

# TAGARELA: MÓDULO DE COMPOSIÇÃO MUSICAL POR MEIO DE MUSICOTERAPIA

Roberto Weege Junior, Dalton Solano dos Reis – Orientador

**Resumo:** *O resumo é uma apresentação concisa dos pontos relevantes de um texto. Informa suficientemente ao leitor, para que este possa decidir sobre a conveniência da leitura do texto inteiro. Deve conter OBRIGATORIAMENTE o OBJETIVO, METODOLOGIA, RESULTADOS e CONCLUSÕES. O resumo não deve ultrapassar 10 linhas e deve ser composto de uma sequência corrente de frases concisas e não de uma enumeração de tópicos. O resumo deve ser escrito em um único texto corrido (sem parágrafos). Deve-se usar a terceira pessoa do singular. As palavras-chave, a seguir, são separadas por ponto, com a primeira letra maiúscula. Caso uma palavra-chave seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]*

**Palavras-chave:** *Ciência da computação. Sistemas de informação. Monografia. Resumo. Formato.*

## 1 INTRODUÇÃO

A música é um elemento que estimula o ser humano. Segundo Queiroz (2000, p. 15), “a música tem como qualidade intrínseca de relaxar a sensoriedade humana, por satisfazê-la, e com isso conduz as pessoas a um estado receptivo e sensível, em especial quanto às emoções”. Tomando proveito desses efeitos, a musicoterapia é um processo, segundo Bruscia (2016, p. 55), em que o musicoterapeuta utiliza diversas facetas da experiência musical, além das relações interpessoais geradas durante a terapia, para otimizar a saúde do paciente.

Atualmente a musicoterapia é aplicada em diversos tipos de estabelecimentos, como escolas, clínicas, casas de repouso e asilos (BRUSCIA, 2016, p. 36). É uma prática utilizada para dar assistência a pessoas com distúrbios emocionais, transtornos psiquiátricos, necessidades especiais, dificuldades especiais, dependências químicas, estresse, pós-trauma, entre outros (BRUSCIA, 2016, p. 36). Nestas aplicações, diversos elementos musicais podem ser utilizados. Bruscia (2016, p. 59) destaca quatro métodos primários utilizados na musicoterapia: escutar, improvisar, recriar e compor. Estas são atividades em que o musicoterapeuta precisa envolver o paciente, sem necessariamente ensinar a ele teoria musical. Esta situação traz desafios que podem ser enfrentados mais facilmente com o auxílio de tecnologia.

A utilização de softwares como ferramenta para terapia ajuda os terapeutas a obter melhores resultados com seus pacientes. De acordo com Watanabe, Tsukimoto D. e Tsukimoto G. (2003, p. 20) quando foi utilizado um computador se verificou “[...] melhora funcional em todos os pacientes, com aumento da destreza e agilidade no uso dos programas e dispositivos”. Watanabe, Tsukimoto D. e Tsukimoto G. (2003, p. 20) também destacam que houve um aumento na motivação dos pacientes com a utilização do computador como ferramenta auxiliar de terapia. Um exemplo de software que pode ser aplicado em atividades terapêuticas é o Tagarela, que é uma plataforma que vem sendo desenvolvida através de trabalhos de conclusão de curso da área de computação da Universidade Regional de Blumenau. O Tagarela foi criado inicialmente como uma plataforma de comunicação alternativa, para auxiliar na melhora da capacidade de comunicação do paciente (FABENI, 2012). Outros módulos começaram a ser adicionados ao tagarela em outros trabalhos, como um módulo de jogo educacional (FERREIRA, 2016), um módulo de ensino de Braille (CAZAGRANDE, 2016), e um módulo destinado ao auxílio de crianças autistas na aquisição e desenvolvimento de linguagem (SAUTNER, 2017).

Considerando o exposto, este trabalho pretende ampliar a abrangência do Tagarela através da construção de um módulo de musicoterapia. Este módulo, em forma de aplicativo para dispositivos móveis, deve auxiliar o musicoterapeuta a aplicar atividades de composição musical. Estas atividades deverão proporcionar ao paciente a possibilidade de manipulação de elementos musicais sem a necessidade de conhecimento teórico musical, aumentando o interesse e a motivação do paciente.

(confirmar o formato dos objetivos aqui)

(Prof Dalton, alterei os objetivos específicos pois o prof Maurício comentou que os da proposta eram requisitos e não objetivos. O que acha?)

Este trabalho tem como objetivo criar para o Tagarela um módulo para facilitar a execução de atividades de composição musical em musicoterapia.

Os objetivos específicos são:

- a) disponibilizar aos musicoterapeutas uma ferramenta que os auxilie em atividades terapêuticas

- personalizadas de composição musical;
- b) permitir que usuários realizem atividades de composição musical mesmo que não possuam conhecimentos de teoria musical.

## 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentará os aspectos da fundamentação teoria. Na primeira seção deste capítulo serão apresentados os conceitos e o framework de desenvolvimento utilizado para desenvolver o aplicativo. A segunda seção apresentará o que já foi desenvolvido até o momento no projeto Tagarela. Por fim, a terceira seção apresentará três trabalhos correlatos com o trabalho relatado neste artigo.

### 2.1 CONCEITOS, TÉCNICAS E/OU FERRAMENTAS

Esta seção descreve os assuntos que fundamentarão o estudo a ser realizado: musicoterapia, teoria musical, protocolo MIDI e *framework* Ionic. Cada um destes assuntos possui uma seção nesta seção.

#### 2.1.1 Musicoterapia

Barcellos (2004, p. 3) afirma que “[...] a educação musical pode contribuir para o desenvolvimento de cérebros normais, enquanto a musicoterapia é fundamental para minimizar os danos que afetam o cérebro, como resultado de tantas patologias”. A musicoterapia é definida por Bruscia (2016, p. 55) como “[...] um processo reflexivo onde o terapeuta ajuda o cliente a otimizar sua saúde, usando variadas facetas da experiência musical e as relações formadas através desta como o ímpeto para a transformação”. Durante o processo de musicoterapia, segundo Bruscia (2016, p. 125), as atividades que podem ser executadas estão relacionadas com ações de improviso, execução, composição e audição musical. Bruscia (2016, p. 125) também afirma que cada uma destas atividades possui características próprias e únicas, e que, portanto, cada uma delas possui também seus próprios potenciais e aplicações.

Sobre as atividades de composição musical, Bruscia (2016, p. 130) afirma que “o terapeuta ajuda o cliente a escrever canções, letras ou peças instrumentais, ou a criar qualquer tipo de produto musical, tais como clipes de música ou fitas de áudio”. Algumas metas deste tipo de atividade são: desenvolver habilidades para expressar e organizar pensamentos e sentimentos, desenvolver habilidades de tomada de decisão e desenvolver habilidades de documentação e comunicação de sua composição para permitir que outros a reproduzam (BRUSCIA, 2016, p. 130). Barcellos (2004, p. 13) afirma que atividades de composição musical ajudam os pacientes a se expressarem. Em um relato de Barcellos (2004, p. 13), a autora expõe que este tipo de atividade fez com que pacientes passassem a expressar seus medos, decepções e esperanças com o passar do tempo, situação que não ocorria de início. Portanto, Barcellos (2004, p. 13) expõe que a atividade de composição musical permitiu que os pacientes desenvolvessem subjetividade.

Segundo Bruscia (2016, p. 68), a musicoterapia é centrada na criatividade e a participação do paciente, ou cliente, em musicoterapia exige dele criatividade. Conforme destacado por Ilari e Araújo (2010, p. 193) “o papel do musicoterapeuta é facilitar o engajamento do cliente com seu ser criativo, e processar os conteúdos expressos pelo cliente”.

#### 2.1.2 Teoria musical

(Posso manter a divisão dessa subseção em outras subseções? Pela aula de erros comuns está errado, mas parece que dessa forma fica organizado)













Med (1996, p. 11) define que as principais características do som são a duração, a intensidade, a altura e o timbre. As mesmas características também são apontadas na obra de Lacerda (1967, p. 1). Nesta seção, para cada um destes elementos será apresentada uma subseção em que a propriedade sonora será explicada e associada à elementos de notação musical.

##### 2.1.2.1 Duração do som

Segundo Lacerda (1967, p. 1), a propriedade sonora de duração é determinada pelo tempo de duração do som. Med (1996, p. 12) caracteriza a mesma propriedade sonora como o tempo em que ocorre a vibração que gera o som. Ambos autores associam a propriedade de duração do som com os elementos de notação musical figura de nota, fórmula de compasso e andamento.

A figura de nota, ou figura de som, é, segundo Priolli (2006, p. 13), a forma gráfica da nota. Priolli (2006, p. 13) afirma que “para representar as várias durações dos sons musicais as notas são escritas sob formas diferentes”. Cada figura de nota possui uma figura de pausa respectiva, que representa a duração do silêncio e que possui a mesma duração da figura de nota correspondente (PRIOLLI, 2006, p. 13). A Figura 1 demonstra o nome e as formas das figuras de notas e das suas respectivas figuras de pausas.

Figura 1 – Figuras de notas e figuras de pausas

nome	semibreve	mínima	semínima	colcheia	semicolcheia	fusa	semifusa
figura							
pausa							
ou							

Fonte: Med (1996).

De acordo com Med (1996, p. 27) existe uma relação de proporção entre as figuras de notas que pode ser binária ou ternária. Quando a divisão é binária, uma semibreve tem a duração de duas mínimas, que tem a duração de duas semínimas e assim por diante, seguindo a sequência apresentada na Figura 1 (MED, 1996, p. 27). Quando é ternária, uma semibreve tem a duração de três mínimas, que tem a duração de três semínimas e assim por diante, seguindo a sequência apresentada na Figura 1 (MED, 1996, p. 29).

Outro elemento que possui relação com a duração do som é a fórmula de compasso. Segundo Lacerda (1967 p.20), a fórmula de compasso é representada por dois números. Um indicando quantos tempos existem em cada compasso e outro indicando a unidade de tempo (LACERDA, 1967, p.20). Os dois números são apresentados um em cima do outro no início da partitura de uma música. De acordo com Lacerda (1967, p.20), número de baixo representa uma figura de nota, que representará a unidade de tempo da música. Caso o número seja 1, a unidade de tempo é a semibreve. Caso seja 2, a mínima. Caso seja 4, a semínima (LACERDA, 1967, p.20). E assim por diante, seguindo a sequência de figuras de notas apresentada na da Figura 1.

O andamento, conforme definido por Lacerda (1967, p. 29), é a velocidade da música. Lacerda (1967, p. 30) expõe que o andamento pode ser determinado com precisão se utilizando de valores em batidas por minuto (bpm). Med (1996, p. 194) exemplifica que existe sempre uma figura de nota com valor equivalente a 1 bpm. Uma vez determinada a figura de nota, a duração das demais é obtida pela relação de proporção entre as figuras de nota. De acordo com Lacerda (1967, p. 30), o andamento pode ser representado através da indicação metronômica. Esta indicação é grafada na partitura por uma figura de nota, seguida de um sinal de igual, seguido da quantidade da figura de nota por minuto (LACERDA, 1967, p. 30).

#### 2.1.2.2 Intensidade do som

Segundo Lacerda (1967, p. 1), a propriedade sonora de intensidade é determinada pela capacidade de o som ser mais forte ou mais fraco. Med (1996, p. 12) caracteriza a mesma propriedade sonora como a amplitude da vibração que gera o som, e complementa relacionando a propriedade ao volume sonoro. Ambos autores associam a propriedade de intensidade do som com o elemento de notação musical sinais de dinâmica.

Lacerda (1967, p. 49) define dinâmica como “[...] a arte de graduar a intensidade sonora na execução musical”. Lacerda (1967, p. 49) também apresenta sinais de dinâmica utilizados para indicar qual a intensidade sonora que deve ser aplicada no trecho musical que conter o sinal. Em ordem crescente de intensidade sonora os sinais de dinâmica apresentados são: ppp, pp, p, mp, mf, f, ff e fff (LACERDA, 1967, p. 49). O p nesta escala significa piano, que é sinônimo de intensidade sonora fraca e o f significa forte, e é sinônimo para intensidade sonora forte (LACERDA, 1967, p. 49).

