

# TAGARELA: MÓDULO DE COMPOSIÇÃO MUSICAL POR MEIO DE MUSICOTERAPIA

Roberto Weege Junior, Dalton Solano dos Reis – Orientador

\*Mover para  
o ponto 1

**Resumo:** Este artigo apresenta o processo de desenvolvimento e testes de um aplicativo que tem como objetivo auxiliar musicoterapeutas na aplicação de atividades de composição musical durante sessões de terapia. O aplicativo foi desenvolvido utilizando o framework Ionic. A atividade de composição musical proporcionada pelo aplicativo consiste em possibilitar que o usuário, que está em atividade terapêutica, escolha fragmentos musicais e edite suas propriedades sonoras. Quando o processo de escolha e edições se finaliza, o usuário tem acesso a uma composição musical personalizada. Para verificar o desempenho do aplicativo, foi realizado um teste em uma aula de música com um aluno autista. O aplicativo alcançou seu objetivo, uma vez que, durante o teste, o aluno conseguiu realizar a utilização do aplicativo e se manteve entretido com o processo de composição.

**Palavras-chave:** Musicoterapia. Aplicativo de musicoterapia. Tagarela. MIDI. Ionic.

## 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 da 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 em atividade terapêutica 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.

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: disponibilizar aos musicoterapeutas uma ferramenta que os auxilie em atividades terapêuticas personalizadas de composição musical; 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 teórica utilizados para a construção do aplicativo. Na primeira seção deste capítulo serão apresentados os conceitos utilizados como base para o desenvolvimento do

O trabalho de Monteiro (2016) trata de questões terapêuticas envolvendo a utilização de música durante o processo de terapia. Algumas das atividades abordadas no trabalho em questão trataram de processos que envolvem criação musical. Apesar do 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 a atividade de composição musical.

*realizado*

Quadro 2 – GenVirtual

|                                |  |
|--------------------------------|--|
| Referência                     | 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, batizado de GenVirtual, 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:<br>a) ARToolkit: responsável pelos processos de realidade aumentada;<br>b) Microsoft Foundation Class Library (MFC): responsável pela interface gráfica com o usuário;<br>c) OpenAL (Open Audio Library): responsável por processos de manipulação de arquivos de áudio;<br>d) API Win32: responsável pela manipulação de mensagens MIDI;<br>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.

X

Quadro 3 – Best Vocal 2005

|                                |   |
|--------------------------------|---|
| Referência                     | Lima (2006).  |
| Objetivos                      | Adequar a tonalidade da música de uma fonte sonora aos limites da extensão vocal do cantor.   |
| Principais funcionalidades     | O software, batizado de Best Vocal 2005, 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. Para tanto, 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 musical proporcionado pelo aplicativo e as ações de alterações de propriedades musicais deste processo. A segunda seção apresenta a carga de dados do aplicativo, detalhando a estrutura

*realizado*

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.

O desenvolvimento do aplicativo foi realizado com o framework Ionic. De acordo com Ionic Framework (2013), o Ionic possibilita que desenvolvedores que sabem desenvolver um website possam utilizar este conhecimento para desenvolver aplicativos para as plataformas mobile, como iOS ou Android. As próximas seções farão referência à elementos do Ionic e a classes criadas durante a implementação do aplicativo. O diagrama de classes do aplicativo está disponibilizado no APÊNDICE A. O código-fonte do aplicativo está disponibilizado em Weege Junior (2018a).

### 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. O APÊNDICE B demonstra a organização da estrutura em questão.

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. Já para iOS o diretório deve ser criado 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 de fontes de composição, 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 5 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.

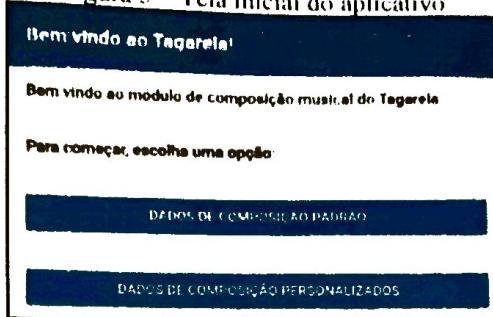
O processo de carga de dados consiste em carregar os dados da fonte de composição para a memória do aplicativo, bem como definir os valores de configuração padrão para a composição. Também é nesta etapa que os dados de configuração anteriormente salvos são recarregados, caso existam. No processo de carga 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

Citar o diagrama de classes

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 e persistência de parâmetros.

