se dá por outros meios como "confiabilidade, tempo de resposta e espaço de armazenamento" (SOMMERVILLE 2007, p. 82). O principal foco dos requisitos não-funcionais não é o sistema como um todo mas algumas características do sistema.

O protótipo proposto deverá contemplar os seguintes requisitos não funcionais:

- apresentar uma interface que esteja de acordo com as configurações do dispositivo móvel, mas que seja amigável ao usuário;
- ser implementado utilizando a linguagem de programação J2ME;
- receber requisição do usuário via teclado ou *touch-screen*;
- ser implementado utilizando o perfil MIDP 2.0 e suas configurações;

3.2 Especificação

Para fazer a especificação deste trabalho foi usada a notação UML juntamente com a ferramenta StarUML[9] versão 5.0.2.1570. StarUML é uma ferramenta *Open Source* que possibilita a criação tanto dos diagramas de Casos de Uso como diagramas de Classe e de Seqüência. Serão explorados os diagramas de casos de uso, seqüência e classes nesta aplicação.

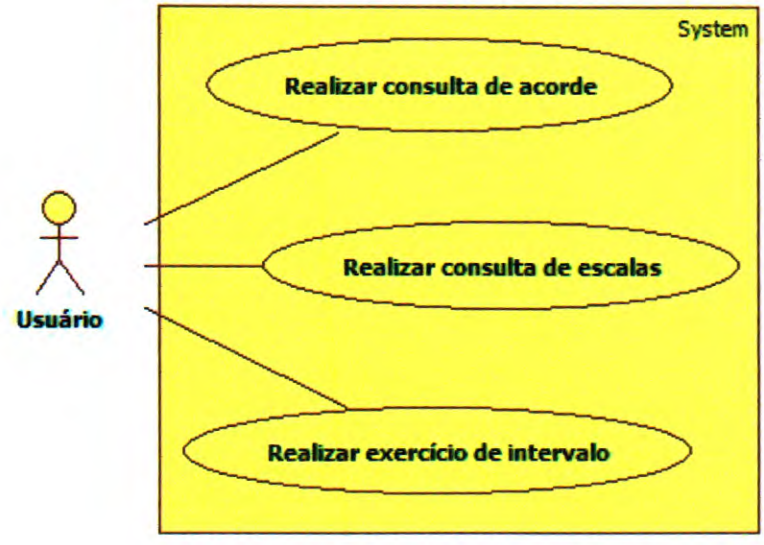
a) Diagrama de casos de uso

De acordo com Silva (2001) esses diagramas são construídos tendo por base os conceitos de ator, casos de uso e relacionamento. Os atores são representações dos objetos externos que interagem com o sistema. Essa entidade não representa um indivíduo, mas uma classe de indivíduos que interage com o sistema com as mesmas características. Os casos de uso são as funções do sistema de interação dos atores e os relacionamentos, as relações que podem ser entre sistemas ou entre sistema e casos de uso.

São especificados os três casos de uso do projeto: realizar consulta de acorde, realizar consulta de escalas e realizar exercício de intervalo.

[9] staruml-5-0-2.soft-free-download.com/PT/

Figura 7 – Casos de uso



O diagrama de casos de uso (figura 7) apresenta os três casos de uso do sistema. Para cada caso de uso são apresentados o cenário principal e os fluxos alternativos nos quadros a seguir.

Quadro 3 – Realizar consulta de acordes

Realizar consulta de acordes: possibilita ao usuário realizar o questionário de teoria musical sobre acordes	
Pré-condição	O usuário deve iniciar escolhendo a opção Acorde no menu inicial.
Cenário principal	1) O usuário seleciona a opção do tipo de acorde. 2) O sistema apresenta a resposta ao usuário. 3) O sistema fica em espera para as opções voltar ou tocar.
Fluxo Alternativo 01	No passo 2, caso o usuário opte por voltar, o sistema apresenta a tela para nova consulta.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por tocar a nota, o sistema permanece na tela do acorde e o dispositivo executa o acorde selecionado
Pós-condição	O sistema apresenta uma mensagem de fechamento.

Quadro 4 – Realizar consulta de escalas

Realizar consulta de escalas: possibilita ao usuário realizar o questionário de teoria

musical sobre escalas	
Pré-condição	O usuário deve iniciar escolhendo a opção escala no menu inicial.
Cenário principal	1) O usuário seleciona a opção do tom do acorde 2) O sistema apresenta a resposta ao usuário. 3) O sistema fica em espera por uma das opções ou voltar ou tocar a escala.
Fluxo Alternativo 01	No passo 2, caso o usuário opte por voltar, o sistema apresenta a tela de consulta de escala.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por tocar, o sistema permanece na tela que se encontra e executa as notas da escala.
Pós-condição	O sistema apresenta uma mensagem de fechamento.