#### 2.1.2.3 Altura do som

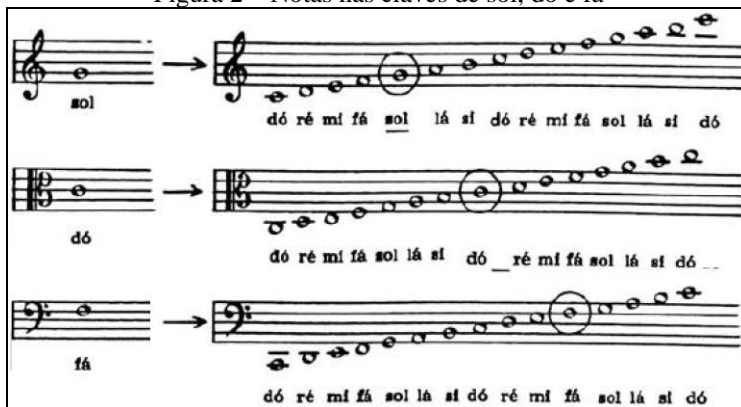
Segundo Lacerda (1967, p. 1), a propriedade sonora de altura é determinada pela capacidade de o som ser mais grave ou mais agudo. Med (1996, p. 11-12) caracteriza a mesma propriedade sonora como a velocidade da vibração que gera o som, e complementa afirmando que quanto maior a velocidade de vibração mais aguda será a nota. Ambos autores associam a propriedade de duração do som com os elementos de notação musical clave e posição da nota no pentagrama.

Priolli (2006, p. 9) define clave como um símbolo utilizado “para determinar o nome da nota e sua altura na escala”. Segundo Med (1996, p. 16), a clave é colocada no início do pentagrama e dá o seu nome à nota que estiver na mesma linha. Segundo Lacerda (1967, p. 5), existem três tipos de clave: clave de sol, clave de fá e clave de dó. A posição da nota no pentagrama determina a altura das notas em relação à nota que recebeu o nome da clave, conforme abordado por Lacerda (1967, p. 5). A Figura 2 apresenta, a esquerda das flechas, os três tipos de

clave grafados em um pentagrama e, a direita das flechas, uma sequência de notas nomeadas, demonstrando como o processo de dedução da nota musical por localização no pentagrama é realizado em cada tipo de clave.

Um meio de alteração de altura de som, de acordo com Lacerda (1967, p. 119), é aplicar uma transposição na música. A transposição altera a altura da música mantendo a diferença de altura entre as notas a mesma (LACERDA, 1967, p. 119). De acordo com Lacerda (1967, p. 80), a tonalidade de uma música tem relação com a escala de notas que é utilizada como base para a construção da música e recebe o mesmo nome da escala de notas, por exemplo: Dó Maior. De acordo com Med ( ) as escalas podem ser identificadas pelos símbolos de sustenido (#) ou bemol (b) apresentados no início da partitura. Conforme exemplificado por Lacerda (1967, p. 119), a transposição pode ocorrer alterando a tonalidade da música.

Figura 2 – Notas nas claves de sol, dó e fá



Fonte: Lacerda (1967).

#### 2.1.2.4 Timbre do som

Segundo Lacerda (1967, p. 1), a propriedade sonora de timbre é determinada pela qualidade do som. Lacerda (1967, p. 1) ainda afirma que esta propriedade é responsável por permitir reconhecer a origem do som. Med (1996, p. 12) caracteriza a mesma propriedade sonora como a “combinação de vibrações determinadas pela espécie do agente que as produz”. Ambos autores associam a propriedade de timbre do som com o elemento de notação musical indicação da voz ou instrumento que deve executar a música.

#### 2.1.3 Protocolo midi

(Revisar as referências. Parece que está apontando somente para o site.)

O protocolo *Musical Instrument Digital Interface* (MIDI), segundo Hass (2013) é um dos principais catalizadores do avanço recente da tecnologia musical. Segundo MIDI Association (2017), o protocolo MIDI oferece a possibilidade de conectar diversos dispositivos de diferentes fabricantes, como instrumentos digitais musicais, computadores e dispositivos moveis. O protocolo em questão permite que pessoas de diversas profissões, como por exemplo músicos, produtores e educadores, possam criar, executar, aprender e compartilhar música (MIDI ASSOCIATION, 2017).

Segundo MIDI Association (2017), o protocolo MIDI proporciona, de forma eficiente, um meio de converter dados de performance musical para dados eletrônicos. Estes dados convertidos podem então ser enviados, por meio de mensagens MIDI, como instrução para um sintetizador musical executar partes musicais (MIDI ASSOCIATION, 2017). Para permitir armazenar e transportar entre dispositivos sequencias de mensagens MIDI, em 1988, foi criado o padrão de arquivo MIDI (HASS, 2013). Isso possibilita, por exemplo, que um compositor salve sua composição musical através de um programa de notação musical em um arquivo MIDI e abra o arquivo em outro software, como um executor de mídia, de forma que as propriedades e características de sua criação sejam recuperadas através do arquivo MIDI (HASS, 2013).

Os arquivos MIDI são compostos por *chunks*, que são compostos por um cabeçalho seguido dos dados do *chunk* (MIDI ASSOCIATION, 2017). O cabeçalho dos *chunks* é composto por 4 caracteres, que identificam o tipo do *chunk*, seguido de 32 bits, que representam o tamanho dos dados do *chunk* (MIDI ASSOCIATION, 2017). Segundo MIDI Association (2017), o protocolo MIDI possui dois tipos de *chunks*, o *header chunk* e o *track chunk*. O *header chunk*, identificado no arquivo por *MThd*, possui dados gerais sobre a música, já o *track chunk* identificado no arquivo por *MTrk*, possui dados específicos de um canal MIDI, onde são registradas as mensagens MIDI, que representam os detalhes da performance musical (MIDI ASSOCIATION, 2017). Segundo MIDI Association (2017), é possível criar até 16 *track chunks*, que serão executados simultaneamente. Alguns exemplos de elementos que são

representados dentro dos canais MIDI são timbre, duração, altura e intensidade de sons (MIDI ASSOCIATION, 2017). A Figura 3 demonstra a macroestrutura de um arquivo MIDI.

Figura 3 – Estrutura do arquivo MIDI

```
MThd <length of header data>
<header data>
MTrk <length of track data>
<track data>
MTrk <length of track data>
<track data>
...
```

Fonte: MIDI Association (2017).

#### 2.1.4 Framework Ionic

(Descrever o framework? No texto de instrução está escrito que ferramentas de implementação já conhecidas não devem ser abordadas. Ionic se enquadra neste caso? Caso seja parte, acredito que posso melhorar o texto para abordar algumas funcionalidades do Angular e explicar melhor os plugins.)

O *framework* Ionic possibilita que desenvolvedores que sabem desenvolver um *website* possam utilizar este conhecimento para desenvolver aplicativos para as plataformas *mobile* (IONIC FRAMEWORK, 2013). De acordo com Ionic Framework (2013) a companhia Ionic foi fundada em 2012 e, até os dias atuais, seu *framework* já foi utilizado por milhões de desenvolvedores para construir milhões de aplicativos para plataformas *mobile*.

Para utilizar o *framework* em questão é necessário realizar primeiro a instalação do Node.js para que então seja possível instalar o Ionic CLI e o Cordova através de linhas de comando (IONIC FRAMEWORK, 2013). Segundo Ionic Framework (2013), o Ionic CLI permite que os aplicativos Ionic sejam criados e gerenciados via linhas de comando. Já o Cordova é o componente utilizado para realizar *build* e *deploy* do aplicativo para plataformas *mobile* (IONIC FRAMEWORK, 2013).

Quando um aplicativo é criado no Ionic, os componentes necessários para o projeto são instalados automaticamente, bem como é configurado o Cordova para o aplicativo (IONIC FRAMEWORK, 2013). Quando o processo de criação do aplicativo se encerra, a estrutura do projeto está criada (IONIC FRAMEWORK, 2013). Nesta estrutura existem os arquivos de código fonte do aplicativo, como arquivos HyperText Markup Language (HTML) e TypeScript (TS) (IONIC FRAMEWORK, 2013).

O Ionic oferece uma vasta gama de componentes que podem ser importados nos arquivos de código fonte, como botões, listas, menus e componentes de mensagem (IONIC FRAMEWORK, 2013). O *framework* também fornece serviços que podem ser importados nas classes do aplicativo para gerenciar o funcionamento de componentes ou do aplicativo (IONIC FRAMEWORK, 2013). Além disto, utilizando o Ionic Native, é possível acessar recursos nativos dos dispositivos *mobile*, como a câmera, a geolocalização ou a vibração do dispositivo (IONIC FRAMEWORK, 2013).

Para executar o aplicativo em plataforma *web*, se faz necessário executar um comando via linha de comando para que o Ionic publique um servidor *web*, possibilitando que o aplicativo seja executado pelo *browser* (IONIC FRAMEWORK, 2013). Para tanto, segundo Ionic Framework (2013), o Ionic converte o código escrito em TypeScript para uma versão correspondente em JavaScript, uma vez que, de forma geral, os *browsers* não executam TypeScript. Para realizar o *build* ou o *deploy* do aplicativo para as plataformas *mobile* se faz necessário atender alguns requisitos de acordo com a plataforma *mobile* que será utilizada (IONIC FRAMEWORK, 2013). Porém, após atendidos os requisitos, basta executar comandos em linha de comando para que o Cordova realize a tarefa em questão, permitindo executar o aplicativo em um dispositivo *mobile* (IONIC FRAMEWORK, 2013).

## 2.2 VERSÃO ANTERIOR DA PLATAFORMA

O Tagarela começou a ser desenvolvido como proposta do trabalho de conclusão de curso em 2012. Deste trabalho até os dias atuais diversos outros trabalhos foram desenvolvidos para realizar migrações de tecnologia, melhorias de funcionalidade e adição de funcionalidades à plataforma.

No trabalho de Fabeni (2012), o Tagarela foi construído como um aplicativo de comunicação alternativa para a plataforma iOS 6. Os objetivos do Tagarela, segundo o autor, eram construir um ambiente para auxiliar na comunicação de pessoas com necessidades especiais e crianças em fase de alfabetização, utilizar imagens e áudios para propiciar imersão ao usuário e possibilitar customizações para atender necessidades específicas dos usuários. O trabalho de conclusão de curso de Marco (2014) realizou a conversão da versão do Tagarela de Fabeni (2012) para a plataforma Android. Já o trabalho de Wippel (2015) realizou a conversão da versão inicial do Tagarela utilizando o *framework* PhoneGap. Uma funcionalidade destes trabalhos é a possibilidade de o usuário pode interagir com um símbolo que pode emitir um som associado. Com esta característica o usuário pode se comunicar por meio dos símbolos existentes na prancha, que foram previamente cadastrados.



Após a versão inicial do Tagarela, Reetz (2013) adicionou a ele um jogo de letras e números para a plataforma Android. Marques (2014) converteu o trabalho de Reetz (2013) para iOS. Esta mesma funcionalidade foi convertida por Ferreira (2016) para os *frameworks* PhoneGap e Ionic. Estes trabalhos resultaram em um aplicativo em que o usuário pode aprender a como desenhar uma letra do alfabeto.

Cazagrande (2016) adicionou um módulo para o ensino do Braille utilizando o *framework* PhoneGap. Neste trabalho foram desenvolvidas atividades relacionadas com apresentação, consulta e prática dos sinais da escrita Braille. Em uma das funcionalidades deste trabalho, o software exibe uma figura e parte do nome do objeto da figura em Braille, o usuário então precisa completar os sinais faltantes na representação do nome do objeto em Braille. Sautner (2017) adicionou ao Tagarela, utilizando o *framework* Ionic, um módulo para auxiliar crianças autistas na aquisição de linguagem. Neste módulo, é exibida a imagem de um objeto ao usuário e o usuário deve pronunciar a palavra que representa o objeto exibido, podendo consultar o som da pronúncia do objeto ao aplicativo.

Considerando todos os trabalhos citados, hoje o Tagarela possui quatro segmentos. Além de ser uma Plataforma de Comunicação Alternativa (PCA), ele possui um módulo de jogo voltado para o aprendizado de símbolos gráficos, um módulo de jogo dedicado ao ensino da linguagem Braille e um módulo voltado para auxiliar crianças autistas na aquisição de linguagem.

### 2.3 TRABALHOS CORRELATOS

(Tenho dúvidas em relação a formatação dos quadros. Preciso escrever melhor os parágrafos de explicação de cada correlato.)