Figura 5 - Tela inicial do aplicativo



Fonte: elaborado pelo autor.

Série 0  
Paulo Góspel

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`.   
Há um atributo (ou) associação entre `MusicalCompositionSourceControl` e `MidiFileControl`

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 divisão 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.

O que são estes tipos?

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. Os eventos considerados pelo aplicativo foram obtidos através da documentação de especificação do protocolo MIDI (THE MIDI MANUFACTURERS ASSOCIATION, 2014).

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 track.   |

Fonte: elaborado pelo autor.

Para cada track processada a função `setupMidiFromFile` 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 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 JavaScript Object Notation (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 em tempo de execução, 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`. [Caso o arquivo de configuração não seja encontrado nenhuma alteração é realizada nos objetos de configuração.] → Confusa essa frase. Rever.

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 D.

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 do exemplo 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 opção.

1 O que é executado?

ou é  
uma  
associação?

Qual será o efeito se não multiplicasse?

### 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 E 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, que determinam valores para toda a fonte de 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 parâmetros gerenciados por este grupo e os componentes de tela relacionados a eles são:

- Estão na figura 12.
- 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 que pode ser aplicado a uma composição;
  - b) valor máximo de andamento: determina o valor máximo que o usuário pode escolher para a propriedade de andamento 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 andamento que pode ser aplicado a 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 da composição. 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 pela fonte de fragmentos musicais 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 e os componentes de tela relacionados a eles, 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 e os componentes de tela relacionados a eles, para cada opção da composição, são:

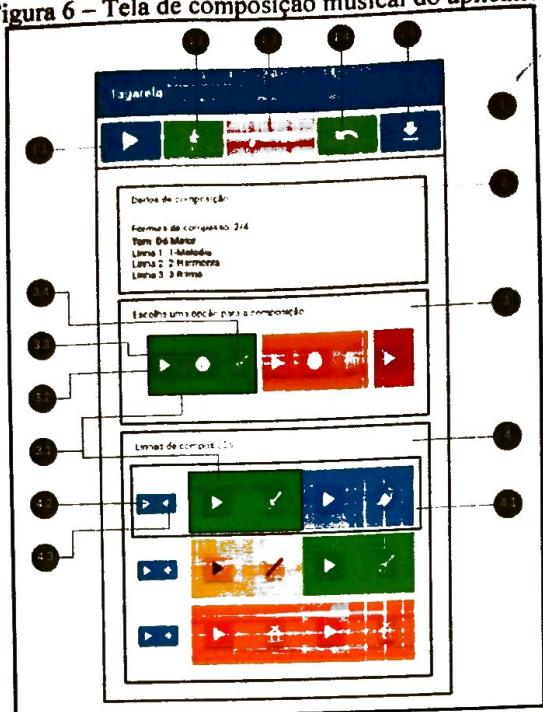
- a) instrumentos para escolha: determina quais instrumentos musicais podem ser selecionados pelo usuário na tela de composição. A entrada do usuário é realizada através de um componente select do Ionic configurado para aceitar seleção de múltiplos valores. Este parâmetro é uma lista de instrumentos 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 musical personalizada e editar as propriedades sonoras dos elementos em questão. A Figura 6 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 6 – 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 pelo 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 pelo item 2. A terceira parte, representada pelo item 3, é o card onde se apresentam as opções. A quarta parte, representada pelo item 4, 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 itens serão apresentados na sequência: 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 D.

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.

