INSTRUÇÕES PARA O ARTIGO DO TRABALHO DE CONCLUSÃO DE CURSO – BCC – SIS

João Pedro da Silva, José Carlos Pereira – 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

O presente *template* deve ser seguido para a confecção dos trabalhos finais de conclusão dos cursos de Ciência da Computação e Sistemas de Informação da FURB. Ele é fortemente (mas não totalmente) baseado no modelo da Sociedade Brasileira de Computação de modo a facilitar posteriores adaptações para publicações das produções dos TCCs em eventos científicos. O *template* proposto segue as normas da ABNT para citações e referências bibliográficas, bem como para referências à figuras, quadros e tabelas.

Para o TCC dos cursos, **o presente artigo está limitado a 20 páginas**, incluindo as referências bibliográficas e excluindo os anexos, alguns dos quais são obrigatórios. Os principais elementos de formatação estão explicados no anexo no fim deste *template*. O formato pode ser diferente caso o artigo seja submetido a um evento científico da área. Neste caso o estudante pode manter o artigo no formato do evento, mas deve anexar o comprovante de submissão. Dúvidas ou problemas devem ser sanados com a coordenação do TCC do curso.

A introdução deve despertar no leitor o interesse pelo texto, apresentando os assuntos que serão tratados e o enfoque que será dado ao tema central. Deve iniciar com uma **contextualização** do estudo a ser realizado, explicando claramente sua origem/motivação. A visão geral do tema deve então ser afunilada até se chegar ao problema a ser pesquisado. Após o problema ter sido identificado, deve-se delimitar que aspectos ou elementos serão tratados. Deve-se deixar bem claro o problema que se quer resolver, com o desenvolvimento do trabalho e **justificar** porque o assunto merece ser estudado.

Deve esclarecer a **formulação do problema** a ser investigado e deve ser finalizado com os **objetivos do trabalho** que podem ser subdivididos **em geral (obrigatório)** e **específicos (opcional)**. O objetivo principal deve ser descrito em uma frase única, usando o verbo no infinitivo. Os objetivos específicos detalham o objetivo principal ou definem subprodutos do trabalho. Também se relacionam a formas de validação ou avaliação do trabalho realizado. Os objetivos devem ser mensuráveis quanto a se e como foram ou não atingidos. Os objetivos específicos devem ser enumerados, usando verbos no infinitivo.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica pode ser subdividida em subseções de acordo com o que for mais conveniente. A seguir são propostas formas de organização sendo que a de correlatos é obrigatória. **Recomenda-se fortemente que esta seção não ultrapasse quatro páginas**.

2.1 CONCEITOS, TÉCNICAS E/OU FERRAMENTAS

Na primeira parte deve-se abordar os conceitos, técnicas e/ou ferramentas mais relevantes envolvidos com o tema, devendo ser omitidas metodologias de especificação e ferramentas de implementação que já são conhecidas.

2.2 VERSÃO ANTERIOR DO SISTEMA/FERRAMENTA/BIBLIOTECA/FRAMEWORK (OPCIONAL)

Quando o projeto propõe uma continuação ou extensão de um Trabalho de Conclusão de Curso (TCC) anterior, deve-se descrevê-lo em uma seção específica.

2.3 TRABALHOS CORRELATOS

Na segunda parte da fundamentação devem ser descritos os trabalhos correlatos. Devem ser incluídos preferencialmente trabalhos acadêmicos com características e funcionalidades semelhantes ao que está sendo produzido.

A descrição deve ser feita em quadros conforme apresentado no Quadro 1. Os itens apresentados no quadro são obrigatórios.

Quadro 1 – Trabalho Correlato 1

Referência	
Objetivos	
Principais funcionalidades	
Ferramentas de desenvolvimento	
Resultados e conclusões	

Fonte: elaborado pelo autor.

Os quadros devem ser finalizados com uma breve análise da relação com o trabalho proposto.

3 DESCRIÇÃO DO APLICATIVO

Este capítulo pretende apresentar os detalhes de especificação e implementação do aplicativo. 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.