Nesta seção serão descritos três trabalhos correlatos que abordam assuntos relacionados à musicoterapia e/ou utilização de tecnologia para manipulação de elementos musicais. O primeiro trabalho, desenvolvido por Monteiro (2016), relata uma experiência de estágio em que foi aplicada a musicoterapia em um contexto escolar. O Quadro 1 apresenta as características deste trabalho. O segundo trabalho, desenvolvido por Corrêa et al. (2013), aborda a construção e aplicação de um software em atividades de musicoterapia. O Quadro 2 apresenta as características deste trabalho. O último trabalho, desenvolvido por Lima (2006), trata de um software construído para manipular, de forma a proporcionar mais conforto aos cantores, elementos de arquivos de música utilizados para gerar acompanhamento durante performances musicais. Este trabalho tem suas características apresentadas no Quadro 3.

Quadro 1 – Trabalho Correlato 1

Referência	A Musicoterapia Em Contexto Escolar: Perturbações Do Comportamento, Espectro Do Autismo E Multideficiência. Monteiro (2016).
Objetivos	Realizar sessões de musicoterapia para alunos de duas instituições de ensino de Portugal. As sessões deveriam promover concentração, autocontrole, disciplina, comunicação e bem-estar físico e emocional para os alunos.
Principais funcionalidades	O planejamento das sessões de musicoterapia se baseou em seis atividades diferentes, cada uma com objetivos específicos próprios. São as atividades: <ul style="list-style-type: none"> <li>a) canção do bom dia: utilizada para contextualizar os participantes do início da sessão de musicoterapia;</li> <li>b) canção: utilizada para abordar a comunicação verbal e a métrica musical, uma vez que o participante deveria cantar ao mesmo tempo que o musicoterapeuta;</li> <li>c) improvisação: utilizada na construção da relação terapêutica e para estimular reações espontâneas do participante através de comunicação, criação e improvisação musical;</li> <li>d) repetição/criação: utilizada para melhorar a capacidade de concentração e memória do participante;</li> <li>e) escuta musical: utilizada para relaxar o participante;</li> <li>f) canção da despedida: utilizada para contextualizar o participante sobre o fim da sessão de musicoterapia.</li> </ul>
Ferramentas de desenvolvimento	Como o trabalho não trata do desenvolvimento de alguma ferramenta tecnológica, não se aplica.
Resultados e conclusões	Segundo Monteiro (2016, p. 64), após a avaliação dos resultados, foi possível concluir que o estágio desenvolvido com as sessões de musicoterapia melhorou a qualidade de vida em âmbito escolar dos alunos que participaram das sessões. Monteiro (2016, p. 67) destaca que em algumas sessões “[...]aparentemente não acontecia nada, por vezes parecia até que a criança não dava conta de que eu estava na sala com ela, no entanto quando existiram progressos com essas crianças tudo fez sentido [...]”. Sendo assim, o trabalho desenvolvido com as sessões de musicoterapia se mostrou bem-sucedido, até para os casos que impuseram mais dificuldades.

Fonte: elaborado pelo autor.

O trabalho de Monteiro (2016) trata de questões terapêuticas envolvendo a utilização de música durante estes processos. Algumas das atividades abordadas no trabalho em questão trataram de processos que envolvem criação

musical. Apesar de o trabalho não tratar da construção de uma ferramenta tecnológica, o trabalho proposto possui associação com o trabalho de Monteiro (2016) por abordar questões terapêuticas envolvendo música. O trabalho proposto, diferente do trabalho de Monteiro (2016), aborda somente uma atividade de composição musical.

Quadro 2 – Trabalho Correlato 2

Referência	GenVirtual. Corrêa et al. (2013).
Objetivos	Utilizar realidade aumentada para apoiar a realização de atividades envolvendo música por crianças com deficiência motora e cognitiva.
Principais funcionalidades	O software em questão possui um módulo de composição livre, um módulo de visualização de partituras e um módulo de jogo de memória.
Ferramentas de desenvolvimento	Foram utilizadas as bibliotecas: a) ARToolkit: responsável pelos processos de realidade aumentada; b) Microsoft Foundation Class Library (MFC): responsável pela interface gráfica com o usuário; c) OpenAL (Open Audio Library): responsável por processos de manipulação de arquivos de áudio. d) API Win32: responsável pela manipulação de mensagens MIDI; e) OpenGL: responsável por processos de manipulação de rotinas gráficas.
Resultados e conclusões	O GenVirtual foi avaliado por especialistas em música e por crianças com déficits motores e cognitivos. Com base nestas avaliações, Corrêa et al. (2013, p. 129) concluíram que o GenVirtual é capaz de apoiar atividades de improvisação, recriação e composição musical e audição sonora e musical. Segundo os autores, os pacientes tiveram seu desempenho e estímulos melhorados com a utilização do GenVirtual, o que acarretou em um aumento de motivação dos pacientes para com a terapia. Segundo os pacientes a terapia foi mais divertida com a utilização do software.

Fonte: elaborado pelo autor.

O GenVirtual de Corrêa et al. (2013) também trata de questões terapêuticas envolvendo música. Assim como o trabalho proposto, o GenVirtual trata do desenvolvimento de uma ferramenta tecnológica para auxiliar em atividades terapêuticas. Ambos os trabalhos possuem como funcionalidade a execução de arquivos MIDI, porém o GenVirtual não realiza edições nesses arquivos, enquanto o trabalho proposto sim. Outro ponto que difere o trabalho correlato do trabalho proposto é a plataforma para qual ele foi desenvolvido. O GenVirtual é um software que é executado em desktop. Já o trabalho proposto será executado em dispositivos móveis

Quadro 3 – Trabalho Correlato 3

Referência	Best Vocal 2005. Lima (2006).
Objetivos	Adequar a tonalidade da música de uma fonte sonora aos limites da extensão vocal do cantor.
Principais funcionalidades	O software é capaz de abrir, reproduzir, salvar e editar características musicais de arquivos MIDI que continham diversos canais simultâneos de áudio.
Ferramentas de desenvolvimento	Linguagem de programação CLEAN.
Resultados e conclusões	Lima (2006, p. 142), expõe em suas conclusões que o trabalho obteve sucesso em realizar as análises e alterações propostas. Um fator que o autor considera em sua análise é a utilização da ferramenta de forma profissional por músicos durante um período de tempo superior a um ano, o que comprova de forma prática o atendimento dos requisitos.

Fonte: elaborado pelo autor.

Embora o Best Vocal 2005 não apresente uma solução focada em atividades terapêuticas, ele realiza alterações em atributos de arquivos MIDI. Esta característica torna o trabalho de Lima (2006) um trabalho correlato do trabalho proposto. O Best Vocal 2005 tem como principal objetivo editar a propriedade sonora de altura de arquivos MIDI. Esta característica o difere do trabalho proposto. Isto pois o trabalho proposto realizará alterações nas propriedades sonoras de altura, duração, intensidade e timbre para possibilitar a atividade de composição musical. O Best Vocal 2005 é executado em desktop. Assim como no caso do GenVirtual, esta característica difere o trabalho proposto do Best Vocal 2005.

### 3 DESCRIÇÃO DO APLICATIVO

Este capítulo pretende apresentar os detalhes de especificação e implementação do aplicativo. O desenvolvimento do aplicativo foi realizado com o framework Ionic. São apresentadas quatro seções. A primeira seção apresenta a visão geral do aplicativo, de forma a ambientar o leitor quanto ao funcionamento do processo de composição e as ações de alterações de propriedades musicais. A segunda seção apresenta os detalhes da carga de dados do aplicativo, detalhando a estrutura de composição e o processo de leitura de arquivos MIDI. A terceira apresenta as

características do processo de configuração, por meio de parâmetros, da composição que será realizada no aplicativo. A última seção apresenta os componentes da tela de composição do aplicativo abordando a funcionalidade de cada componente. Devido à dependência de recursos locais, o aplicativo funciona somente em dispositivos móveis. Durante as seções serão referenciados os plugins do Ionic utilizados no desenvolvimento. Também serão citados os elementos, como classes, que compõem o aplicativo. O diagrama de classes do aplicativo pode ser encontrado no APÊNDICE A.

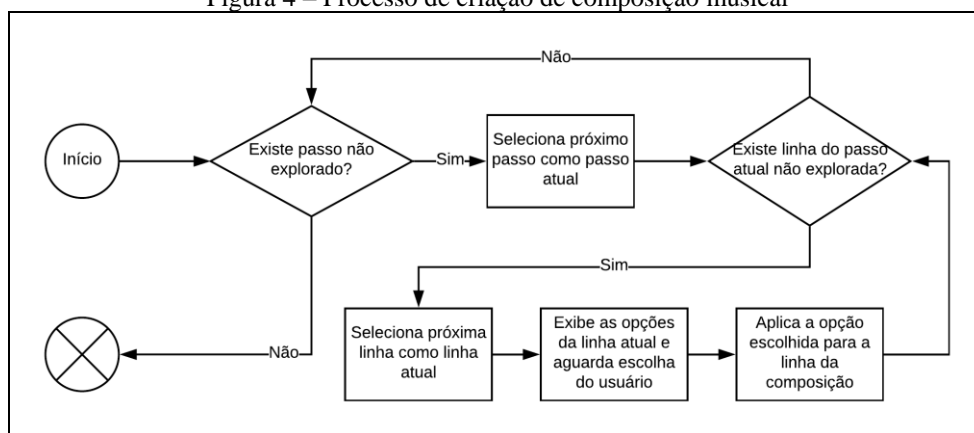
### 3.1 VISÃO GERAL DO APLICATIVO

O aplicativo disponibiliza para o usuário uma forma de combinar fragmentos musicais para gerar uma composição musical personalizada. O processo de criação da composição musical personalizada é organizado em etapas. Em cada etapa o usuário deve escolher um fragmento musical dentre os possíveis fragmentos exibidos pelo aplicativo. Quando um fragmento é escolhido ele é adicionado à composição, os fragmentos não escolhidos são descartados e uma nova etapa de escolha se inicia. Para possibilitar esta dinâmica, os fragmentos musicais são agrupados em dois níveis. O primeiro nível existe para que seja possível segregar os fragmentos de acordo com o posicionamento que eles podem assumir dentro da composição. Já o segundo nível pretende segregar os fragmentos em razão da sua função musical. Para estruturar os dados dos fragmentos musicais e organizar a dinâmica de composição musical o aplicativo utiliza quatro elementos, são eles:

- c) opção: é o elemento que representa um fragmento musical;
- d) linha: é o elemento que agrupa as opções;
- e) passo: é o elemento que agrupa linhas;
- f) fonte de composição: é o elemento que agrupa passos.

O aplicativo utiliza uma fonte de composição para configurar o processo de criação de uma nova composição musical personalizada. Quando o processo de criação da composição é iniciado, com base nos passos existentes na fonte de composição, o aplicativo divide a composição em macro etapas, gerando uma macro etapa de composição por passo. Dentro de cada passo, para cada linha existente no passo, o aplicativo gera uma etapa de composição, que contém as opções disponíveis ao usuário para aquela combinação de agrupamento. Com base nesses dados, o aplicativo exibe as opções, de etapa em etapa, para que o usuário realize a escolha de um fragmento musical por etapa. O processo de composição se inicia com a exibição das opções da primeira linha do primeiro passo. Assim que o usuário realiza a escolha de uma opção o aplicativo adiciona sua escolha à composição e passa a exibir as opções da segunda linha do primeiro passo, se existir mais de uma linha. Quando todas as linhas do primeiro passo forem exploradas, o aplicativo passa a exibir os dados do segundo passo, se existir mais de um passo. Esse processo se repete até que todos os passos tiverem sido explorados. Quando todos os passos tiverem sido explorados a composição musical está completa. Todos os passos de uma composição musical devem possuir as mesmas linhas em seu agrupamento. A Figura 4 exibe o fluxograma que representa o processo de criação de composição musical.

Figura 4 – Processo de criação de composição musical



Fonte: elaborado pelo autor.

O agrupamento dos fragmentos musicais em grupos de passos e linhas é realizado da forma como foi apresentada para possibilitar restrições em relação à posicionamento e função musical de cada fragmento musical. O agrupamento de linha existe para que, a cada passo de composição, o usuário tenha que escolher um elemento que tem uma função musical para a composição, como, por exemplo, melodia ou ritmo. Já o agrupamento de passo tem relação com a segregação de posicionamento. Para que a composição criada com o aplicativo funcione de acordo com as regras de teoria musical e harmonia, todos os elementos dentro de um passo devem ser compatíveis uns com os outros, independentemente da linha em que os fragmentos musicais se encontram. Devido a estes dois agrupamentos, caso os dados da fonte de composição estejam consistentes entre si, a composição resultado será consistente.



