Player MR:
Reprodutor de Áudio com Legendas

Marcos Momm[#1], Rodrigo de Moraes[#2]

[#]Curso de Bacharelado em Ciência da Computação – Instituto Federal Catarinense (IFC)
89160-240 – Rio do Sul – SC, Brasil

[1]marcos.momm@hotmail.com

[2]moraesrodrigo6@gmail.com

***Resumo* — O desenvolvimento do player de música MR demonstra a necessidade da utilização da programação concorrente, em que a linguagem de programação Java tem o suporte para a programação concorrente. A utilização do JavaFX para a criação de uma interface gráfica para o usuário e o grande diferencial do player de música que são as legendas da músicas no padrão Json. Enquanto a música vai sendo executada vai aparecendo a letra da música ao mesmo tempo em que a música vai sendo executada, para acontecer tudo isso existe a necessidade da programação concorrente.**

***Palavras-chave* — Player de música MR, Java, Programação concorrente, JavaFX, Json.**

I. INTRODUÇÃO

O desenvolvimento de um player de música com a possibilidade de ter a legenda das músicas demonstra a necessidade do uso de *threads* na programação do player, na qual, existe a necessidade de processos no controle do tempo da música, a execução do arquivo de música e na apresentação da legenda da música para o usuário, vários processos ocorrendo ao mesmo tempo.

Nos sistemas operacionais tradicionais, cada processo possui um espaço de endereçamento e um único fluxo de controle. Entretanto, existem situações em que é preferível, ou mesmo necessária, a execução praticamente paralela de vários fluxos de controle no mesmo espaço de endereçamento, [1]. Esses fluxos de controle são conhecidos como *threads*. Em um programa *multithread* há várias *threads* rodando concorrentemente com um único espaço de memória e cada uma com sua própria sequência de instruções.

O desenvolvimento do players de música MR, onde, MR seria as letras inicias de Marcos e Rodrigo, os autores desse trabalho foi utilizado a linguagem de programação Java com o seu suporte para programação concorrente e de toda a lógica do player de música, JavaFX responsável por toda a parte gráfica e layout e disposição dos botão na interface do player de música e foi utilizado o formato da estrutura do Json, usado nos arquivos de legenda.

II. FUNDAMENTAÇÂO TEÒRICA

Para a criação do player de música MR, foi utilizado as seguintes tecnologias para a realização da tarefa. A linguagem de programação Java que tem a possibilidade da programação concorrente com o uso das APIs e que é responsável por todo o funcionamento do player de música, na qual, cada botão que o usuário clica é a linguagem Java rodando por trás do player para a realização do evento do botão em que o usuário clicou.

Para a parte gráfica foi utilizado o JavaFX, para a criação da interface, botões e de todo o layout do player de música, para os arquivos de legenda foi utilizado a mesma estrutura dos arquivos Json e na parte da persistência em que o usuário pode salvar a sua playlist foi utilizado a mesma estrutura do arquivo Json.

Para a criação das legendas foi utilizada o padrão do Json (*JavaScript Object Notation*). Json é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de ser simples, tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o *parsing* dessas informações. Isto explica o fato de o Json ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados, [2].

A ideia utilizada no Json para representa informações é tremendamente simples, para cada valor representado, atribui-se um nome (ou rótulo) que descreve o seu significado. Seguindo esse padrão é que foi montado os arquivos de legenda, em que os rótulos utilizados nos arquivos de legenda são: sequencia, tempo_inicial, tempo_final e o texto, que é a parte da letra da música cantada entre o tempo inicial e o tempo final.

Na interface gráfica do player de música MR foi utilizado o JavaFX, que é uma família de produtos desenvolvidos na Sun Microsystems. JavaFX é uma plataforma que inclui uma linguagem de script declarativa e de alto desempenho para oferecer e criar interfaces gráficas, [4].

O foco primário do JavaFX é tornar o desenvolvimento de

interface gráfica do usuário fácil e adotar características mais atraentes como efeitos visuais, som e animação. O JavaFX inclui uma *framework* pronto para suportar componentes gráficos e facilmente incluir características de multimídia. Usando a plataforma Java como seu núcleo, o JavaFX funciona continuamente como a plataforma Java e pode facilmente alavancar seu código Java existente. Isso também permite que o JavaFX implemente a capacidade "escreva uma vez, execute em qualquer lugar" oferecido pela plataforma Java, [4].