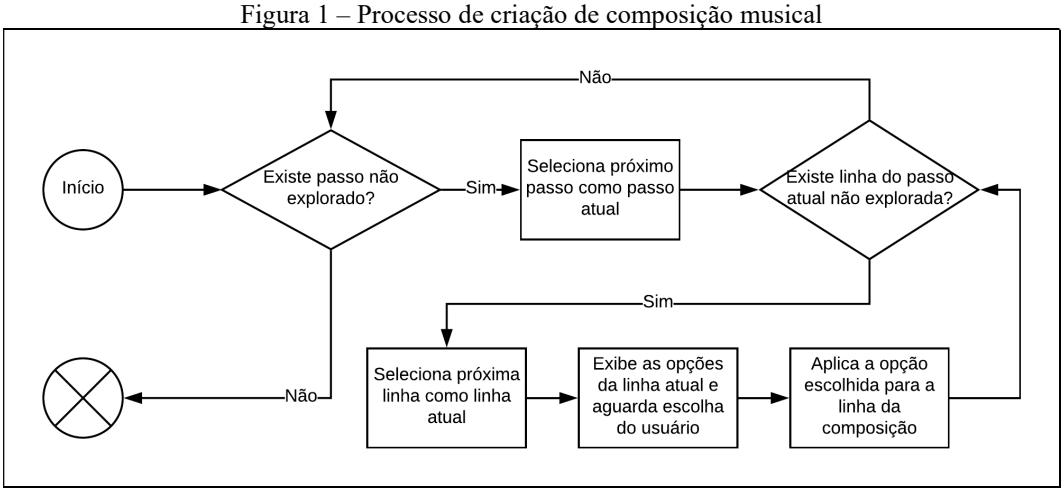
3.1 VISÃO GERAL DO APLICATIVO

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:

a) `opção`: é o elemento que representa um fragmento musical;
b) `linha`: é o elemento que agrupa as `opções`;
c) `passo`: é o elemento que agrupa `linhas`;
d) `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 1 exibe o fluxograma que representa o processo de criação de composição musical.

Figura 1 – 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 2 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 2 – Relação entre propriedades sonoras editáveis e propriedades da composição

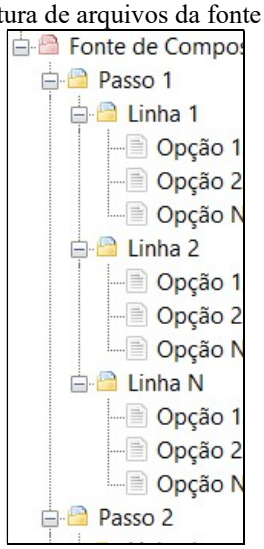
Propriedade sonora	Elemento de composição alterado	Funcionamento
Timbre	`opção`	O timbre é alterado quando a propriedade de instrumento musical da `opção` é alterada.
Volume	`linha`	O volume é alterado quando a propriedade de volume da `linha` é alterada.
Duração	`fonte de composição`	A duração é alterada quando a propriedade de velocidade da composição é alterada.
Frequência	`fonte de composição`	A frequência 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 2 demonstra a organização da estrutura em questão.

Figura 2 – Estrutura de arquivos da fonte de composição



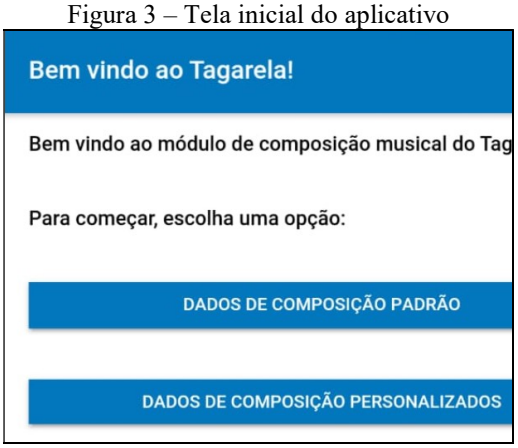
Fonte: elaborado pelo autor.

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 3 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 3 – 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 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 3 apresenta os eventos MIDI que são importados pelo aplicativo e o motivo pelo qual eles são considerados.

Quadro 3 – 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 frequência.
`Note Off`	Evento que representa o fim da execução de uma nota musical. Este evento possui associação com propriedades sonoras de duração e frequência.
`Key Signature`	Evento que representa a tonalidade. Este evento possui associação com a propriedade sonora de frequência.
`Time Signature`	Evento que representa as propriedades de formula de compasso. Este evento possui associação com a propriedade sonora de velocidade.
`End Of Track`	Evento que representa o fim de uma `track`.

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 Quadro 4 demonstra essas validações. Se qualquer uma destas validações falharem o processamento é encerrado e uma mensagem de erro, específica para cada situação, é exibida ao usuário.

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

Validação	Descrição
Tamanho de `delta time`	O tamanho da indicação de `delta time` 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 `MidiFileControl`.
Mais de um `channel` por `track`	O aplicativo não permite que uma `track` possua notas sendo executadas em mais de um `channel`.
Mais de um evento `Time Signature` com valores diferentes entre eles	O aplicativo não suporta alterações de `Time Signature` em um objeto `Midi`. 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 `Time Signature` 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 `Time Signature` de todos os objetos `Midi` de uma fonte de composição possuam os mesmos valores.
Mais de um evento `Key Signature` com valores diferentes entre eles	O aplicativo não suporta alterações de `Key Signature` em um objeto `Midi`. 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 `Time Signature` com atributo `mode` 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 `Key Signature` de todos os objetos `Midi` de uma fonte de composição possuam os mesmos valores para o atributo `mode`.
Mais de um evento `End Of Track`	Cada `track` deve possuir somente um evento de finalização.
Evento `End Of Track` antes do fim da `track`	O evento de finalização de `track` deve ocorrer somente no fim da `track`.
Ausência de evento `End Of Track`	O aplicativo exige que um evento `End Of Track` seja encontrado no arquivo MIDI.
Ausência de evento `Time Signature`	O aplicativo exige que um evento `Time Signature` seja encontrado no arquivo MIDI.
Ausência de evento `Key Signature`	O aplicativo exige que um evento `Key Signature` seja encontrado no arquivo MIDI.
Ausência de evento `Set Tempo`	O aplicativo exige que um evento `Set Tempo` seja encontrado no arquivo

	MIDI.