Além de possibilitar a escolha de fragmentos musicais para a geração de uma composição musical personalizada, o aplicativo também permite a edição de propriedades sonoras dos fragmentos musicais. Quatro são as propriedades que podem ser alteradas e essas alterações podem surtir efeito a nível de composição, de linha ou de opção. Quando uma propriedade sonora de opção é alterada, o novo valor escolhido somente altera a opção editada. Quando uma propriedade sonora de linha é alterada, todas as opções que já foram escolhidas para a linha terão a propriedade alterada editada. Quando uma propriedade sonora da composição for alterada a nova propriedade é aplicada em todos os fragmentos da fonte de composição. O Quadro 4 demonstra a relação das alterações de propriedades sonoras com os elementos de composição e relaciona os elementos que o aplicativo permite alterar com as propriedades sonoras.

Quadro 4 – Relação entre propriedades sonoras editáveis e propriedades da composição

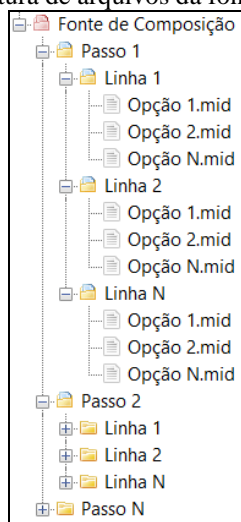
Propriedade sonora	Elemento de composição alterado	Funcionamento
Timbre	opção	O timbre é alterado quando a propriedade de instrumento musical da opção é alterada.
Intensidade	linha	A intensidade é alterada quando a propriedade de volume da linha é alterada.
Duração	fonte de composição	A duração é alterada quando a propriedade de velocidade da composição é alterada.
Altura	fonte de composição	A altura é alterada quando a propriedade de tonalidade da composição é alterada.

Fonte: elaborado pelo autor.

### 3.2 CARGA DE DADOS DE COMPOSIÇÃO

Para que os fragmentos musicais sejam disponibilizados para o aplicativo é utilizado o sistema de arquivos do dispositivo móvel. Com base em pastas e arquivos, a estrutura dos elementos de composição é representada. Uma pasta representa a fonte de composição. Dentro desta pasta, outras pastas representam os passos. Cada pasta dentro da pasta de fonte de composição representa um passo. Dentro de cada pasta de passo, outras pastas representam as linhas. Cada pasta dentro de uma pasta de passo representa uma linha. Dentro de cada pasta de linha, arquivos MIDI representam as opções. Cada arquivo pode conter dados de uma ou várias opções. A Figura 5 demonstra a organização da estrutura em questão.

Figura 5 – Estrutura de arquivos da fonte de composição

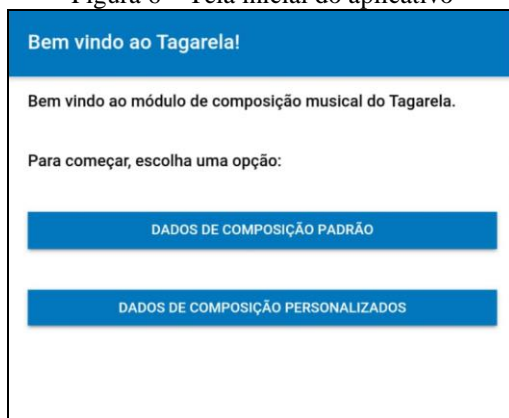


Fonte: elaborado pelo autor.

Por padrão, o aplicativo contém quatro fontes de composição. Estas fontes de composição são disponibilizadas com a instalação do aplicativo. Porém, para possibilitar que o usuário crie fontes de composição personalizadas, é possível criar esta estrutura em um diretório externo à aplicação. O caminho relativo do diretório monitorado pelo aplicativo, Tagarela/Musicoterapia/Composicoes, é o mesmo para dispositivos Android ou iOS. Porém, para dispositivos Android o diretório deve ser criado na raiz do armazenamento interno do dispositivo e para iOS no diretório de documentos. Estas pastas devem ser criadas manualmente pelo usuário se for necessário utilizar uma fonte de composição personalizada. Isto pois, em alguns dispositivos, as pastas criadas pelo aplicativo não ficam visíveis ao usuário. Todas as pastas existentes na pasta Composicoes são consideradas fontes de composição pelo aplicativo.

Para que o aplicativo consiga fazer distinção entre os dados padrão e os dados personalizados, a tela inicial da aplicação dispõe de dois botões. O primeiro botão realiza a leitura de dados de fontes de composição padrão. Já o segundo realiza a leitura de dados do diretório de fontes de composição personalizadas. A Figura 6 demonstra a tela inicial do aplicativo. Quando qualquer um dos botões for pressionado, uma lista contendo o nome dos diretórios de fonte de composição é exibida. Se nenhuma pasta de fonte de composição for encontrada, uma mensagem é exibida ao usuário informando sobre a ausência de fontes de composição. Quando uma das opções listadas é selecionada, o processo de carga de dados de composição é iniciado.

Figura 6 – Tela inicial do aplicativo



Fonte: elaborado pelo autor.

O processo de carga de dados consiste em carregar os dados da fonte de composição para a memória do aplicativo, bem como definir os valores de configuração padrão para a composição. Também é nesta etapa que os dados de configuração anteriormente salvos são recarregados, caso existam. No processo de carga de dados as classes `MusicalCompositionConfigControl` e `MusicalCompositionSourceControl` são instanciadas. Estas classes são responsáveis por instanciar e carregar os dados para as classes `MusicalCompositionConfig` e `MusicalCompositionSource` respectivamente.

O primeiro método executado pelo processo de carga de dados é o `loadConfigs` da classe `MusicalCompositionConfigControl`. Este método é responsável por instanciar a classe de configuração `MusicalCompositionConfig` e definir os valores dos seus atributos. Esta classe de configuração contém dados sobre a localização dos elementos de composição no sistema de arquivos do dispositivo. Ou seja, os caminhos relativos de cada passo, linha e opção existentes na fonte de composição escolhida são atributos desta classe. Além disto, esta classe também armazena dados de parâmetros da composição. A próxima seção abordará as questões que dizem respeito à parametrização do processo de composição.

Após a criação da classe modelo de configuração, a classe modelo das fontes de composição é instanciada. Este processo ocorre com a execução do método `loadSources` da classe `MusicalCompositionSourceControl` que recebe como parâmetro a instância da classe `MusicalCompositionConfig`, criada na execução da função anterior. Este método lê os arquivos MIDI de acordo com os caminhos determinados na classe de configuração e carrega para a memória do dispositivo os dados MIDI que são pertinentes à composição. Para isto, para cada opção mapeada no objeto de configuração, a função `setupMidiFromFile` da classe `MidiFileControl` é executada. A classe `MidiFileControl` é atributo da classe `MusicalCompositionSourceControl`.

A função `setupMidiFromFile` recebe uma string binária como parâmetro e retorna um array de objetos `Midi`. Cada objeto do array retornado representa uma opção para o grupo linha/ passo que está sendo processado. A string binária é obtida através da leitura do arquivo MIDI pelo plugin `File` do Ionic. A função `setupMidiFromFile` realiza a leitura, caractere a caractere, da string binária determinando quais dados são importados para um objeto `Midi` e se a configuração do arquivo está de acordo com o esperado pelo aplicativo. Primeiramente a função em questão realiza as verificações dos dados existentes no cabeçalho do arquivo MIDI. Somente são carregados os arquivos MIDI tipo 0 e 1. Os arquivos MIDI tipo 2 geram uma exceção na função em questão, gerando uma mensagem específica para o usuário e interrompendo o processo de carga de dados. Outra validação realizada é o tipo de `division` do arquivo MIDI. Somente é suportado pelo aplicativo o tipo 0. Arquivos MIDI com tipos diferentes deste também geram exceção, mensagem ao usuário e interrompem o processo de carga de dados.

Após realizar a validação do cabeçalho MIDI, para cada `track` do arquivo MIDI, é instanciada uma classe `Midi` de tipo 0. O retorno da função `setupMidiFromFile` é um array de instâncias `Midi` contendo os objetos criados neste processamento. Para cada evento MIDI de cada uma das `tracks` existentes no arquivo MIDI, a função

calcula o `delta time` do evento e verifica se o evento MIDI deve ser importado. Caso o evento seja de um tipo que não é importado, o `delta time` é somado ao próximo evento que será importado. Isto para manter a proporção de tempo entre os eventos importados consistente. O Quadro 5 apresenta os eventos MIDI que são importados pelo aplicativo e o motivo pelo qual eles são considerados.

Quadro 5 – Eventos MIDI carregados pelo aplicativo

Evento MIDI	Função do evento para o aplicativo
Note On	Evento que representa o início da execução de uma nota musical. Este evento possui associação com propriedades sonoras de duração e altura.
Note Off	Evento que representa o fim da execução de uma nota musical. Este evento possui associação com propriedades sonoras de duração e altura.
Key Signature	Evento que representa a tonalidade. Este evento possui associação com a propriedade sonora de altura.
Time Signature	Evento que representa as propriedades de fórmula de compasso. Este evento possui associação com a propriedade sonora de duração.
End Of Track	Evento que representa o fim de uma <code>track</code> .

Fonte: elaborado pelo autor.

Para cada `track` processada a função em questão realiza a validação de alguns dados para garantir a consistência dos objetos `Midi` criados. Para este processamento o segundo parâmetro da função é considerado. Este parâmetro é um objeto `Midi` que foi importado anteriormente. Este parâmetro é utilizado para comparar alguns eventos e garantir a compatibilidade entre as opções da fonte de composição. O APÊNDICE C APÊNDICE C demonstra essas validações. Se qualquer uma das validações apontadas no apêndice falharem, o processamento é encerrado e uma mensagem de erro, específica para cada situação, é exibida ao usuário.

A terceira etapa do processo de carga de dados é a execução do método `loadSavedConfigs` da classe `MusicalCompositionConfigControl`. Este método é responsável por buscar o arquivo contendo os dados de parametrização da fonte de composição e carregar seus valores. O arquivo de configuração é um arquivo JSON nomeado `config.json` que possui como atributos os mesmos atributos da classe `MusicalCompositionConfig`. Este arquivo é salvo após a etapa de parametrização da fonte de composição. A localização do arquivo de configuração varia dependendo do tipo da fonte de composição. Se for uma fonte de composição padrão, o arquivo é encontrado no mesmo caminho relativo em que se encontra a composição. Porém, como não é possível gravar dados no diretório de aplicação do aplicativo, o diretório de dados do aplicativo é utilizado como diretório raiz. Se for uma fonte de composição personalizada, o arquivo se encontra na pasta que representa a fonte de composição. Se a função `loadSavedConfigs` encontrar o arquivo de configuração, é realizada uma verificação de coerência da estrutura do arquivo com a estrutura da fonte de composição que está sendo carregada. Caso a verificação falhe, o arquivo de configuração é apagado e uma mensagem é exibida ao usuário informando sobre a inconsistência. Neste caso o processo de carga de dados é interrompido. Caso o arquivo seja consistente com a estrutura da fonte de composição, todos os valores dos atributos do arquivo de configuração são aplicados aos atributos do objeto `MusicalCompositionConfig`.

A execução do método `determinateMidiChannelsAttributesValues` da classe `MusicalCompositionConfigControl` é a quarta etapa do processo de carga de dados. Este método é responsável por determinar os valores de instrumentos musicais disponíveis e o instrumento musical padrão para cada uma das opções disponíveis na fonte de composição. Sendo assim, o método altera atributos do objeto `MusicalCompositionConfig`. Caso a opção possua um objeto `Midi` que utiliza o canal reservado para os instrumentos de percussão, o método em questão atribui a lista de instrumentos de percussão para o atributo de instrumentos musicais disponíveis para a opção. Neste caso, o valor do atributo de instrumento padrão passa a ser o primeiro elemento da lista em questão. Caso contrário, os valores dos dois atributos são determinados pelos valores do arquivo de configuração. Caso o arquivo não tenha sido carregado, o valor utilizado é a lista de instrumentos melódicos para o atributo de instrumentos musicais disponíveis. O valor do atributo de instrumento musical padrão passa a ser o primeiro elemento da lista de instrumentos musicais disponíveis. As listas de instrumentos musicais de percussão e melódicos são constantes da classe `MusicalCompositionConfigControl`. Os elementos destas listas são expostos no APÊNDICE B APÊNDICE B.

A quinta etapa do processo de carga de dados é a execução do método `setTempoAndKeySignatureValues` da classe `MusicalCompositionConfigControl`. O método em questão é responsável somente por buscar os valores de fórmula de compasso e modo de tonalidade que serão utilizados na composição. Como todos os objetos `Midi` de uma fonte de composição devem possuir a mesma fórmula de compasso e o mesmo modo de tonalidade, os valores da primeira opção da fonte de composição são adotados para determinar os valores para os atributos `numerator`, `denominator` e `mode` do objeto `MusicalCompositionConfig`.