Quadro 5 – Realizar exercício de intervalo

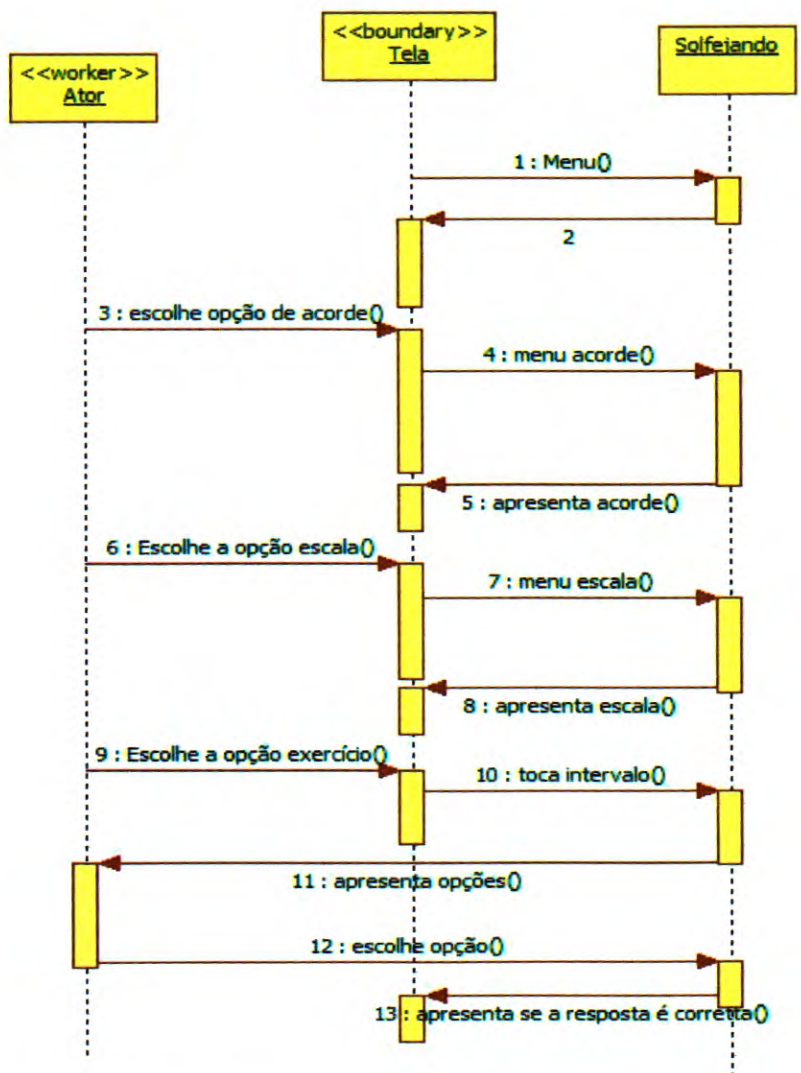
Realizar exercício de intervalo: possibilita ao usuário realizar o exercício de treinamento de teoria musical sobre intervalos.	
Pré-condição	O usuário deve iniciar escolhendo a opção exercício no menu inicial.
Cenário principal	1) O sistema executa os sons de um determinado intervalo 2) O sistema apresenta várias opções de resposta ao usuário. 3) O sistema fica em espera para que o usuário escolha entre as opções voltar, tocar ou ok.
Fluxo Alternativo 01	No passo 2, caso usuário escolha ok, o sistema verifica se a alternativa é correta e informa ao usuário se ele acertou o exercício ou não.
Fluxo Alternativo 02	No passo 3, caso o usuário opte por tocar, o sistema permanece na tela que se encontra e executa as notas do intervalo.
Fluxo alternativo 03	No passo 4 caso o usuário opte por voltar, o sistema apresenta o menu inicial.
Pós-condição	O sistema apresenta uma mensagem de fechamento.

b) Diagrama de seqüência

O diagrama de seqüência apresenta a troca de mensagens entre diversos objetos da aplicação, numa situação específica e num tempo determinado. No diagrama de seqüência pode-se ver com clareza a ordem e os momentos nos quais as mensagens são enviadas para os objetos.

A seqüência da execução do protótipo é mostrada na figura 8 a seguir. Neste diagrama pode-se observar toda a interação do protótipo através das requisições feitas pelo usuário.