Fonte: elaborado pelo autor.

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 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 XXX.

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`.

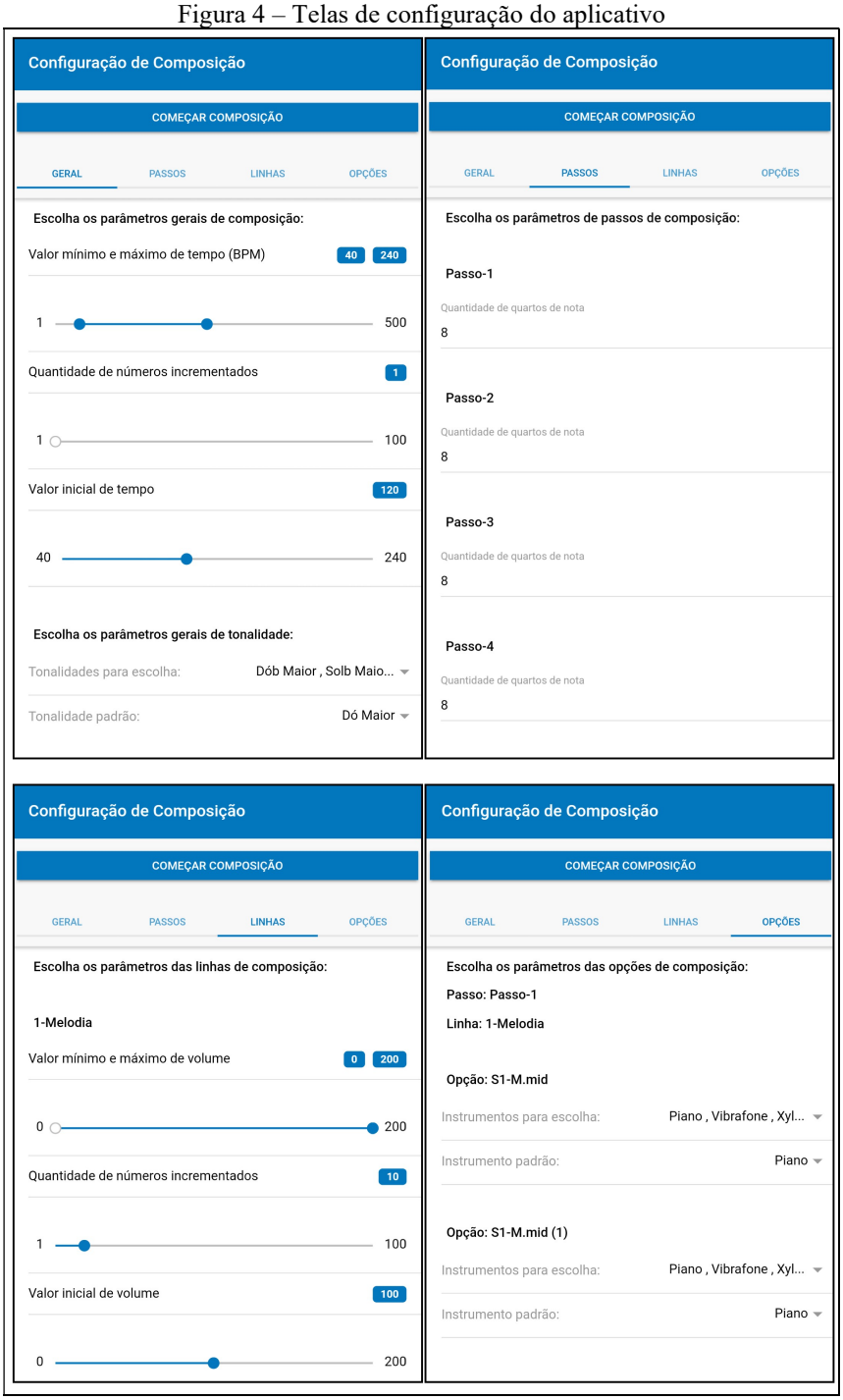
A sexta e última etapa do processo de carga de dados é a execução do método `normalizeTimeDivision` da classe `MusicalCompositionConfigControl`. Este método percorre todos os objetos `Midi` da `fonte de composição` para determinar o maior valor do atributo `timeDivision.metric` desses objetos. Após o 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 `timeDivision.metric`, 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 `deltaTime` 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 x 100 = 200. Portanto, para converter o valor de `timeDivision.metric` do objeto `Midi` para 200 é necessário multiplicar os valores de `deltaTime` 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. A Figura 4 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`.