A sexta e última etapa do processo de carga de dados é a execução do método `normalizeTimeDivision` da classe `MusicalCompositionConfigControl`. Este método percorre todos os objetos `Midi` da fonte de composição para determinar o maior valor do atributo `timeDivision.metric` desses objetos. Após o método `ajustMidiTimeDivision` da classe `MidiControl` é executado para cada um dos objetos `Midi` existentes na fonte de composição utilizando como parâmetro o maior valor de `timeDivision.metric`. A classe `MidiControl` é um atributo da classe `MusicalCompositionConfigControl`. O método `ajustMidiTimeDivision` realiza a adequação do valor do atributo `timeDivision.metric` de um objeto `Midi`. Ele recebe como parâmetro o valor que deve ser aplicado ao atributo em questão, além do próprio objeto `Midi`. Com base no valor atual de `timeDivision.metric` do objeto `Midi` e o valor do parâmetro é gerado um valor de proporção que servirá como fator de multiplicação para os valores de `delta time` de todos os eventos do objeto `Midi`. Por exemplo, se o valor atual de `timeDivision.metric` do objeto `Midi` é 100 e o novo valor é 200 o fator de proporção é 2, pois  $2 \times 100 = 200$ . Portanto, para converter o valor de `timeDivision.metric` do objeto `Midi` para 200 é necessário multiplicar os valores de `delta time` de todos os eventos de todas as `tracks` do objeto `Midi` por 2. Esta normalização se faz necessária para que, no momento de junções de objetos `Midi` durante a fase de composição, os eventos MIDI se mantenham consistentes em relação à duração dos eventos de cada composição.

### 3.3 CONFIGURAÇÃO DE COMPOSIÇÃO

Uma vez que os dados do objeto `MusicalCompositionConfig` foram carregados pelo processo de carga de dados, a etapa de parametrização do processo de composição se inicia. Nesta etapa, o usuário pode determinar valores para parâmetros de opções, linhas, passos ou fonte de composição. Esses parâmetros serão utilizados durante o processo de composição para determinar quais valores o usuário pode aplicar às propriedades sonoras da composição. Os parâmetros também podem determinar o tamanho mínimo das opções em cada passo ou determinar se dados serão exibidos ou não durante o processo de composição. O funcionamento dos componentes configurados pelos parâmetros descritos nesta seção será abordado com mais detalhes na próxima seção. O APÊNDICE D demonstra a tela de configuração do aplicativo e seus valores para os quatro grupos de parâmetros. Todos os parâmetros alterados nesta tela alteram os atributos do objeto `MusicalCompositionConfig`. Inicialmente os valores em questão podem ser valores padrão, caso não tenha sido carregado o arquivo de configuração, ou os valores carregados do arquivo de configuração. A medida que o usuário efetua alterações nos componentes das telas de configuração esses dados são alterados.

Os parâmetros gerais da composição são os que definem os valores que podem ser aplicados para atributos de propriedades sonoras que afetam toda a fonte de composição. Além disso, este grupo de parâmetros possui um parâmetro de controle de exibição do card de informações da composição. Os parâmetros deste grupo afetam o comportamento de três componentes na tela de composição. O primeiro componente é o `range` do `Ionic` responsável por determinar a propriedade de andamento da composição. O segundo componente é o componente `ListPopoverComponent` que determina o valor para a tonalidade da composição. O último componente é o `card` do `Ionic` em que são exibidas informações sobre a composição. Os componentes `Ionic` e parâmetros gerenciados por este grupo de são:

- a) valor mínimo de andamento: determina o valor mínimo que o usuário pode escolher para a propriedade de andamento da composição. A entrada do usuário é realizada através de um componente `range` do `Ionic`. Os valores mínimo e máximo para este parâmetro são definidos como constantes na classe `Midi`. O valor definido pelo usuário será utilizado como o valor mínimo do componente `range` que representa o andamento em uma composição;
- b) valor máximo de andamento: determina o valor máximo que o usuário pode escolher para a propriedade de tempo da composição. Os valores mínimo e máximo para este parâmetro são definidos como constantes na classe `Midi`. A entrada do usuário é realizada através do mesmo componente `range` do parâmetro de valor mínimo. O valor definido pelo usuário será utilizado como o valor máximo do componente `range` que representa o tempo em uma composição;
- c) quantidade do número de incrementos: determina o fator de incremento e decremento utilizado pelo componente `range` do `Ionic` que o usuário irá utilizar na tela de composição para determinar o valor de andamento. O valor deste parâmetro deve estar entre 1 e metade da diferença entre os valores máximo e mínimo determinados nos dois primeiros parâmetros;
- d) valor inicial de andamento: determina o valor que o parâmetro de andamento terá no início da composição. Este valor deve estar entre os valores dos dois primeiros parâmetros;
- e) tonalidades para escolha: determina quais tonalidades podem ser selecionadas pelo usuário na tela de composição. A entrada do usuário é realizada através de um componente `select` do `Ionic` configurado para aceitar seleção de múltiplos valores. Este parâmetro é uma lista de tonalidades que é utilizada como parâmetro para o componente `ListPopoverComponent`, que é utilizado na seleção da tonalidade. Os valores desta lista de parâmetro são definidos como constantes na classe `Midi`;
- f) tonalidade padrão: determina o valor inicial de tonalidade. A entrada do usuário é realizada através de um

componente `select` do Ionic configurado para não aceitar seleção de múltiplos valores. Os valores da lista de tonalidade são definidos como constantes na classe `Midi`;

- g) exibir dados de composição: determina se o `card` de informações da composição será exibido na tela de composição. A entrada do usuário é realizada através de um componente `checkbox` do Ionic.

Os parâmetros de `passo` não têm efeito sobre as propriedades sonoras da composição musical. Existe somente um parâmetro por `passo`. Cada parâmetro define o tamanho mínimo dos fragmentos musicais do `passo`. A unidade de medida desse valor é o quarto de nota, ou seja, a quantidade de `semínimas` existentes no fragmento musical. Portanto, o nome do parâmetro é quantidade de quartos de nota. Este parâmetro será utilizado na configuração da composição musical para garantir que, no mínimo, cada fragmento do `passo` possua a quantidade parametrizada de quartos de nota. Este parâmetro se faz necessário para casos em que o fragmento musical é finalizado com pausas ao invés de notas. Neste caso o arquivo `Midi` pode ser finalizado sem manter os dados de pausas. O parâmetro em questão permite que o aplicativo insira estes dados de pausas se eles não existirem. A entrada destes parâmetros é realizada por componentes de entrada que aceitam somente valores numéricos.

Os parâmetros de `linhas` controlam valores referentes ao parâmetro de volume de cada `linha` da composição. Os parâmetros de `linha` afetam o comportamento de um componente na tela de composição para cada `linha`. Os componentes são `ranges` do Ionic, responsáveis por determinar a propriedade de volume da `linha`. Os parâmetros gerenciados por este grupo de parâmetros, para cada `linha` de composição, são:

- a) valor mínimo de volume: determina o valor mínimo que o usuário pode escolher para a propriedade de volume da `linha`. A entrada do usuário é realizada através de um componente `range` do Ionic. Os valores mínimo e máximo para este parâmetro são definidos como constantes na classe `Midi`. O valor definido pelo usuário será utilizado como o valor mínimo do componente `range` que representa o volume em uma `linha`;
- b) valor máximo de volume: determina o valor máximo que o usuário pode escolher para a propriedade de volume da `linha`. Os valores mínimo e máximo para este parâmetro são definidos como constantes na classe `Midi`. A entrada do usuário é realizada através do mesmo componente `range` do parâmetro de valor mínimo. O valor definido pelo usuário será utilizado como o valor máximo do componente `range` que representa o volume em uma `linha`;
- c) quantidade de número de incrementos: determina o fator de incremento e decremento utilizado pelo componente `range` do Ionic que o usuário irá utilizar na tela de composição para determinar o valor de volume de `linha`. O valor deste parâmetro deve estar entre 1 e metade da diferença entre os valores máximo e mínimo determinados nos dois primeiros parâmetros;
- d) valor inicial de volume: determina o valor que o parâmetro de volume terá no início da composição para a `linha`. Este valor deve estar entre os valores dos dois primeiros parâmetros.

Os parâmetros de `opção` da composição são os que definem os valores que podem ser aplicados para atributos de timbre das opções. Os parâmetros de `opção` afetam o comportamento de um componente na tela de composição para cada `opção`. Os componentes são `ListPopoverComponent`, responsáveis por determinar a propriedade de timbre da `opção`. Os parâmetros gerenciados por este grupo de parâmetros, para cada `opção` da composição, são:

- a) instrumentos para escolha: determina quais instrumentos musicais podem ser selecionados pelo usuário na tela de composição. A entrada do usuário é realizada através de um componente `select` do Ionic configurado para aceitar seleção de múltiplos valores. Este parâmetro é uma lista de instrumentos que é utilizada como parâmetro para o componente `ListPopoverComponent`, que é utilizado na seleção do instrumento musical. Os valores desta lista de parâmetro são definidos no processo de carga de dados de composição;
- b) instrumento padrão: determina o valor inicial de instrumento musical. A entrada do usuário é realizada através de um componente `select` do Ionic configurado para não aceitar seleção de múltiplos valores. Os valores desta lista de parâmetro são definidos no processo de carga de dados de composição.

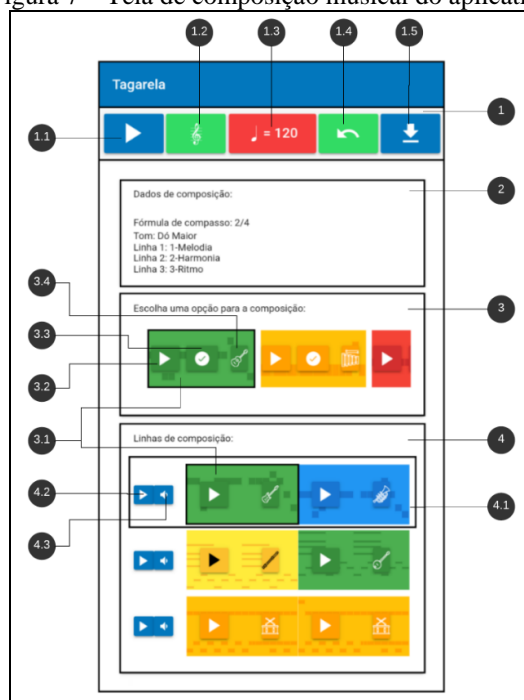
O processo de parametrização da composição se encerra quando o botão `começar composição` é pressionado. Nesse momento o arquivo de configuração é persistido, de acordo com o tipo da fonte de composição e o objeto `MusicalCompositionControl` é instanciado. Esta classe de controle recebe o objeto `MusicalCompositionConfig` como parâmetro e realiza a configuração do objeto `MusicalComposition`. O processo de instância desse objeto importa todos os valores de configuração para a classe de composição. Nesta etapa também é criada a estrutura de matriz de opções. A matriz de opções é uma matriz tridimensional que é construída para realizar o mapeamento de todas as opções disponíveis para todos os passos e linhas da composição. Esta matriz é um atributo da classe `MusicalCompositionControl` e é criada através das opções disponíveis no objeto `MusicalCompositionConfig`. Para cada opção adicionada na matriz, o tamanho dos objetos `Midi` é ajustado de acordo com o parâmetro de tamanho mínimo de fragmento musical. O atributo de `espectro` de cada opção também

é criado neste momento. O espectro é representado pela classe `MidiSpectrum`. Esta classe armazena dados sobre duração total da opção e de cada nota musical presente na opção. Estes dados serão utilizados posteriormente na geração da imagem do espectro. A imagem em questão é abordada na próxima seção.

### 3.4 PROCESSO DE COMPOSIÇÃO

Uma vez que o processo de configuração é concluído, a tela de composição musical é exibida. Neste ponto as classes de controle e modelo de composição já estão instanciadas e com os dados carregados para que o processo de composição se inicie. A partir deste momento a atividade apresentada pelo fluxograma da Figura 4 se inicia. Nesta etapa o usuário interage com a representação dos elementos de composição existentes na tela do aplicativo para montar sua composição personalizada e editar as suas propriedades sonoras. A Figura 7 apresenta a tela de composição do aplicativo com seus componentes identificados por números. Esta seção referenciará estes componentes para descrever as particularidades de cada ação de composição ou exibição de dados.