Figura 8 – Diagrama de seqüência



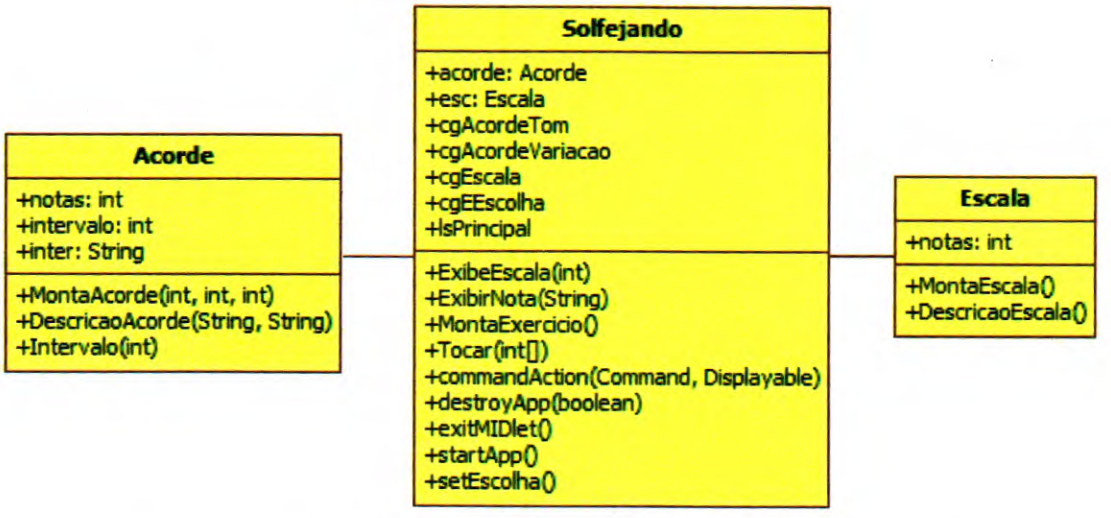
c) Diagramas de classes

As classes são de fundamental importância quando se utiliza a programação orientada a objetos. Conceitualmente, podemos entender as classes como descrições genéricas ou coletivas de entidades do mundo real, ou seja, elas definem os atributos e os métodos de

um conjunto de objetos. Segundo Silva (2001) o diagrama de classes mostra as diferentes classes de um sistema e como elas se relacionam.

O sistema implementado foi dividido em três classes: Escala, Acorde e Solfejando(que é a classe principal). A figura 9 a seguir mostra as classes do sistema exibindo os métodos das mesmas.

Figura 9 – Classes do Sistema



A classe Acorde tem como atributos um vetor de inteiros para as notas (notas: int[]), um vetor de inteiros para os intervalos (intervalo: int[]) e a variável de nome inter para os intervalos (inter: String). Os métodos dessa classe são:

- MontaAcorde(int, int, int): esse método recebe três inteiros como parâmetros. O primeiro de 52 a 63 dá o tom do acorde; o segundo recebe 1 ou 2 e determina o tipo do acorde entre maior e menor; o terceiro recebe 4 ou 7 e serve para determinar o complemento do acorde entre quarta e sétima.
- DescricaoAcorde(String, String): recebe como parâmetros duas *Strings*: uma para a descrição do acorde e a outra para guardar o nome do acorde selecionado.
- Intervalo(int): recebe um inteiro como parâmetro para ser a nota fundamental do intervalo e utiliza um método *random* para sortear a nota seguinte. O quadro 6 mostra o código de sorteio do método intervalo.

Quadro 6 – Sorteio no método intervalo

```
Random r = new Random();
int j = r.nextInt(4);
intervalo[0] = i + 52;
```

A classe Solfejando apresenta os seguintes atributos: quatro grupos de escolha (cgAcordeTom, cgAcordeVariacao, cgEscala e cgEEscolha); um atributo acorde do tipo Acorde, uma variável esc do tipo Escala, e uma lista apresentada na tela inicial (lsPrincipal). Aqui os métodos dessa classe que são:

- ExibeEscala(int): recebe um valor inteiro de 1 a 14 do cgEscala correspondente aos tons de C a B sem os sustenidos e bemóis (futuramente eles poderão ser incluídos);
- ExibirNota(String): recebe uma String correspondente à união da escolha feita em cgAcordeTom e cgAcordeVariacao;
- MontaExercicio(): monta exercícios de intervalo de modo aleatório;
- Tocar(): método usado para gerar os tons MIDI, tanto dos acordes, escalas como exercícios de intervalo;
- commandAction(Command, Displayable): é onde ocorre todo o processo do protótipo. Todos os métodos do MIDlet para fazer alguma coisa necessitam ter uma chamada no commandAction. O quadro 7 mostra o trecho do comando Ok do resultado do Exercício de intervalo.