Figura 4 – Telas de configuração do aplicativo



Fonte: elaborado pelo autor.

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 tempo da composição. O segundo componente é o componente personalizado `list-`

popover 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 tempo: determina o valor mínimo que o usuário pode escolher para a propriedade de tempo 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 tempo em uma composição;
b) valor máximo de tempo: 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 tempo. 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 tempo: determina o valor que o parâmetro de tempo 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 list-popover, 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. Portanto, o nome do parâmetro é quantidade de quartos de nota. Este parâmetro será utilizado na configuração da composição musical 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 cara cada `opção`. Os componentes são `buttons` do Ionic, responsáveis por determinar a propriedade de timbre da opção. Os parâmetros gerenciados por este grupo de parâmetros, para cada opção da composição, são:

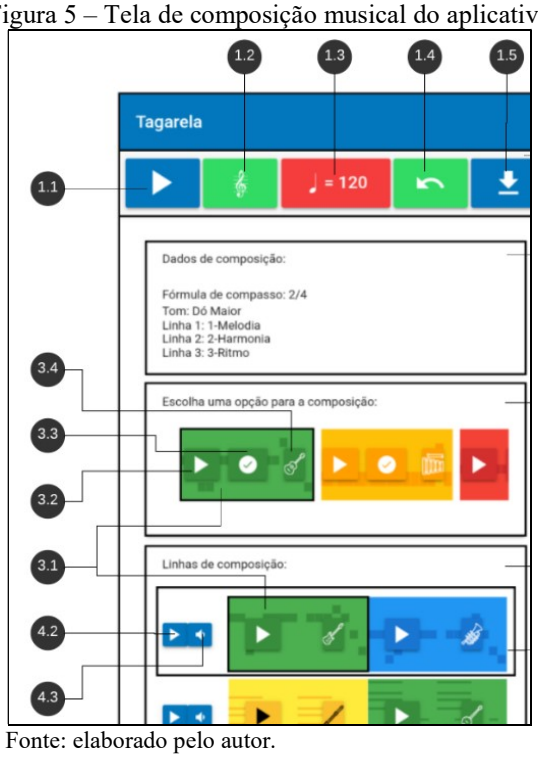
a) instrumentos para escolha: determina quais instrumentos musicais podem ser selecionados pelo usuário na tela de composição. A entrada do usuário é realizada através de um componente `select` do Ionic configurado para aceitar seleção de múltiplos valores. Este parâmetro é uma lista de instrumentos que é utilizada como parâmetro para o componente `list-popover`, 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. Além disso o atributo de `espectro` de cada `opção` é criado. 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 1 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 5 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 5 – 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, é exibida em forma de um retângulo. Este objeto é controlado pelo objeto ChoiceComponent. A imagem exibida no fundo da representação de opção 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 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 background. O comprimento do retângulo é relativo à duração da nota musical na composição. As cores de background 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 XXX.

Cada representação de opção contem 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 frequência. 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 sua track. Com isto, as classes de controle dos componentes gráficos não necessitam tomar conhecimento sobre os detalhes de estrutura da classe Midi. O Quadro 5 apresenta uma descrição das funções de alteração de propriedades sonoras de objetos Midi que são utilizadas no processo de composição.

Quadro 5 – Descrição das funções de alteração de propriedades musicais

Propriedade sonora	Método da classe Midi	Funcionamento
Timbre	applyInstrumentChange	Recebe como parâmetro o número do instrumento musical. Para cada track do objeto Midi, elimina os eventos de definição de instrumento e adiciona, como o primeiro evento da track, um novo evento de definição de instrumento musical de acordo com o parâmetro.
Volume	applyVolumeChange	Recebe como parâmetro o valor percentual que deve ser aplicado ao volume. Para cada track do objeto Midi, para cada evento de inicio 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	applyTempoChange	Recebe como parâmetro o valor de tempo. Para cada track do objeto Midi, edita o evento que determina a velocidade de execução de acordo com o parâmetro.
Frequência	applyNoteTranspose	Recebe como parâmetro o número da armadura de clave. Para cada track do objeto Midi, edita o evento que determina a armadura de execução 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 elemento 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. 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 `getIonIconToMidiNumber` 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 XXX.

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 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 volume, timbre, velocidade 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 inicio ou finalização de notas musicais, determinação de instrumento musical ou finalização de track 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 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 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 Midis e retorna um novo objeto Midi. Este novo objeto é um objeto Midi que contem 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 velocidade 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 atributa de velocidade da composição. O item 1.3 exibe a figura de nota que corresponde ao quarto de nota e a velocidade atual da composição.