Figura 7 – Tela de composição musical do aplicativo



Fonte: elaborado pelo autor.

A tela de composição pode ser dividida em quatro macro partes. A primeira delas, representada no item 1, é a barra de controle geral da composição. A segunda parte é o card que exibe os dados de composição e é identificada no item 2. A terceira parte, representada no item 3, é o card onde se apresentam as opções. A quarta parte é o card onde se apresentam as opções escolhidas pelo usuário alocadas nas linhas e posições correspondentes à escolha. Por questões de ordenação e sequência do processo de composição, os cards serão apresentados na sequência de itens: 3, 4, 1 e 2.

O objeto `MusicalCompositionControl`, que é a classe de controle da composição, possui como atributos duas variáveis numéricas para controlar em qual etapa de composição o aplicativo está. Uma das variáveis controla o índice do passo e a outra o índice da linha. Estas variáveis são utilizadas para acessar, através da matriz de opções, as opções que são apresentadas no item 3. Cada uma das opções, que são representadas pelo item 3.1, são exibidas em forma de um retângulo. Este objeto é controlado pelo objeto `ChoiceComponent`. A imagem exibida no fundo deste item é a representação do espectro do objeto `Midi` associado à opção. Para gerar esta imagem os dados do espectro da opção são percorridos e transformados em uma imagem *Scalable Vector Graphics* (SVG). Este processo é realizado na função `getEncodedSVGSpectrum` do provider `MidiSpectrumSvgProvider`. A altura da imagem criada é um valor fixo. Neste espaço são criadas linhas para representar as notas. A quantidade de linhas criadas é a diferença entre a nota musical mais aguda e a mais grave da opção. Para melhorar a visualização do espectro são adicionadas duas linhas vazias, uma no limite superior e uma no limite inferior da imagem. Já o comprimento da imagem é relativo à quantidade de quartos de nota existentes na opção. Cada nota musical existente na opção é representada na linha que corresponde à nota musical por um retângulo de cor mais clara que o fundo da imagem. O comprimento do retângulo é relativo à duração da nota musical na composição. As cores de fundo da imagem e de nota musical são definidas de acordo com o instrumento musical que está associado à opção. A relação entre as cores e os instrumentos musicais está disposta no APÊNDICE B.



Cada representação de opção contém três botões representados por: item 3.2, item 3.3 e item 3.4. O item 3.2 é o botão que inicia a reprodução do objeto Midi. Quando este botão é pressionado o método `playMidi` da classe `ChoiceComponent` é executado. Este método executa o método `applyOptionChanges` da classe `MusicalCompositionControl` passando como parâmetro o objeto Midi associado ao componente `ChoiceComponent`. A função `applyOptionChanges` aplica ao objeto passado por parâmetro as alterações de propriedades sonoras de timbre, duração e altura. Após as alterações de propriedades sonoras, a função `playMidi` inicia o componente `PlayMidiComponent` passando como parâmetro o objeto Midi alterado e seu espectro.

As alterações de propriedades sonoras são realizadas por métodos dos objetos Midi. Os parâmetros utilizados para estas alterações são atributos da classe `MusicalComposition` ou de seus atributos. Quando uma alteração desta natureza é realizada o objeto Midi é o responsável por realizar todas as alterações em eventos MIDI de suas tracks. Com isto, as classes de controle dos componentes gráficos não necessitam tomar conhecimento sobre os detalhes de estrutura da classe Midi. O Quadro 6 apresenta uma descrição das funções de alteração de propriedades sonoras de objetos Midi que são utilizadas no processo de composição.

Quadro 6 – Descrição das funções de alteração de propriedades musicais

Propriedade sonora	Método da classe Midi	Funcionamento
Timbre	<code>applyInstrumentChange</code>	Recebe como parâmetro o número do instrumento musical. Para cada track do objeto Midi, elimina os eventos de definição de instrumento musical e adiciona, como o primeiro evento da track, um novo evento de definição de instrumento musical de acordo com o parâmetro.
Intensidade	<code>applyVolumeChange</code>	Recebe como parâmetro o valor percentual que deve ser aplicado ao volume. Para cada track do objeto Midi, para cada evento de início de execução de nota musical, calcula volume do evento utilizando o parâmetro e o atributo do volume original do evento. Por fim atribui o valor obtido para o valor de volume do evento.
Duração	<code>applyTempoChange</code>	Recebe como parâmetro o valor de andamento. Para cada track do objeto Midi, edita o evento que determina a velocidade de execução de acordo com o parâmetro.
Altura	<code>applyNoteTranspose</code>	Recebe como parâmetro o número da tonalidade. Para cada track do objeto Midi, edita o evento que determina a tonalidade, de acordo com o parâmetro. Calcula o fator de conversão de acordo com a diferença entre o parâmetro e o valor atual. Para cada evento de início ou finalização de execução de nota musical aplica o fator de conversão.

Fonte: elaborado pelo autor.

Toda a execução de objetos Midi é realizada pelo componente `PlayMidiComponent`. Quando este componente é iniciado ele realiza a gravação do arquivo Midi na área temporária do aplicativo e executa o arquivo gerado. Para tanto, são utilizados os plugins `File` e `Media` respectivamente. Juntamente com a execução do áudio do objeto Midi, este componente exibe o espectro recebido por parâmetro e um indicador que se move sobre o espectro para demonstrar qual parte do espectro está em execução. Desta forma, além da representação sonora, o usuário também tem acesso à uma representação gráfica do áudio. A altura do espectro exibido pelo componente em questão é sempre a mesma, independentemente da quantidade de notas existentes no espectro. Já o comprimento do espectro depende da quantidade de quartos de nota existentes no objeto Midi. Sendo assim, é possível que o espectro não seja exibido todo na tela do dispositivo. Quando isto ocorre, enquanto do objeto Midi está sendo executado, o indicador de execução navega até o centro da tela do dispositivo e para. Neste momento a imagem do espectro começa a se mover horizontalmente para a esquerda até que o fim do espectro seja exibido no canto direito da tela. Agora o indicador continua seu movimento até o fim do espectro. Desta forma, o espectro todo é exibido e a indicação de execução permanece consistente entre o áudio e a imagem.

O item 3.3 é o botão que realiza a escolha do fragmento musical como parte componente da composição musical. Quando este botão é utilizado o método `applyChoice` do objeto `ChoiceComponent` é executado. Este método executa o método `applyChoice` do objeto `MusicalCompositionControl` passando como parâmetro a opção associada ao componente em questão. O objeto `MusicalCompositionControl` atribui a opção selecionada para a linha que está identificada pelo índice de linha. O índice de linha é então incrementado. Se a opção pertence à última linha, o índice de linha é reiniciado e o índice de passo é incrementado. Se o índice de passo for

incrementado para um valor maior que a quantidade de passos existentes na composição, a composição é então finalizada. Com as atualizações dos índices o item 3 atualiza sua lista de opções disponíveis.

O item 3.4 é o botão que altera o instrumento musical que está associado com a opção. Quando o usuário clica neste botão, o componente `ListPopoverComponent` é criado para que a lista de instrumentos musicais seja exibida para que o usuário realize a escolha de um instrumento. Este componente recebe como parâmetro, além de uma lista de valores que devem ser exibidos, duas funções e um método. A primeira das funções, que é o atributo `iconFunction` do componente, é utilizada para obter o ícone que deve ser exibido na frente de cada opção da lista. A segunda função, armazenada pelo atributo `nameFunction`, é utilizada para identificar o nome que deve ser exibido na lista. O método é o atributo `callback`, que é a responsável por atribuir o valor escolhido pelo usuário ao atributo de composição. Todos estes atributos recebem como parâmetro um dos valores da lista de valores recebida por parâmetro. No caso do item 3.4 a lista de valores a ser exibida é a lista de instrumentos musicais definidos na fase de configuração da composição para a opção. A função de busca de ícone é a `getIconToMidiNumber` do provider `VisualMidiProvider`. Do mesmo provider, a função `getInstrumentNameToMidiNumber` é utilizada para identificar o nome do elemento da lista. Já o método que atribui a escolha do usuário ao atributo de instrumento musical da opção é o `changeInstrumentMidiNumber` do objeto `ChoiceComponent`. O item 3.1 utiliza a cor do instrumento musical escolhido como tema para os botões. A relação de instrumentos musicais e cores está disponível no APÊNDICE B.

O item 4 é o card que representa as linhas da composição musical. Para cada linha, um item 4.1 é criado. Assim como no item 3, os itens 3.1 também são exibidos no item 4.1, porém sem o item 3.3 associado. Cada item 3.1 representa as opções já escolhidas pelo usuário para a linha da composição. A ordem de apresentação das opções segue a ordem em que o usuário efetuou a escolha. Além da representação das opções que compõem a linha, no início dos itens 4.1, existem dois botões. Estes botões, item 4.2 e item 4.3, representam as opções de execução da linha e alteração do volume da linha respectivamente. Quando item 4.2 é acionado, o processo de execução de todas as opções que compõem a linha se inicia. Quando isto ocorre, o processo de execução do áudio é o mesmo que ocorre no item 3.2. Porém é necessário criar um novo arquivo Midi e concatenar os espectros de todas as opções envolvidas para representar a linha em questão. Para isso, o método `playMidi` do componente `LineControlComponent` executa o método `applyLineChanges` da classe `MusicalCompositionControl`. Este método realiza as alterações das propriedades sonoras de intensidade, timbre, duração e altura em todas as opções da linha passada como parâmetro e atualiza o atributo `midi` da linha com um novo objeto `Midi`. O novo objeto `Midi` é obtido através da função `concatenateMidisChannels` da classe `MidiControl`, que é atributo da classe `MusicalCompositionControl`. Esta função recebe dois objetos `Midi` como parâmetro e retorna um novo objeto `Midi` que é resultado da concatenação das tracks de cada objeto recebido por parâmetro. O objeto `Midi` que será retornado é um clone do primeiro parâmetro. Após todos os atributos do primeiro parâmetro serem atribuídos ao objeto de retorno, os eventos de finalização de track são removidos do objeto de retorno. O `delta time` do evento em questão é somado ao primeiro evento do segundo parâmetro. Todos os eventos de início ou finalização de notas musicais, determinação de instrumento musical ou finalização de track do segundo parâmetro são adicionados ao fim da track do objeto de retorno. Os eventos que não são importados transbordam seu valor de `delta time` para o próximo evento que será considerado.

O espectro que representa a linha é obtido através da função `concatenateSpectrums` do provider `MidiSpectrumSvgProvider`. Esta função recebe como parâmetro uma matriz de espectros. Para cada linha da matriz em questão é reservada uma porcentagem igual da altura da imagem. Esta divisão fornece para cada linha da matriz o mesmo espaço, independentemente da quantidade de notas existentes em cada linha da matriz. Cada coluna do parâmetro representa o espectro de uma opção da linha. Assim como na função `getEncodedSVGSpectrum`, cada linha da imagem, dentro de cada linha do parâmetro, representa uma nota. Porém na função `concatenateSpectrums` a quantidade de linhas não é definida em relação aos limites de nota do espectro de uma opção, mas sim em relação a diferença de notas de todos os espectros de uma linha. Uma vez definidas as quantidades de linhas de cada linha do parâmetro, as notas dos espectros são atribuídas à linha que representa a nota musical. O comprimento da imagem é determinado pela maior quantidade de quartos de notas entre as linhas do parâmetro. O espaço ocupado por cada nota musical é determinado por sua duração. As cores utilizadas para representar cada uma das opções que compõem a imagem são definidas pelo instrumento musical associado à opção, assim como na função `getEncodedSVGSpectrum`;

O item 4.3 permite ao usuário alterar o volume da linha. Para tanto, quando o botão representado pelo item é pressionado, o componente `SlidePopoverComponent` é iniciado. Este componente exibe um elemento `ion-range` do Ionic. Os parâmetros de valor mínimo, valor máximo e quantidade de passos deste componente são os definidos na fase de configuração da composição. Assim como no `ListPopoverComponent`, o componente em questão recebe como parâmetro um método para atualizar o atributo de composição. O método utilizado neste momento é o

`changeVolume` do componente `LineControlComponent`. Quando o botão salvar do componente `SlidePopoverComponent` é pressionado, o método de alteração de volume é executado recebendo como parâmetro o valor de entrada definido pelo usuário.