Quadro 7 - Comando Ok

```
if (command == cmOkExercicio){
   String teste  = inter1 + " " + acorde.inter;
   String teste2 = cgEEscolha.getString(cgEEscolha.getSelectedIndex());
   if (teste.equals(teste2)){
      siExercicio.setText("Resposta correta!");
   }else{
      siExercicio.setText("Alternativa incorreta! "
            + "A alternativa correta é: " + teste);
   }
```

A classe Escala tem como atributo um vetor de inteiros para guardar a escala (notas: int[]) e os seguintes métodos:

- MontaEscala(int, int): recebe dois inteiros vindos a partir das escolhas feitas cgEscala. O método monta a escala e retorna as notas para serem executadas.
- DescricaoEscala(String, int): recebe como parâmetros uma String para guardar a descrição da escala e um inteiro para o tipo de escala (maior ou menor)

3.3 Implementação

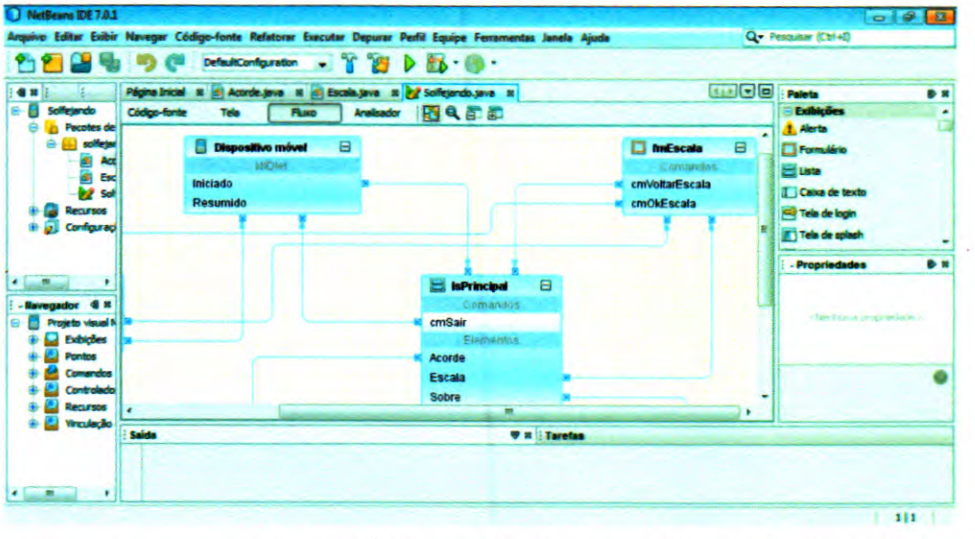
Depois de feito todo o levantamento bibliográfico foi possível desenvolver um protótipo que mostrasse os acordes, escalas e exercícios de intervalo reproduzindo o som dos mesmos para os usuários.

Para o desenvolvimento deste protótipo foi utilizada a linguagem de desenvolvimento Java, mas especificamente J2ME, que foi escolhida pelos seguintes aspectos, dentre outros:

- Um vasto suporte dos fabricantes de dispositivos móveis;
- Grande disposição de material para pesquisa;
- Disponibilidade de APIs "universais" sem necessidade de grandes adequações de um aparelho para outro;

No desenvolvimento do trabalho foi usado o NetBeans IDE 7.0.1[10]. O NetBeans está licenciado sob a *Common Development and Distribution License* (CDDL) e a *General Public License* (GNU), podendo ser utilizado para desenvolvimento de projetos tanto pagos quanto gratuitos. Ele foi escolhido devido a sua interface mais intuitiva que a do Eclipse.

Figura 10 – Ambiente de desenvolvimento do NetBeans



Para os testes foi utilizado o Wireless Toolkit 2.5.2 licenciado pela Sun Microsystems. Este toolkit foi escolhido por dar suporte a uma gama de aparelhos genéricos aos mais diversos modelos do mercado. O uso do emulador é necessário no desenvolvimento deste tipo de aplicação para se corrigir os erros passo a passo. Pois possibilita que sejam feitos testes a cada etapa do desenvolvimento.

---

[10] http://netbeans.org/downloads/index.html

3.3.1 Realizar consulta de acordes

No quadro 8 é mostrado trechos do código do comando Ok para consulta de acordes e do método para montar os acordes. O método responsável por montar os acordes é o MontaAcorde(int, int, int). As variáveis inteiras para montar o acorde são capturadas no formulário fmAcorde. Esse formulário tem dois grupos de escolha (cgAcordeTom e cgAcordeVaricao) que ao serem escolhidos no commandAction são repassados como mostra o trecho do comando Ok do fmAcorde.