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 FilePrivider 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 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 à 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) modo: 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.

3.5 TEXTO DE ORIENTAÇÃO

Nesta seção devem ser descritos os **aspectos mais relevantes de especificação e implementação** para a compreensão sobre o trabalho desenvolvido. O título "DESCRIÇÃO" pode ser complementado com "DO SOFTWARE", "DA FERRAMENTA" ou "DO PROTÓTIPO" ou aquilo que melhor representar o que foi desenvolvido. A organização desta seção é livre e deve ser seguida uma metodologia própria para cada tipo de trabalho. Reitera-se que, em função da limitação do número de páginas, a descrição deve contemplar o que é mais significativo para a compreensão do que foi desenvolvido.

Destaca-se que os diagramas desenvolvidos bem como outros aspectos de especificação deverão obrigatoriamente constar nos anexos. Opcionalmente poderá constar os códigos mais relevantes da implementação, bem como aa telas do trabalho desenvolvido.

4 RESULTADOS

De modo a ampliar o seu caráter científico, todos os TCCs devem apresentar e discutir resultados não limitados à comparação com os trabalhos correlatos. Devem ser apresentados os casos de testes do software, destacando objetivo do teste, como foi realizada a coleta de dados e a apresentação dos resultados obtidos, preferencialmente em forma de gráficos ou tabelas, fazendo comentários sobre os mesmos. Também é sugerida a comparação com os trabalhos correlatos apresentados na fundamentação teórica.

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

As referências devem ser apresentadas em ordem alfabética. Só podem ser inseridas nas referências os documentos citados ao longo da monografia. Todos os documentos citados obrigatoriamente têm que estar inseridos nas referências. A seguir são apresentados alguns exemplos de referências bibliográficas. Destaca-se que deve ser seguida a norma da ABNT.

[parte de um documento:]

AMADO, Gilles. Coesão organizacional e ilusão coletiva. In: MOTTA, Fernando C. P.; FREITAS, Maria E. (Org.). **Vida psíquica e organização**. Rio de Janeiro: FGV, 2000. p. 103-115.

[trabalho acadêmico ou monografia (TCC/Estágio, especialização, dissertação, tese):]

AMBONI, Narcisa F. **Estratégias organizacionais**: um estudo de multicasos em sistemas universitários federais das capitais da região sul do país. 1995. 143 f. Dissertação (Mestrado em Administração) - Curso de Pós-Graduação em Administração, Universidade Federal de Santa Catarina, Florianópolis.

[norma técnica:]

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: informação e documentação: referências - elaboração. Rio de Janeiro, 2002a. 24 p.

_____. **NBR 10520**: informação e documentação: citações em documentos: apresentação. Rio de Janeiro, 2002b. 7 p.

_____. **NBR 14724**: informação e documentação: trabalhos acadêmicos: apresentação. Rio de Janeiro, 2011. 11 p.

[livro:]

BASTOS, Lília R.; PAIXÃO, Lyra; FERNANDES, Lúcia M. **Manual para a elaboração de projetos e relatórios de pesquisa, teses e dissertações**. Rio de Janeiro: Zahar, 1979.

[trabalho acadêmico ou monografia (TCC/Estágio, especialização, dissertação, tese):]

BRUXEL, Jorge L. **Definição de um interpretador para a linguagem Portugol, utilizando gramática de atributos**. 1996. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

[verbete de enciclopédia em meio eletrônico:]

EDITORES gráficos. In: WIKIPEDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2006. Disponível em: <http://pt.wikipedia.org/wiki/Editores_graficos>. Acesso em: 13 maio 2006.

[artigo em evento:]

FRALEIGH, Arnold. The Algerian of independence. In: ANNUAL MEETING OF THE AMERICAN SOCIETY OF INTERNATIONAL LAW, 61, 1967, Washington. **Proceedings…** Washington: Society of International Law, 1967. p. 6-12.

[norma técnica:]

IBGE. **Normas para apresentação tabular**. 3. ed. Rio de Janeiro, 1993. 61 p. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/monografias/GEBIS%20-%20RJ/normastabular.pdf>. Acesso em: 27 ago. 2013.

[artigo em periódico:]

KNUTH, Donald E. Semantic of context-free languages. **Mathematical Systems Theory**, New York, v. 2, n. 2, p. 33-50, jan./mar. 1968.

[trabalho acadêmico ou monografia (TCC/Estágio, especialização, dissertação, tese):]

SCHUBERT, Lucas A. **Aplicativo para controle de ferrovia utilizando processamento em tempo real e redes de Petri**. 2003. 76 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

[página da internet com autor]

SCHULER, João P. S. **Tutorial de Delphi**. Porto Alegre, [2002]. Disponível em: <http://www.schulers.com/jpss/pascal/dtut/>. Acesso em: 27 ago. 2013.

[página da internet sem autor]

SCHRATCH. **Program, imagine, share**. [S.l.], [2013?]. Disponível em: <https://scratch.mit.edu/>. Acesso em: 27 maio 2013.