A linguagem de programação Java disponibiliza a concorrência através da linguagem e das APIs (*Application Programming Interface* - Interface de Programação de Aplicações). Cada *thread* tem sua própria pilha de chamadas de método e seu próprio contador de programa, o que possibilita a execução simultânea com outras *threads* e o compartilhamento de recursos no nível do aplicativo. Diferentemente de linguagens que não possuem capacidades de *multithreading* integradas, as quais devem realizar chamadas não portáveis para primitivos de *multithreading* do sistema operacional, no Java os primitivos de *multithreading* estão incluídos como parte da própria linguagem e de suas bibliotecas. Esta distinção facilita a portabilidade entre plataformas com a manipulação de *threads*, em relação a outras linguagens [5].

A classe *Thread* no Java, já é algo nativo na própria linguagem, podendo ser estendida ou implementada através da classe *Runnable* em qualquer classe criada. E através do método *Run* é onde o programador coloca as linhas de código ou chamada de métodos para que estes executem concorrentemente, conforme figura abaixo.

```
12  public class MinhaThread implements Runnable{
13
14      @Override
15      public void run() {
16
17      }
18  }
```

Fig. 1. Classe que implementa a interface Runnable

III. DESENVOLVIMENTO DO PLAYER DE MÚSICA MR

O desenvolvimento do player se deu com intuito de proporcionar ao usuário, além da disponibilidade de reproduzir arquivos mp3, as opções de cadastrar e salvar playlists de músicas e visualizar legendas de suas músicas.

Para o cadastro das playlists, basta utilizar o canto esquerdo da interface, onde as músicas podem ser selecionadas, salvadas e editadas. Para salvar ou abrir uma playlist, pode ser utilizado o menu "Arquivo" da interface. Essas opções podem ser vistas na imagem da interface, apresentada na figura abaixo.

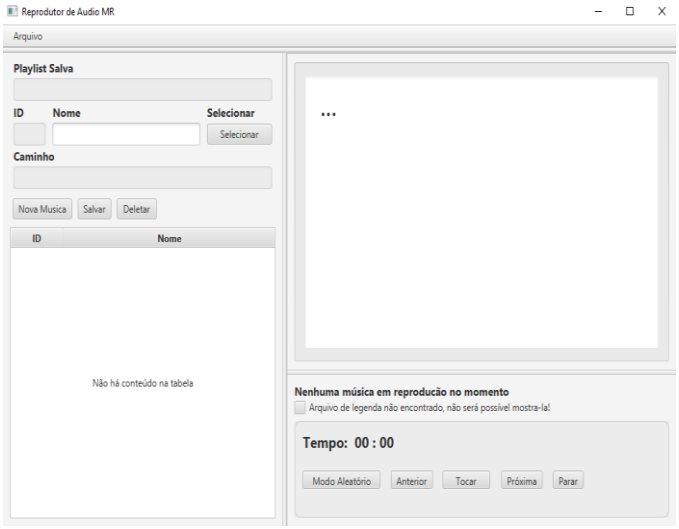

Fig. 2. Interface do Reprodutor de Áudio MR (Acervo do autor)

Ao salvar uma playlist, o programa salva um arquivo com extensão "plt", que também, é a única extensão de arquivo reconhecida na hora de se abrir uma playlist. Dentro deste arquivo, as músicas são salvas na estrutura de Json, seguindo o mesmo modelo da classe "Musica", que pode ser observada no diagrama de classe junto ao Anexo 1. Basicamente, em tempo de execução, uma playlist é manipulada através de uma lista de objetos do tipo "Musica" e a partir do momento do salvamento, esta lista de objetos, que é do tipo "ObservableList", bastante utilizado junto ao JavaFX, é convertida em um Json. O processo inverso ocorre no momento que uma playlist é aberta, ou seja, ao abrir uma playlist, o programa lê o arquivo selecionado em busca de uma Json no mesmo formato de uma lista de objetos do tipo "Musica", e assim realiza a conversão do Json para uma "ObservableList".