Quadro 8 – Consulta de acordes

```
if (command == cmOkAcorde) {
        /*
        * Pega o acorde e a variaçao selecionados
        * ac    = acorde
        * var   = variacao
        */
        String ac = cgAcordeTom.getString(cgAcordeTom.getSelectedIndex());
        String var = cgAcordeVariacao.getString(cgAcordeVariacao.getSelectedIndex());

        //...
        if ("C".equals(ac)){
                ac1 = 52;
        }

        //...
        if ("7".equals(var)){
                tipo = 1;
                comp = 7;


public void MontaAcorde(int tom, int tipo, int comp){
        if (tipo == 1){
                notas[0] = tom;
                notas[1] = tom + 4;
                notas[2] = tom + 7;
                if (comp == 4){ // quarta
                        notas[1] = notas[1] + 1;
                }
                if (comp == 7){ // setima
                        notas[3] = tom + 10;
                }

}
```

3.3.2 Realizar consulta de escalas

Para realização da consulta de escalas o método criado foi o MontaEscala(int, int). Este método recebe requisição do formulário fmEscala que tem um grupo de escolha com os tons C, Cm, D, Dm, E, Em, F, Fm, G, Gm, A, Am, B e Bm. Estes tons foram escolhidos por serem os mais usados na música. Em versões futuras poderão ser adicionados os tons com sustenidos e bemóis. O método para montar escala tem seu código descrito no quadro 9.

Quadro 9 – Método para montar escalas

```
public void MontaEscala(int tom, int tipo){
        if (tipo == 1){        // escala maior
                notas [0] = tom;
                notas [1] = tom + 2;
                notas [2] = tom + 4;
                notas [3] = tom + 5;
                notas [4] = tom + 7;
                notas [5] = tom + 9;
                notas [6] = tom + 11;
                notas [7] = tom + 12; // uma oitava acima
        } else  if (tipo == 2) { // escala menor
                notas [0] = tom;
                notas [1] = tom + 2;
                notas [2] = tom + 3;
                notas [3] = tom + 5;
                notas [4] = tom + 7;
                notas [5] = tom + 8;
                notas [6] = tom + 10;
                notas [7] = tom + 12; // uma oitava acima
        }
}
```

3.3.3 Exercícios de intervalo

Para gerar os exercícios de intervalo foram criados dois métodos: uma na classe Solfejando – MontaExercicio(), e outro na classe Acorde – Intervalo(). O método MontaExercicio() é chamado no commandAction(Command, Displayable). Ele sorteia aleatoriamente um número de 1 a 13 e repassa para ser montado o intervalo no método Intervalo(). Esse método recebe o número sorteado anteriormente e soma com a nota fundamental C (52). Depois é feito o sorteio de um outro número entre 1 e 4 para compor o intervalo a ser tocado. O quadro 10 mostra o código do método Intervalo().

Quadro 10 – Método para compor intervalos

```
public void Intervalo(int i){
        Random r = new Random();
        int j = r.nextInt(4);
        intervalo[0] = i + 52;
        switch (j){
                case 1:{
                        intervalo[1] = intervalo[0] + 4;
                        inter = "3";
                }
                case 2:{
                        intervalo[1] = intervalo[0] + 5;
                        inter = "4";
                }
                case 3:{
                        intervalo[1] = intervalo[0] + 7;
                        inter = "5";

                case 4:{
                        intervalo[1] = intervalo[0] + 11;
                        inter = "7";

        }
}
```

No código o sorteio é feito utilizando o método Random() da classe *java.util.Random*. O número sorteado é armazenado em um vetor inteiro de duas posições, onde o primeiro número é aquele sorteado anteriormente em MontaIntervalo().

Todas as notas que serão tocadas no sistema são repassadas para o método Tocar(int[]). Esse método utiliza a classe *javax.microedition.media.Manager* para gerar os tons MIDI. O código deste método está descrito no quadro 11.

Quadro 11 – Método Tocar

```
public void Tocar(int[] note){
        for (int i = 0; i < note.length; i ++){
                try {
                        Manager.playTone(note[i], 600, 100);
                }catch (MediaException me){
                }
        }
}
```

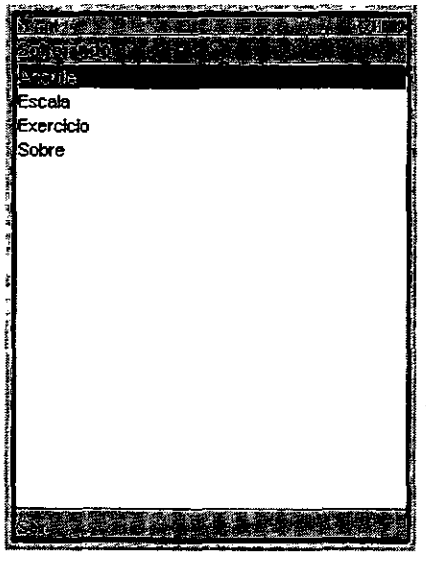
3.4 Operacionalidade

Nesta seção, estão descritos os processos de operacionalização do sistema com as suas funções. Nela serão apresentadas as principais funções do sistema na sua interatividade com o usuário.