[relatório de pesquisa:]

VARGAS, Douglas N. **Editor dirigido por sintaxe**. 1992. Relatório de pesquisa n. 240 arquivado na Pró-Reitoria de Pesquisa, Universidade Regional de Blumenau, Blumenau.

APÊNDICE A – DIAGRAMAS DE ESPECIFICAÇÃO

Este apêndice é obrigatório. Nele devem constar todos os diagramas de especificação desenvolvidos no trabalho. Os diagramas devem conter legendas numeradas na sequência do artigo.

APÊNDICE B – NOME

Podem ser inseridos outros apêndices no artigo tais como códigos de implementação, telas de interface, instrumentos de coleta de dados, entre outros. **Apêndices são textos elaborados pelo autor** a fim de complementar sua argumentação. Os apêndices são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deve haver no mínimo uma referência no texto anterior para cada apêndice. Colocar sempre um preâmbulo no apêndice. Caso existam tabelas ou ilustrações, identifique-as através da legenda, seguindo a numeração normal das legendas do artigo.

ANEXO A – DESCRIÇÃO

Elemento opcional, **anexos são documentos não elaborados pelo autor**, que servem de fundamentação, comprovação ou ilustração, como mapas, leis, estatutos, entre outros. Os anexos são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deve haver no mínimo uma referência no texto anterior para cada anexo. Colocar sempre um preâmbulo no anexo. Caso existam tabelas ou ilustrações, identifique-as através da legenda, seguindo a numeração normal das legendas do artigo.

6 DESCRIÇÃO DA FORMATAÇÃO

A seguir são apresentadas observações gerais sobre o texto do artigo do Trabalho de Conclusão de Curso (TCC). **Observa-se que esta descrição deve ser retirada do texto final.**

Na confecção do texto deve-se:

a) usar frases curtas. Segundo Teodorowitsch (2003, p. 3), “Frases com mais de duas linhas aumentam o risco de o leitor não compreender a ideia ou de entendê-la de forma equivocada.”;
b) usar linguagem impessoal (usar a terceira pessoa do singular) e verbo na voz ativa (a ação é praticada pelo sujeito), com conexão entre os parágrafos;
c) não usar palavras coloquiais;
d) não usar palavras repetidas em demasia;
e) usar verbos no presente quando for referir-se a partes do trabalho que já se encontram disponíveis no texto;
f) destacar palavras em língua estrangeira em itálico, conforme descrito abaixo:
   - nome de software, ferramenta, aplicativo, linguagem de programação, plataforma, empresa: não deve ser escrito em itálico (exemplos: Delphi 7, Pascal, Object Pascal, Java, JavaScript, Java 2 Micro Edition, Basic, Microsoft Visual C++, C, Windows, Linux, MySQL, Oracle, Eclipse 3.0, Enterprise Architect, Rational Rose, Microsoft, Sun Microsystems),
   - citações: o sobrenome do autor ou o nome da instituição responsável pela autoria do documento citado não deve ser escrito em itálico (exemplo: Segundo Sun Microsystems (2004), ...),
   - palavras em língua estrangeira encontradas nos dicionários nacionais: não devem ser grafadas em itálico (exemplos: software, hardware, web, Internet),
   - demais palavras em língua estrangeira: devem ser escritas em itálico (exemplos: *palmtop*, *classpath*, *play*, etc.). No entanto, Teodorowitsch (2003, p. 7), sugere que alguns termos em língua inglesa devem ser substituídos por termos em português (exemplos: núcleo em vez de *kernel*, aprendizagem de máquina em vez de *machine learning*, etc.);
g) observar as seguintes regras quanto ao uso de siglas:
   - colocar as siglas entre parênteses precedidas pela forma completa do nome, quando aparecem pela primeira vez no texto (exemplos: Associação Brasileira de Normas Técnicas (ABNT), Trabalho de Conclusão de Curso (TCC)). Caso exista uma lista de siglas na parte pré-textual do volume final, pode-se usar somente a sigla, quando aparecer pela primeira vez no texto,
   - usar apenas a sigla nas demais ocorrências no texto,
   - escrever as siglas em letras maiúsculas e não usar itálico,
   - escrever o plural das siglas sem apóstrofo (exemplos: PCs, APIs, PDAs) e determinar o gênero da sigla conforme o gênero do primeiro substantivo do seu nome (exemplo: o TCC – o Trabalho de Conclusão de Curso).

6.1 FORMATAÇÃO

A formatação geral para apresentação do documento, descrita na NBR 14724 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2011), é a seguinte:

a) o texto divide-se em capítulos, seções e subseções (até cinco divisões);
b) a apresentação de citações em documentos deve seguir a NBR 10520 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b);
c) a descrição das referências bibliográficas deve estar de acordo com a NBR 6023 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002a).