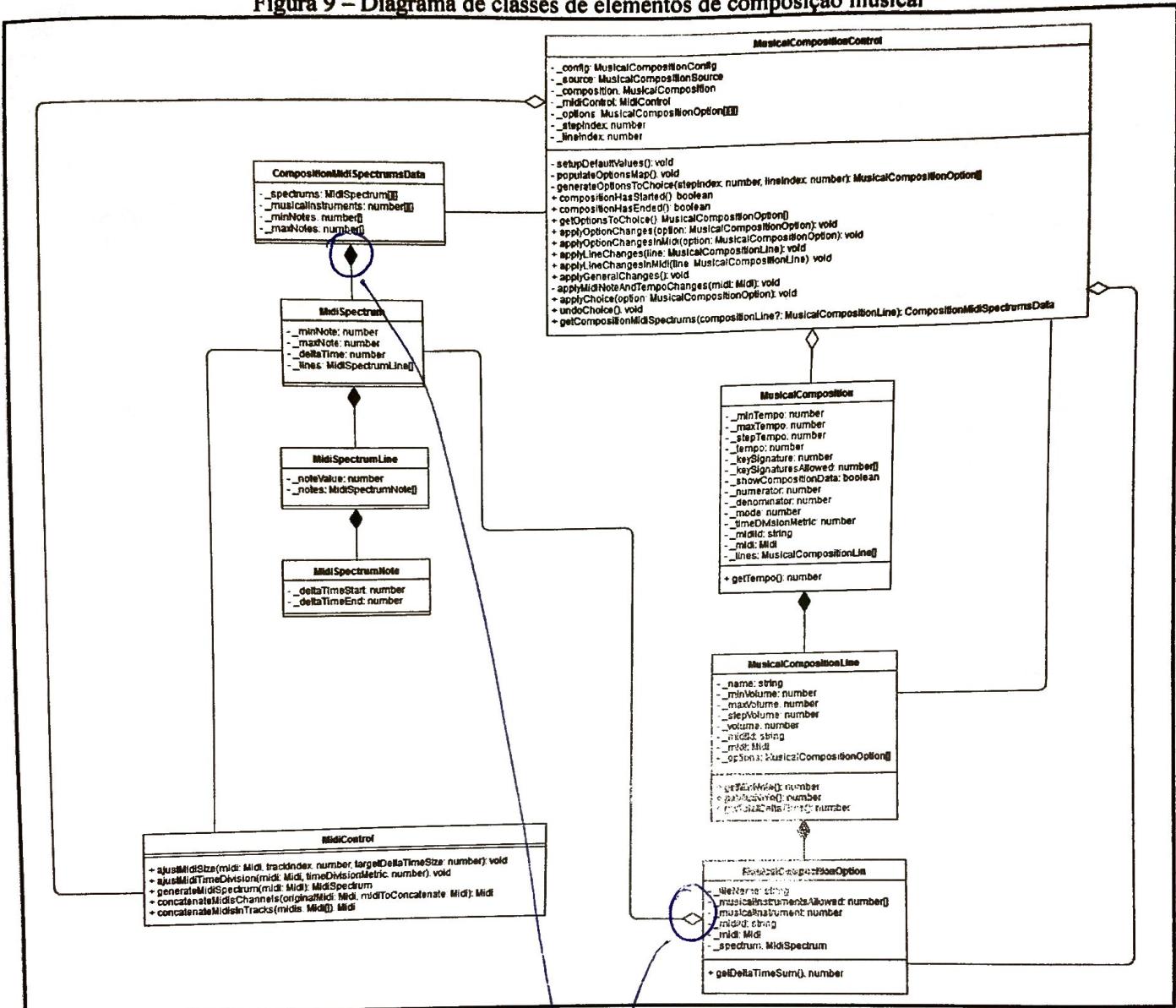
As alterações de propriedades sonoras são realizadas por métodos dos objetos `Midi`. Os parâmetros utilizados para estas alterações são atributos da classe `MusicalComposition` ou de seus atributos. Quando uma alteração desta natureza é realizada, o objeto `Midi` é o responsável por realizar todas as alterações em eventos MIDI de suas tracks.

Os números estão pouco legíveis.  
Usar letra preta com fundo branco.

Citar o Apêndice A  
figura 7.

Não é possível  
puxar isso  
na figura 6.

Figura 9 – Diagrama de classes de elementos de composição musical



Fonte: elaborado pelo autor.

MidiSpectrum não pode ser uma composição de CompositionMidiSpectrumData se a classe é uma agregação de MusicalCompositionOption