3.4.1 Consulta de Acordes

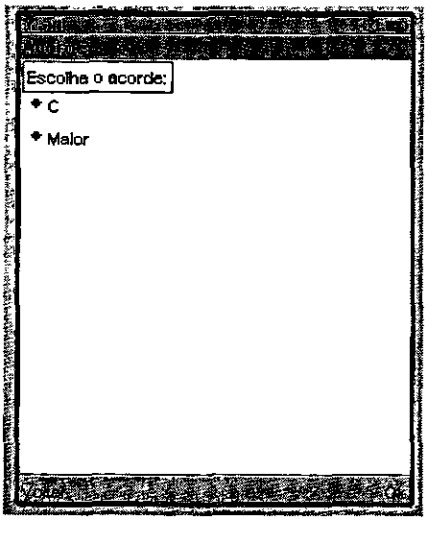
Na tela de Menu (figura 11) o usuário tem a opção de escolher entre Acorde, Escala, Exercício e Sobre. Ao selecionar uma delas será encaminhado para a tela desejada.

Figura 11 – Menu inicial



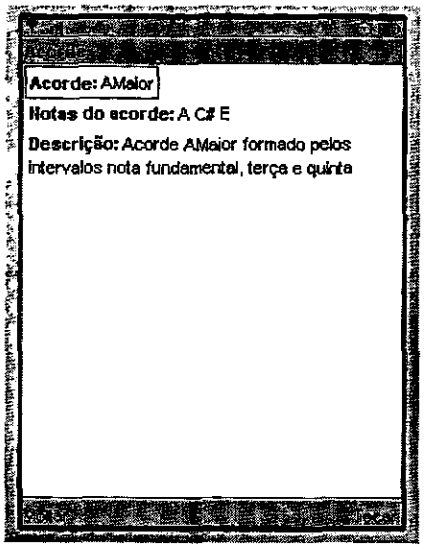
Ao escolher a opção Acorde o usuário é encaminhado para a tela de seleção do acorde desejado conforme a figura 12.

Figura 12 – Seleção de acorde



Nesta tela o usuário seleciona o tom do acorde entre C, C#, D, D#, E, F, F#, G, G#, A, A# e B. E depois escolhe o complemento do acorde Maior, m, 7, m7 ou 4. Ao escolher o acorde o usuário é encaminhado para a tela onde mostra o tom escolhido (figura 13), as notas que compõem o tom e as opções de Tocar – ouvir o acorde, ou Voltar para uma nova consulta de acorde.

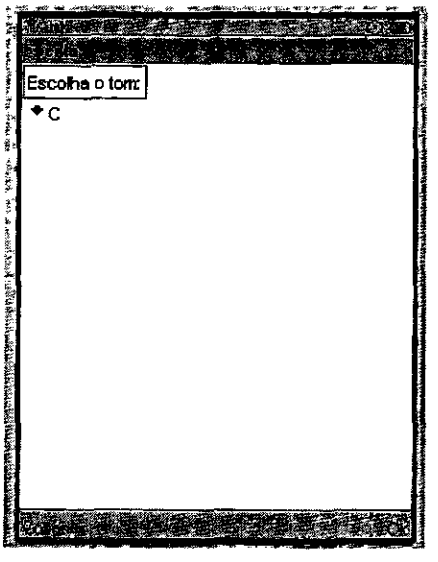
Figura 13 – Exibir acorde



3.4.2 Consulta de Escalas

Ao escolher a opção de Escala no Menu inicial o usuário será direcionado para a tela de consulta de escala, conforme a figura 14.

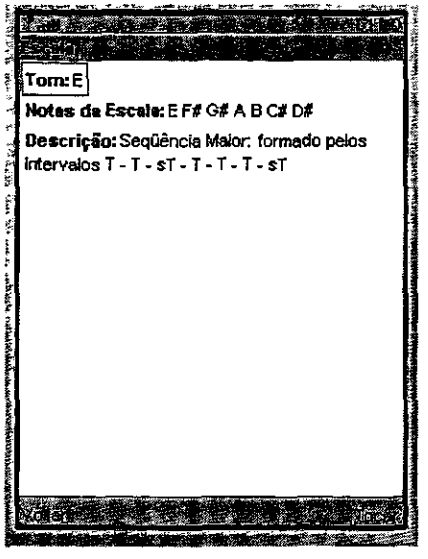
Figura 14 – Escolha de escala



Essa tela só tem o campo onde o usuário escolhe o tom entre C, Cm, D, Dm, E,

Em, F, Fm, G, Gm, A, Am, B e Bm, podendo futuramente ser implementado também os acordes com sustenidos e bemóis. Ao selecionar o item desejado para a escala uma nova tela se abrirá como mostrado na figura 15. Nesta estará descrito o tom escolhido e as notas que compõem a escala. O usuário tem à sua disposição as opções de Voltar e Tocar – ouvir as notas da escala.