Observa-se ainda que todo capítulo, seção ou subseção deve ter no mínimo um texto relacionado.

A monografia deve ser digitada usando as fontes e formatação de parágrafos deste modelo, indicadas no Quadro 6.

Quadro 6– Estilos do modelo

USO	**FORMATO**
título de capítulo ou seção primária (1)	TF-TÍTULO 1 (Times New Roman, 10pt, negrito, maiúsculas)
título de seção secundária (1.1)	TF-TÍTULO 2 (Times New Roman, 10pt, maiúsculas)
título de seção terciária (1.1.1)	TF-Título 3 (Times New Roman, 10pt, minúsculas, exceto a 1ª letra da 1ª palavra do título e de nomes próprios)
título de seção quaternária (1.1.1.1)	TF-Título 4 (mesma formatação seção ternária)
título de seção quinária (1.1.1.1.1)	TF-Título 5 (mesma formatação seção ternária)
texto	TF-TEXTO (Times New Roman, 10pt)
citação direta com mais de três linhas	TF-CITAÇÃO (Times New Roman, 9pt, com recuo de 4 cm)
itens (alíneas)	ver descrição abaixo (Times New Roman, 10pt)
referência bibliográfica	TF-REFERÊNCIA ITEM (Times New Roman, 10pt, alinhada à margem esquerda)
fonte, legenda, texto de quadro/tabela e figura	TF-FONTE (Times New Roman, 9pt, alinhada à margem esquerda do quadro/tabela) TF-LEGENDA, (Times New Roman, 10pt, centralizada) TF-TEXTO- QUADRO (Times New Roman, 10pt) TF-FIGURA (Times New Roman, 10pt, centralizada)

Fonte: elaborado pelo autor.

O espaçamento, também definido no modelo, deve ser conforme indicado no Quadro 7.

Quadro 7 - Espaçamento

USO	**ESPAÇAMENTO**
título de capítulo ou seção primária (1) título de seção secundária (1.1) título de seção terciária (1.1.1) título de seção quaternária (1.1.1.1) título da seção quinária (1.1.1.1.1)	espaço simples, com 12pt antes do parágrafo
texto	espaço simples, com 6 pt antes do parágrafo
citação direta com mais de três linhas	espaço simples com 6pt antes e depois do parágrafo
itens (alíneas)	espaço simples, com 6 pt antes do parágrafo
referência bibliográfica	espaço simples, com 6 pt antes do parágrafo
legenda e texto de ilustração/tabela	espaço simples, com 6 pt antes do parágrafo
fonte	espaço simples, com 0pt antes do parágrafo

Fonte: elaborado pelo autor.

Na disposição gráfica de itens (alíneas) devem ser observados os seguintes quesitos:

a) o texto que antecede os itens termina com dois pontos;
b) cada item deve iniciar com uma letra minúscula seguida de fecha parênteses e terminar com um ponto e vírgula, sendo que o último item termina com ponto (FORMATO: TF-ALÍNEA);
c) o texto de cada item inicia com letra minúscula, exceto nomes próprios;
d) quando contiver subitens, os mesmos devem iniciar com hífen colocado sob a primeira letra do texto do item correspondente (FORMATO: TF-SUBALÍNEA nível 1 ou TF-SUBALÍNEA nível 2, conforme o caso). Nesse caso, cada subitem deve terminar com uma vírgula, exceto o último que termina com ponto ou com ponto e vírgula.

Segue um exemplo:

a) cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA);
b) cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA):
   - cada subitem (nível 1) inicia com letra minúscula, cada subitem (nível 1) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 1):
     - cada subitem (nível 2) inicia com letra minúscula, cada subitem (nível 2) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 2),
     - cada subitem (nível 2) inicia com letra minúscula, cada subitem (nível 2) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 2),
   - cada subitem (nível 1) inicia com letra minúscula, cada subitem (nível 1) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 1);
c) cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA).

6.1.1.1 Exemplo de título de seção quaternária [FORMATO: TF-TÍTULO 4]

Formato: TF-TEXTO.

6.1.1.1.1 Exemplo de título de seção quinária [FORMATO: TF-TÍTULO 5]

Formato: TF-TEXTO.

6.1.2 Formatação de quadros, figuras e tabelas

Um quadro contém apenas informações textuais, que podem ser agrupadas em colunas. Uma figura contém, além das informações textuais, pelo menos um elemento gráfico. Uma tabela é uma apresentação tabular de informações **numéricas** relacionadas.

Os quadros, figuras e tabelas são identificados na parte superior por uma legenda (a qual deve estar centralizada) composta pela palavra designativa (Figura, Quadro ou Tabela, conforme o caso), seguida de seu número em algarismo arábico (usar numeração progressiva, uma sequência para os quadros, outra para as figuras e outra para as tabelas), de hífen e do título. As ilustrações devem:

a) aparecer centralizadas no texto;
b) estar delimitadas por uma moldura simples;
c) aparecer numa única página (quando o tamanho não exceder o da página), inclusive a legenda;
d) serem inseridas o mais próximo possível do trecho a que se referem pela primeira vez.