O item 1 é o item que contém as opções que aplicam ações gerais à composição. O item 1.1 é o botão que executa a composição completa. O item 1.2 é o botão que aplica alterações de tonalidade. O item 1.3 é o botão que altera o tempo da composição. O item 1.4 realiza o download da composição. O item 1.1 realiza a execução da composição pelo mesmo componente que realiza a execução da opção ou da linha. Quando este botão é acionado o método `playMidi` do componente `GeneralControlComponent` é executado. Este método executa o método `applyGeneralChanges` da classe `MusicalCompositionControl`. O método `applyGeneralChanges` é responsável por aplicar todas as alterações de propriedades sonoras para todas as linhas da composição, além de atualizar o objeto `Midi` de cada uma das linhas. Uma vez que todas as linhas estão com os objetos `Midi` atualizados a função `concatenateMidisInTracks` do objeto `MidiControl` é executada. Esta função recebe como parâmetro um array de `Midi` e retorna um novo objeto `Midi`. Este novo objeto contém todas as tracks existentes nos arquivos do array. Portanto este objeto permite a execução de todas as linhas de composição de forma simultânea. O espectro que será passado de parâmetro para o componente `PlayMidiComponent` é gerado da mesma forma que o espectro da linha. Porém a matriz de espectro pode possuir mais de uma linha neste caso, diferente do caso da execução da linha. O item 1.1 é exibido somente se a primeira escolha já foi realizada pelo usuário.

O item 1.2 é o botão responsável por realizar a alteração da tonalidade da composição. Quando este botão é acionado, assim como ocorre com o item 3.4, um componente `ListPopoverComponent` é criado. Este componente recebe como parâmetro a lista de tonalidades definida na fase de configuração da composição musical. A função para obter o nome da tonalidade é a `getKeySignatureName` do provider `VisualMidiProvider`. A função `getIonIconToKeySignatureNumber` do mesmo provider é utilizada para obter o ícone de cada item da lista. O método que altera a propriedade de tonalidade da composição é o `setKeySignature` do componente `GeneralControlComponent`. A função `getIonIconToKeySignatureNumber` também é responsável por determinar o ícone que é exibido no botão em questão em razão da tonalidade atual da composição.

O item 1.3 é responsável por realizar a alteração de andamento da composição. Este botão, assim como o item 4.3, quando acionado exibe um objeto `SlidePopoverComponent`. Os parâmetros utilizados para este objeto são os valores determinados na fase de configuração da composição. Neste objeto o usuário pode alterar o valor de tempo e, ao clicar no botão salvar, o método `setTempo` do objeto `GeneralControlComponent` se encarrega de atualizar o atributo de andamento da composição. O item 1.3 exibe a figura de nota que corresponde ao quarto de nota e a velocidade atual da composição, seguindo o modelo de indicação metronômica.

O item 1.4 é responsável por desfazer a última escolha do usuário. Quando este botão é utilizado a última escolha realizada pelo usuário é removida dos atributos da composição. Com estas alterações os elementos da tela de composição são atualizados. Também é realizado o decremento dos índices de controle de linha e passo atuais da classe `MusicalCompositionControl`. Esta opção é executada pelo método `undo` do componente `GeneralControlComponent`. Este método, por sua vez, executa o método `undoChoice` do objeto `MusicalCompositionControl`. O item 1.4 somente é exibido quando o usuário já iniciou a composição.

O último botão, o item 1.5, é o botão que realiza o download da composição. Este botão executa o método `downloadComposition` do componente `GeneralControlComponent`. Quando este botão é acionado ele inicia o componente `DownloadMidiComponent`. Este componente é responsável por solicitar ao usuário o nome do arquivo que será salvo, confirmar a possível sobrescrição do arquivo, e utilizar o provider `FileProvider` para realizar a gravação do arquivo na pasta de downloads do dispositivo. O objeto `Midi` é obtido através de parâmetro e o arquivo `Midi` é gerado através da gravação da string binária objeto `Midi`. A string binária é obtida através da execução da função `getBinaryString` da classe `MidiFileControl`, que é um atributo da classe `DownloadMidiComponent`. A função `getBinaryString` realiza a conversão de todos os dados da classe `Midi` e de seus atributos para valores em hexadecimal. Este processamento é realizado conforme as regras do protocolo `Midi`. Após isso, o valor obtido é convertido para a string binária que será persistida. O objeto `Midi` que é passado como parâmetro para o componente `DownloadMidiComponent` é obtido pelo mesmo método que gera o objeto `Midi` para a execução de toda a composição. O item 1.5 somente está disponível quando todas as etapas de composição forem concluídas.

O item 2 é o card responsável por apresentar ao usuário dados sobre a composição. Este card é exibido, ou não, de acordo com a escolha realizada na fase de configuração. Os dados exibidos são referentes a propriedades globais da composição ou dados atuais sobre as etapas de composição. Os dados exibidos são:

- a) fórmula de compasso: valor determinado pela leitura do evento `MIDI` responsável por armazenar estes dados;

- b) tom: valor determinado pela leitura do evento MIDI responsável por armazenar dados de tonalidade;
- c) linha atual: valor do índice de linha do objeto `MusicalCompositionControl`;
- d) quantidade de linhas: quantidade total de linhas da composição;
- e) passo atual: valor do índice de passo do objeto `MusicalCompositionControl`;
- f) quantidade de passos: quantidade total de passos da composição;
- g) nome das linhas: nome de cada linha existente na composição.

## 4 RESULTADOS

Este capítulo apresenta os testes realizados com o aplicativo. Serão apresentadas três seções para abordar os testes realizados das principais funcionalidades, dos testes com profissionais da área de música e dos testes em atividades terapêuticas.

### 4.1 TESTES DE FUNCIONALIDADES

Testes meus:

- Configuração de composições personalizadas;
- iOS

### 4.2 TESTES DE UTILIZAÇÃO POR PROFISSIONAIS

Para validar os aspectos de utilização do aplicativo e verificar a possibilidade de utilização dele como ferramenta durante um processo de terapia, foram realizadas duas entrevistas com dois profissionais. O primeiro entrevistado foi o professor Eusébio..., que é professor de música na FURB. O segundo entrevistado foi o musicoterapeuta Naldo Nogueira, que atua em Blumenau com aulas de música e atividades terapêuticas para criação autistas. Para ambas as entrevistas foi utilizado o mesmo roteiro de apresentação do aplicativo. Este roteiro está disponível no apêndice XXXX. As entrevistas tiveram como propósito apresentar a dinâmica proporcionada pelo aplicativo, explorar todas as funcionalidades do processo de composição e do processo de configuração e obter um retorno dos profissionais em relação a sugestões de melhoria e viabilidade de utilização.

Na primeira entrevista, realizada com o professor Eusébio, todas as funcionalidades dispostas em tela funcionaram conforme o esperado. As ações de todos os botões foram entendidas sem a necessidade de explicações. Porém, a dinâmica de composição não ficou clara somente com uma explicação inicial. Foi necessário realizar as escolhas de opções para todas as linhas do primeiro passo para que o entendimento total da atividade ocorresse. Quando o processo de composição do segundo passo foi iniciado, a dinâmica foi entendida e daí em diante não surgiram mais dúvidas em relação ao processo de composição. Nos testes de configuração também não houveram questionamentos, com exceção dos parâmetros de passo, e todos os atributos editados durante a fase de composição funcionaram conforme o esperado. Em relação aos parâmetros de passo, que realizam a adequação do tamanho dos fragmentos musicais, ocorreu um questionamento em relação à possibilidade de não disponibilizar ao usuário esta funcionalidade. Neste caso, o próprio aplicativo realizaria ajustes de acordo com a análise de todos os fragmentos existentes em um passo. Foi explicado que, devido não ser possível garantir que os fragmentos musicais possuam sempre todos os dados necessários para o fim de um fragmento musical, esta opção foi mantida como um parâmetro de configuração. As sugestões de melhorias realizadas pelo professor Eusébio estão dispostas no quadro XXX. Quando questionado a respeito da viabilidade da utilização do aplicativo em um processo de terapia o professor respondeu que parece viável, mas que ele não teria propriedade suficiente sobre o assunto para afirmar, já que está não é sua área de atuação.

Quadro 7 – Descrição das funções de alteração de propriedades musicais

Sugestão	Justificativa
Não exibir a página de configuração para o usuário que realizará a atividade de composição.	A alteração de parâmetros deve ser realizada por quem está configurando a composição para o usuário. Caso ele altere dados de configuração a composição pode perder o propósito para o qual ela foi construída.
Repensar a função do card de informações da composição.	Não está tão amigável quanto o restante da página de composição. Para a dinâmica de composição talvez seria mais interessante apresentar os dados de composição nos próprios elementos de composição.
Possibilitar a troca de uma opção por outra sem necessitar utilizar o botão voltar.	
Exibir o indicador de execução no card que apresenta as opções já escolhidas	

Aprimorar os controles de execução para que seja possível, através do indicador de execução selecionar de qual trecho a composição deve iniciar a tocar. E possibilitar parar a execução.	
Repensar o formato com que a dinâmica se apresenta ao usuário.	

Fonte: elaborado pelo autor.

A segunda entrevista, realizada com o musicoterapeuta Naldo, obteve resultados similares à entrevista com o professor Eusébio. Ambos conseguiram utilizar todas as funcionalidade do aplicativo. Uma diferença entre as duas entrevistas foi o dispositivo utilizado para a demonstração. Enquanto na entrevista com o professor Eusébio foi utilizado um tablet, na entrevista com o Sr. Naldo foi utilizado um celular. Devido ao tamanho reduzido dos elementos em tela o usuário sentiu dificuldade para realizar algumas ações com botões muito próximos. A dinâmica também não foi completamente entendida somente com a primeira explicação. Foi necessário, assim como na primeira entrevista, realizar todas as escolhas do primeiro passo para que o processo fizesse sentido para o usuário. Após isso nenhuma dificuldade foi enfrentada. Quando questionado sobre a possibilidade de utilização do aplicativo em algum processo de terapia, a resposta foi positiva. Segundo o musicoterapeuta, é necessário que o aplicativo consiga prender a atenção do usuário, e as cores utilizadas e a representação dos arquivos MIDI em espectros coloridos devem ajudar. Devido às questões de usabilidade, o Sr. Naldo ressaltou que é necessário que o usuário possua idade suficiente para entender os conceitos envolvidos na dinâmica de composição. Para efetuar testes com o usuário final, o Sr. Naldo se disponibilizou para realizar testes com alunos autistas. Estes testes são abordados na próxima seção. Nenhuma sugestão de melhoria para o aplicativo foi realizada.

#### 4.3 TESTES DE UTILIZAÇÃO EM ATIVIDADE TERAPÊUTICA

**Resultado dos testes com os alunos do Naldo.**

### 5 CONCLUSÕES

As conclusões devem refletir os principais resultados alcançados, realizando uma avaliação em relação aos objetivos previamente formulados. Deve-se deixar claro se os objetivos foram atendidos, se as ferramentas utilizadas foram adequadas e quais as principais contribuições do trabalho sociais ou práticas para o seu grupo de usuários bem como para o desenvolvimento científico e ou tecnológico da área.

Deve-se incluir também as limitações e as possíveis extensões do TCC.

#### REFERÊNCIAS

- BARCELLOS, Lia Rejane Mendes. **Musicoterapia**: alguns escritos. Rio de Janeiro: Enelivros, 2004.
- BRUSCIA, Kenneth, E. **Definindo musicoterapia**. 3. ed. Dallas: Barcelona Publishers, 2016.
- CAZAGRANDA, Lucas. 2016. **Aprendendo Braille**: o ensino do sistema Braille com o uso do tagarela. 2016. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- CORRÊA, Ana Grasielle Dionísio et al. Introdução ao GenVirtual: uma interface musical com realidade aumentada para apoiar o “fazer musical” de pessoas com deficiência motora e cognitiva. **Revista Brasileira de Informática na Educação**, Rio de Janeiro, v. 21, n. 2, p. 118-131, 2013.
- FABENI, Alan Filipe Cardozo. **Tagarela**: aplicativo para comunicação alternativa no IOS. 2012. 106 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Universidade Regional de Blumenau, Blumenau.
- FERREIRA, André Felipe. **Tagarela**: módulo jogo de letras e número. 2016. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- HASS, Jeffrey. **Introduction to Computer Music**: Volume One. 2013. Disponível em: <[http://www.indiana.edu/~emusic/etext/MIDI/chapter3\\_MIDI.shtml](http://www.indiana.edu/~emusic/etext/MIDI/chapter3_MIDI.shtml)>. Acesso em: 08 set. 2017.
- ILARI, Beatriz Senoi; ARAÚJO, Rosane Cardoso de. **Mentes em música**. Curitiba: UFPR, 2010.
- IONIC FRAMEWORK. **The top open source framework for building amazing mobile apps**. 2013. Disponível em: <<https://ionicframework.com/>>. Acesso em: 08 set. 2017.