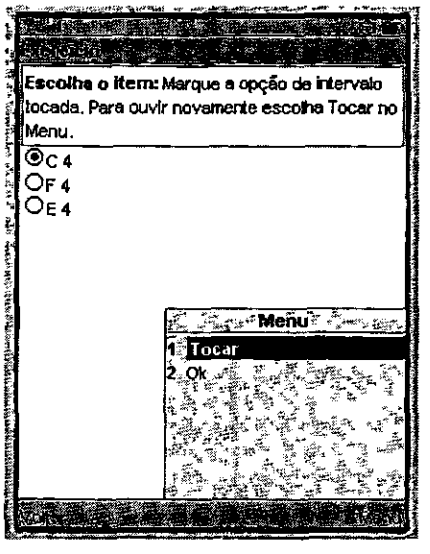
Figura 15 – Exibir Escala



3.4.3 Exercício de Intervalo

Ao selecionar a opção de Exercício o sistema tocará um intervalo de duas notas e se abrirá uma tela com a opção de seleção para a resposta correta para o intervalo, como mostra a figura 16.

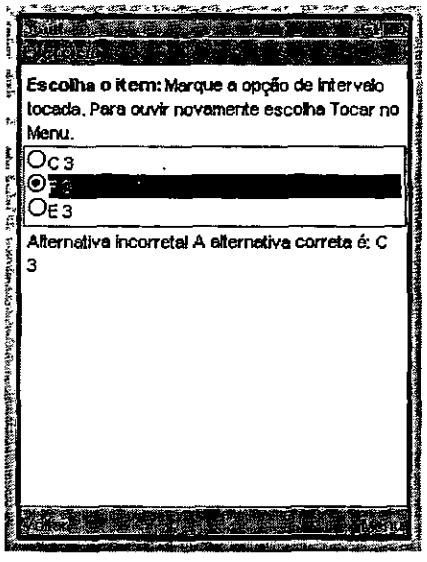
Figura 16 – Questionário de exercício de intervalo



Nesta tela o usuário tem as opções Voltar, Tocar e Ok. Ao escolher Voltar o

sistema abrirá o Menu Inicial; escolhendo Tocar o sistema executará novamente as notas do intervalo e escolhendo Ok o sistema verifica a opção escolhida pelo usuário no *ChoiceGroup* e indica se a resposta é correta ou não.

Figura 17 – Resultado do exercício de intervalo



A figura 17 mostra o resultado da escolha do usuário. Como a escolha não está correta o sistema apresenta uma mensagem indicando a alternativa certa para o intervalo.

4 CONSIDERAÇÕES FINAIS

O objetivo principal deste trabalho foi proporcionar aos usuários um dicionário de acordes voltado para a teoria musical. Foram estudados os conceitos fundamentais, assim como as técnicas adequadas para o desenvolvimento deste trabalho. Além disso, foi feita a análise e demonstração do funcionamento do protótipo.

Concluída a proposta inicial recomenda-se que sejam feitos testes em aulas de música, principalmente em escolas onde o ensino de música já é obrigatório. Como a maioria dos estudantes possui um aparelho celular, este vai ser mais um recurso que o professor poderá usar nas suas aulas principalmente porque o mesmo dá grande ênfase à teoria musical.

Diferentemente de outras aplicações existentes no mercado, o protótipo desenvolvido dá mais importância à teoria musical que, espera-se, fará com que o aluno aprenda o que está tocando em detrimento de apenas decorar posições dos dedos para fazer os acordes e as escalas.

A plataforma Java se mostrou muito eficaz para desenvolver esta aplicação, dando muitas possibilidades de complementação ao sistema. Como trabalhos futuros propõe-se a melhoria da interface gráfica, a possibilidade dos usuários inserirem novos acordes não conhecidos pelo sistema, bem como a possibilidade do usuário poder escolher o som de que instrumento deseja ouvir, como por exemplo flauta, violão, piano e cordas.

(UFPE). Disponível em: <http://www.tiagobarros.org/docs/wjogos2003-j2me-brew-symbian.PDF>. Acesso em: 10 jun. 2011.

KESTRING, Marcelo Ricardo, **Protótipo de software para treinamento auditivo de músicos em dispositivos móveis utilizando JME.** Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau: 2009. Disponível em: <http://www.inf.furb.br/tcc/index.php?cd=6&tcc=1184>. Acesso em: 05 mai. 2011.