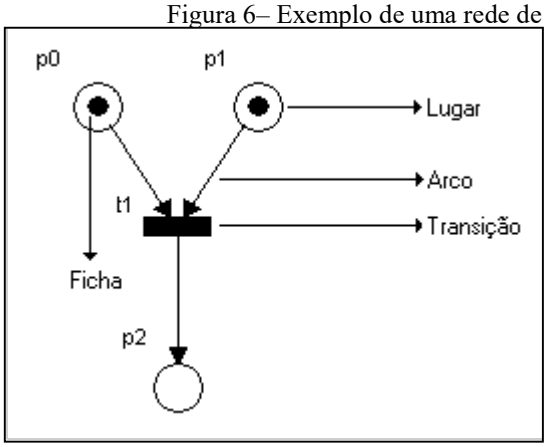
Toda ilustração deve ter fonte, alinhada à margem esquerda da ilustração. Quando foi o próprio autor que fez a ilustração, deve inserir o texto: “Fonte: elaborado pelo autor”.

Observa-se que quando um código fonte for descrito dentro de um quadro, deve-se utilizar letra do tipo `courier new 9pt`. (TF-CÓDIGO-FONTE)

Exemplos de como se deve referenciar uma figura e um quadro e como descrevê-los são mostrados a seguir.]

Um exemplo de uma rede de Petri pode ser visto na Figura 6.

Figura 6– Exemplo de uma rede de Petri



Fonte: Schubert (2003, p. 18).

Um exemplo de código fonte gerado a partir de uma especificação pode ser visto no Quadro 8.

Quadro 8 – Funções que verificam se as transições estão sensibilizadas

```
Function TEstruturaMalha.T1Sensibilizada: boolean;
begin
 result := (Fp2 and Fp4);
end;

function TEstruturaMalha.T2Sensibilizada: boolean;
begin
 result := (Fp1 and Fp3);
end;

function TEstruturaMalha.T3Sensibilizada: boolean;
begin
 result := (Fp2 and Fp4);
end;
```

Fonte: Schubert (2003, p. 63).

Não devem ser usadas bordas (linhas) verticais para fechar as tabelas. Exemplo de como se deve referenciar uma tabela e como descrevê-la é mostrado a seguir.

A quantidade de trabalhos finais realizados no Curso de Ciência da Computação (de 2010 até 2014) é apresentada na Tabela 1.

Tabela 1 – Trabalhos finais realizados no Curso de Ciência da Computação

Ano	Estágios	TCC´s	Totais
2010/1	0	16	16
2010/2	0	21	21
2011/1	0	25	25
2011/2	0	23	23
2012/1	0	23	23
2012/2	0	22	22
2013/1	0	25	25
2013/2	0	16	16
2014/1	0	18	18
2014/2	0	13	13
	0	**202**	**202**

Fonte: elaborado pelo autor.

6.1.3 Exemplos de citações retiradas de documentos ou de nomes constituintes de uma entidade

A apresentação de citações em documentos deve seguir a NBR 10520 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b). O sistema a ser usado é o alfabético. Exemplos de citações são: “Numa publicação recente (SEBESTA, 2000) é exposto ...” e “Segundo Silva et al. (1987), execução controlada de programas é ...”.

Quando a citação referir-se a uma parte específica do documento consultado, especificar no texto da monografia a(s) página(s). Esta(s) deverá(ão) seguir a data, separada(s) por vírgula(s) e precedida(s) pelo designativo que a(s) caracteriza(m). Como exemplo, mostra-se: “(SCHIMT, 1999, p. 50)” ou “... visto que Schimt (1999, p. 50) implementou ...”.

As citações diretas (transcrição textual de parte da obra do autor consultado), no texto, com mais de três linhas, devem ser destacadas com recuo de 4 cm da margem esquerda, com letra menor que a do texto utilizado e sem as aspas (FORMATO: TF-CITAÇÃO), conforme o exemplo a seguir.

> A Associação Brasileira de Normas Técnicas (ABNT) é o Fórum Nacional de Normalização. As Normas Brasileiras, cujo conteúdo é de responsabilidade dos Comitês Brasileiros (ABNT/CB) e dos Organismos de Normalização Setorial (ABNT/ONS), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidades, laboratórios e outros). (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b, p. 1).

Quando da citação de um nome (identificador) constituinte de uma entidade em um texto, deve-se utilizar o tipo de letra *courier new*, com tamanho dez (10). Para facilitar a formatação, existe o estilo de palavra denominado TF-COURIER10. Como exemplo cita-se nome de classe, atributo ou método. A seguir é apresentado um exemplo.

As classes `TTabelaTransicao` e `TExpressaoRegular` são classes de interface, porém estão sendo consideradas como classes de domínio da aplicação.