LACERDA, Osvaldo. **Compendio de teoria elementar da música**. 3.ed. São Paulo: Ricordi Brasileira, 1967.

LIMA, Sandra Fernandes de Oliveira. **Um sistema para transposição automática de sequências midi baseada em alcance vocal**. 2006. 233 f. Dissertação (Mestrado em Ciências). Universidade Federal de Uberlândia, Uberlândia.

MARCO, Darlan Diego de. **Tagarela**: aplicativo de comunicação alternativa na plataforma Android. 2014. 93 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Universidade Regional de Blumenau, Blumenau.

MARQUES, Elvis Merten. **Tagarela 2.0**: framework de comunicação alternativa, módulo de jogos. 2014. 47 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MED, Bohumil. **Teoria da música**. 4. ed. Brasília: Musimed, 1996.

MIDI ASSOCIATION. **MIDI news, resources and press**. 2017. Disponível em: <<https://www.midi.org>>. Acesso em: 08 set. 2017.

MONTEIRO, Raquel Sofia Carvalho. **A musicoterapia em contexto escolar**: perturbações do comportamento, espectro do autismo e multideficiência. 2016. 69 f. Relatório de Estágio (Mestrado em Musicoterapia). Faculdade de Ciências Humanas e Sociais, Universidade Lusíada de Lisboa, Lisboa.

PRIOLLI, Maria Luisa de Mattos. **Princípios básicos da música para a juventude**. 48. ed. Rio de Janeiro: Casa Oliveira de Músicas LTDA., 2006.

QUEIROZ, Gregorio Jose Pereira De. **A música compõe o homem, o homem compõe a música**. São Paulo: Editora Cultrix, 2000.

REETZ, Wagner Jean. **Jogo de letras/números voltado para tecnologia assistiva no Android**. 2013. 63 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Universidade Regional de Blumenau, Blumenau.

SAUTNER, Guilherme. **Tagarela**: módulo de desenvolvimento e aquisição de linguagem para crianças autistas. 2017. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

WATANABE, Marli Kiyoko Fujikawa; TSUKIMOTO, Denise Rodrigues; TSUKIMOTO, Gracinda Rodrigues. Terapia ocupacional e o uso do computador como recurso terapêutico. **Acta Fisiátrica**, São Paulo, v. 10, n. 1, p. 17-20, 2003.

WIPPEL, André Filipe. **Tagarela**: integração e melhorias no aplicativo de rede de comunicação alternativa. 2015. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

## APÊNDICE A – DIAGRAMA DE CLASSES

**Posso fazer uma versão reduzida mesmo? Acho a versão final (+/- 55 classes) não vai caber**


Este apêndice apresenta o diagrama de classes do aplicativo. A Figura XXX apresenta o diagrama em formato reduzido em virtude do tamanho do diagrama completo. São abordados neste apêndice somente as classes relevantes para complementar a descrição do aplicativo realizada no capítulo 3. A versão completa do diagrama de classes está disponível em XXXX.

Diagrama
















## APÊNDICE B – RELAÇÃO DE INSTRUMENTOS MUSICAIS UTILIZADOS PELO APLICATIVO

Este apêndice apresenta os instrumentos musicais, os ícones que representam cada instrumento e as cores utilizadas nos componentes da tela de composição quando estes estão associados à um instrumento. O Quadro 8 apresenta estes elementos.

Quadro 8 – Instrumentos musicais utilizados no aplicativo

Instrumento Musical	Ícone	Número MIDI	Grupo de instrumentos	Cor de background	Cor de nota e botões
Piano		1	Melódico	#F44336	#D32F2F



Vibrafone		12	Melódico	#FFC107	#FFA000
Xylofone		14	Melódico	#FFC107	#FFA000
Acordeon		22	Melódico	#F44336	#D32F2F
Violão		25	Melódico	#4CAF50	#388E3C
Guitarra		27	Melódico	#4CAF50	#388E3C
Contrabaixo elétrico		34	Melódico	#4CAF50	#388E3C
Violino		42	Melódico	#4CAF50	#388E3C
Harpa		47	Melódico	#4CAF50	#388E3C
Trompete		57	Melódico	#2196F3	#1976D2
Trombone		58	Melódico	#2196F3	#1976D2
Trompa		62	Melódico	#2196F3	#1976D2
Saxofone		66	Melódico	#FFEB3B	#FBC02D
Flauta		74	Melódico	#FFEB3B	#FBC02D
Banjo		106	Melódico	#4CAF50	#388E3C
Bateria		-	Rítmico	#FFC107	#FFA000

Fonte: elaborado pelo autor.

A primeira coluna do Quadro 8 apresenta o nome do instrumento musical. A segunda coluna apresenta o ícone associado à cada instrumento musical. Estes ícones foram adquiridos através da loja virtual da [Iconcrafts \(2018\)](#). O grupo de ícones utilizado foi o `Musical instruments`. O ícone do aplicativo também foi obtido no mesmo grupo de ícones. A terceira coluna apresenta os números que apresentam os instrumentos musicais no protocolo MIDI. A relação entre os instrumentos musicais e os números são dispostos em [MIDI \(1996, p. 150???\)](#). O instrumento musical Bateria não possui um número musical associado, uma vez que todos os dados de execução de instrumentos de percussão são realizados no `channel 10`, conforme abordado em [MIDI \(1996, p. 16???\)](#). A quarta coluna apresenta a qual o agrupamento de instrumentos o instrumento pertence. Este agrupamento é utilizado na fase de configuração da composição musical para determinar quais instrumentos podem ser aplicados para a opção em virtude o `channel` em que a opção mantém seus eventos MIDI.

As colunas de cores apresentam as cores usadas nas imagens de espectro e cores de botões. A quarta coluna apresenta a cor, em hexadecimal, das cores utilizadas no fundo das imagens de espectro de acordo com o instrumento associado à classe `Midi` que será representada. Cada nota representada no espectro utiliza a cor apresentada na quinta coluna. A cor da mesma coluna também define a cor dos botões dos componentes `ChoiceComponent`. As cores foram definidas por meio do sistema de cores apresentado por **Material Design (2018)**. Para determinar a cor de cada instrumento eles foram categorizados de acordo com suas características de produção sonora. Por exemplo, os instrumentos de cordas são representados por cores verdes.

## APÊNDICE C – VALIDAÇÕES DO PROCESSO DE CARGA DE DADOS DE ARQUIVOS MIDI

Este apêndice apresenta a relação de validações realizadas durante o processo de criação das instâncias `Midi` no aplicativo. Este processo ocorre durante o processo de carga de dados de composição e validações são executadas na função `setupMidiFromFile` da classe `MidiFileControl`. As validações estão dispostas no Quadro 9.

Quadro 9 – Validações do processo de carga de dados de arquivos MIDI

Validação	Descrição
Tamanho de <code>delta time</code>	O tamanho da indicação de <code>delta time</code> não pode ser superior ao valor máximo permitido pelo protocolo.
Evento MIDI não identificado	Cada evento MIDI deve possuir os caracteres de identificação mapeados como constantes da classe <code>MidiFileControl</code> .
Mais de um <code>channel</code> por <code>track</code>	O aplicativo não permite que uma <code>track</code> possua notas sendo executadas em mais de um <code>channel</code> .
Mais de um evento <code>Time Signature</code> com valores diferentes entre eles	O aplicativo não suporta alterações de <code>Time Signature</code> em um objeto <code>Midi</code> . Caso mais de um evento deste tipo seja encontrado e os valores de ambos os eventos sejam iguais, somente o primeiro evento é carregado.
Evento <code>Time Signature</code> com valor diferente do evento de mesmo tipo do objeto de comparação obtido por parâmetro	Para que os dados de composição sejam consistentes entre si o aplicativo exige que os eventos <code>Time Signature</code> de todos os objetos <code>Midi</code> de uma fonte de composição possuam os mesmos valores.
Mais de um evento <code>Key Signature</code> com valores diferentes entre eles	O aplicativo não suporta alterações de <code>Key Signature</code> em um objeto <code>Midi</code> . Caso mais de um evento deste tipo seja encontrado e os valores de ambos os eventos sejam iguais, somente o primeiro evento é carregado.
Evento <code>Time Signature</code> com atributo <code>mode</code> diferente do evento de mesmo tipo do objeto de comparação obtido por parâmetro	Para que os dados de composição sejam consistentes entre si o aplicativo exige que os eventos <code>Key Signature</code> de todos os objetos <code>Midi</code> de uma fonte de composição possuam os mesmos valores para o atributo <code>mode</code> .
Mais de um evento <code>End Of Track</code>	Cada <code>track</code> deve possuir somente um evento de finalização.
Evento <code>End Of Track</code> antes do fim da <code>track</code>	O evento de finalização de <code>track</code> deve ocorrer somente no fim da <code>track</code> .
Ausência de evento <code>End Of Track</code>	O aplicativo exige que um evento <code>End Of Track</code> seja encontrado em cada <code>track</code> do arquivo MIDI.
Ausência de evento <code>Time Signature</code>	O aplicativo exige que um evento <code>Time Signature</code> seja encontrado no arquivo MIDI.
Ausência de evento <code>Key Signature</code>	O aplicativo exige que um evento <code>Key Signature</code> seja encontrado no arquivo MIDI.
Ausência de evento <code>Set Tempo</code>	O aplicativo exige que um evento <code>Set Tempo</code> seja encontrado no arquivo MIDI.

Fonte: elaborado pelo autor.

## APÊNDICE D – PÁGINA DE CONFIGURAÇÃO DE COMPOSIÇÃO MUSICAL

**Talvez dividir a imagem em 4 imagens e dispor elas deitadas?**

Este apêndice apresenta a página de configuração de composição musical. A Figura 8 apresenta os quatro grupos de configuração que o aplicativo disponibiliza durante a fase de configuração da composição musical.

Figura 8 – Tela de configuração do aplicativo

Configuração de Composição		Configuração de Composição	
<p>COMEÇAR COMPOSIÇÃO</p> <p>GERAL PASSOS LINHAS OPÇÕES</p> <p>Escolha os parâmetros gerais de composição:</p> <p>Valor mínimo e máximo de tempo (BPM) <span>40</span> <span>240</span></p> <p>1 <span>0</span> <span>500</span></p> <p>Quantidade de números incrementados <span>1</span></p> <p>1 <span>0</span> <span>100</span></p> <p>Valor inicial de tempo <span>120</span></p> <p>40 <span>0</span> <span>240</span></p> <p>Escolha os parâmetros gerais de tonalidade:</p> <p>Tonalidades para escolha: Dób Maior, Solb Maio... ▾</p> <p>Tonalidade padrão: Dó Maior ▾</p>		<p>COMEÇAR COMPOSIÇÃO</p> <p>GERAL PASSOS LINHAS OPÇÕES</p> <p>Escolha os parâmetros de passos de composição:</p> <p>Passo-1</p> <p>Quantidade de quartos de nota</p> <p>8</p> <p>Passo-2</p> <p>Quantidade de quartos de nota</p> <p>8</p> <p>Passo-3</p> <p>Quantidade de quartos de nota</p> <p>8</p> <p>Passo-4</p> <p>Quantidade de quartos de nota</p> <p>8</p>	
<p>Configuração de Composição</p> <p>COMEÇAR COMPOSIÇÃO</p> <p>GERAL PASSOS LINHAS OPÇÕES</p> <p>Escolha os parâmetros das linhas de composição:</p> <p>1-Melodia</p> <p>Valor mínimo e máximo de volume <span>0</span> <span>200</span></p> <p>0 <span>0</span> <span>200</span></p> <p>Quantidade de números incrementados <span>10</span></p> <p>1 <span>0</span> <span>100</span></p> <p>Valor inicial de volume <span>100</span></p> <p>0 <span>0</span> <span>200</span></p>		<p>Configuração de Composição</p> <p>COMEÇAR COMPOSIÇÃO</p> <p>GERAL PASSOS LINHAS OPÇÕES</p> <p>Escolha os parâmetros das opções de composição:</p> <p>Passo: Passo-1</p> <p>Linha: 1-Melodia</p> <p>Opção: S1-M.mid</p> <p>Instrumentos para escolha: Piano, Vibrafone, Xyl... ▾</p> <p>Instrumento padrão: Piano ▾</p> <p>Opção: S1-M.mid (1)</p> <p>Instrumentos para escolha: Piano, Vibrafone, Xyl... ▾</p> <p>Instrumento padrão: Piano ▾</p>	

Fonte: elaborado pelo autor.