LOUREIRO, Maurício A.; PAULA, Hugo B. de. **Timbre de um instrumento musical: caracterização e representação.** Per Musi – Revista Acadêmica de Música – n.14, 110 p., jul - dez, 2006. p. 57 – 81. Disponível em: <http://www.musica.ufmg.br/permusi/port/numeros/14/Num14_cap_05.pdf>. Acesso em: 20 jun. 2011.

MMA - MIDI Manufacturers Association. **The Complete MIDI 1.0 Detailed Specification.** Disponível em: <http://www.midi.org/techspecs/midispec.php>. Acesso em: 20 jan. 2012.

MILETTO, E. M., et al. **Minicurso: introdução à computação musical.** In: CBCOMP - CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 4., Itajaí, 2004. "Anais...", Itajaí, 2004, p. 883-902.

MOORE, Martin L. **SoundBlaster: o livro definitivo.** Rio de Janeiro: Campus, 1994.

MORRIS, Ben. **The Symbian OS Architecure SourceBook – Design and evolution of a Mobile OS.** Chichester: John Wiley & Sons, 2007. Disponível em: <http://www.4shared.com/get/BxsuQcpQ/WileyTheSymbianOSArchitectureS.html>. Acesso em: 20 out. 2011.

Oracle. **Mobile Information Devie Profile (MIDP); JSR 118.** Disponível em: <http://www.oracle.com/technetwork/java/index-jsp-138820.html>. Acesso em: 10 jan. 2012.

____. **Connected Limited Device Configuration (CLDC); JSR 139.** Disponível em: <http://java.sun.com/products/cldc/>. Acesso em: 12 jun. 2012.

OLIVEIRA, Rogério Coimbra de. **Dispositivos móveis portáteis: tecnologias envolvidas no desenvolvimento de aplicativos para smartphones.** Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Unidade Universitária de Ciências Exatas e Tecnológicas, Universidade Estadual de Goiás, Anápolis 2011. Disponível em: <http://pt.scribd.com/doc/74798706/Monografia-DISPOSITIVOS-MOVEIS-PORTATEIS>. Acesso em: 05 jul. 2012.

PINTO, Theophilo Augusto. **MIDI – Tipos de mensagens.** Disponível em: <http://www.musicaudio.net/midi/tipos_de_mensagens_midi.htm>. Acesso em: 10 jan. 2012.

Plataforma Series 60. Disponível em: <http://www.developer.nokia.com/Community/Wiki/Plataforma_Series_60>. Acesso em: 03 jan. 2012.

PRESSMAN, Roger S. **Engenharia de software**. Tradução: José Carlos Barbosa dos Santos; revisão técnica: José Carlos Maldonado, Paulo César Masiero, Rosely Sanches. São Paulo: Pearson Makron Books, 2009.

SCHLEUSS, Rafael. **Protótipo de software para gerenciamento de servidores Linux através de dispositivos móveis com sistema operacional Symbian S60.** Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau: 2010. Disponível em: <www.bc.furb.br/docs/MO/2010/344267_1_1.pdf>. Acesso em: 27 ago. 2012.

SEBESTA, Robert W. **Conceitos de linguagens de programação.** 5ª ed. Tradução: José Carlos Barbosa dos Santos. Porto Alegre: Bookman, 2003.

SILVA, Alberto Manuel Rodrigues da. VIDEIRA, Carlos Alberto Escaleira. **UML, Metodologias e Ferramentas Case**. Portugal: Edições Centro Atlântica, 2011. Disponível em: <http://www.centroatl.pt/titulos/tecnologias/imagens/uml-excerto-centro_atlantico.pdf > Acesso em: 25 mai. 2012.

SOMMERVILLE, Ian. **Engenharia de software.** 8ª ed. Tradução: Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa; revisão técnica: Kechi Kirama. São Paulo: Pearson Addison-Wesley, 2007.

Roland. Disponível em: <http://www.roland.com/products/en/TD-6K/>. Acesso em: 20 jun. 2012.

ROMEIRO, Bruna Georgina Bunzen de Albuquerque. **Desenvolvimento de aplicativos para dispositivos móveis na Plataforma J2ME.** Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação – Escola Politécnica de Pernambuco – Universidade de Pernambuco, Recife: 2005. Disponível em: <pt.scribd.com/doc/63757897/BrunaRomeiro > Acesso em: 10 jan 2012.

TOMEDI, R. A. B. **Protótipo de um sistema para auxílio ao treinamento da percepção musical**. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau: 2002. Disponível em: <http://www.inf.furb.br/tcc/index.php?cd=7&tcc=682>. Acesso em: 05 mai. 2011.

UEDA, Leo Kazuhiro, KON, Fabio. **Andante – composição e performance musical utilizando agentes móveis**. Disponível em: <gsd.ime.usp.br/andante/docs/andante-sub112003.pdf>. Acesso em: 28 mar. 2012.