As legendas das músicas, funcionam de forma semelhante a arquivos de legendas de vídeos, porém neste caso, no padrão de Json. O padrão de legenda utilizado pelo programa pode ser visto na imagem a seguir.

```
{
    "Arquivo": "PorcaVeia-LuzDoMeuRancho",
    "letra": [
        {
            "sequencia": 1,
            "tempo_inicial": "00:18",
            "tempo_final": "00:21",
            "texto": "Se às vezes me esqueço em função da lida,"
        },
        {
            "sequencia": 2,
            "tempo_inicial": "00:22",
            "tempo_final": "00:26",
            "texto": "de dizer que a vida pra mim é você,"
        }
    ]
}
```

Fig. 3. Estrutura Json do arquivo de legendas. (Acervo do Autor)

Como pôde ser acompanhado, o arquivo de legenda, armazena dois atributos, um deles contendo o nome do arquivo,

e o outro atributo sendo uma lista de frases, que compõem a letra da música. Nesta lista de frases, tem-se um vetor de objetos, onde cada um destes objetos, representam um trecho da música. Neste objeto, além de termos o atributo "texto" temos também, outros 3 atributos. O "tempo_inicial" é referente ao tempo de início que a legenda irá aparecer na tela, e o "tempo_final", será o momento em que a frase desaparecerá da tela. O atributo "sequencia" serve apenas como controle na sequência das frases.

As legendas são salvas em arquivos com extensão "lgd" e deverão ser criadas manualmente pelo usuário que desejar que sua música seja legendada. O nome do arquivo, deverá ter o mesmo nome do arquivo mp3, porém com a diferença que a extensão do arquivo será "lgd". Com isso o programa entenderá automaticamente que aquela música possui legenda e a apresentará na tela.

Referente à programação concorrente, foram utilizadas duas threads principais, uma thread para a reprodução da música no java, utilizando uma biblioteca de terceiros atendendo pelo nome "Player", e a outra thread controlando os componentes da tela, ou seja, a mudança do contador do tempo e das frases de legendas. Ao iniciar ou pausar uma música, as duas threads funcionam em conjunto para que a música e a legenda não saiam de sincronia.

Um pequeno problema que foi encontrado ao utilizar as threads do java junto ao JavaFX foi que encontrou-se necessário a utilização de threads secundárias para a alteração de elementos gráficos na tela. Ou seja, quando se está executando um thread java, para que se possa editar algum componente na tela, é necessário disparar uma thread do tipo "runLater" do JavaFX, para que não haja conflito e que o componente seja editado sem problemas na tela. Segue na próxima imagem, o código referente a função que modifica a legenda da música. Neste método, está sendo criada uma thread (linha 207) com método "Plataform.runLater". E logo após a declaração da thread, já está sendo chamada a execução dela (linha 219).

```
206     public void mudaLegenda() {
207         Thread t = new Thread(() -> {
209             Platform.runLater(new Runnable() {
                    @Override
                    public void run() {
211                     if (mostraLegenda) {
212                         txtLegenda.setText(legenda);
213                     } else {
214                         txtLegenda.setText("...");
215                     }
216                 }
217             });
218         });
219         t.start();
220     }
```

Fig. 4. Função para alteração da legenda na tela. (Acervo do autor)

IV. Conclusão

O desenvolvimento deste trabalho apresentou o desenvolvimento do player de música MR em que tem a possibilidade de o usuário além de ouvir música ter a possibilidade de ter a legenda da música sempre mostrada ao mesmo tempo em que a música vai sendo executada.

Para que todos esses eventos ocorram, execução da música, legendas se mostrou necessário uma linguagem de programação com suporte à programação concorrente, onde, a linguagem de programação Java tem o suporte para a programação concorrente. Na parte da interface foi utilizado o JavaFX, na criação do layout e disposição dos botões do player de música.

Na parte das legendas do player de música, que seria o diferencial, foi utilizado a estrutura Json, armazenando o tempo inicial e o tempo final e o texto que é a parte da letra da música cantado nesse intervalo de tempo.

Referências

[1] A. S. Tanenbaum and A. S. Woodhull, **Sistemas Operacionais, projeto e implementação**, 3 ed. Porto Alegre: Bookmann, 2008.

[2] CÔRREA, Eduardo. **Introdução ao formato JSON.** Disponível em: < http://www.devmedia.com.br/introducao-ao-formato-json/25275 >. Acessado em: 19 mar. 2017 às 18h15min.

[4] CLARKE, Jim; CONNORS, Jim; BRUNO, Eric. **Java FX: desenvolvimento de aplicações de internet ricas**. Rio de Janeiro: Alta Books, 2010. 325 p.

[5] H. M. Deitel and P. J. Deitel, **Java Como Programar**, 8 ed. São Paulo: Pearson Prentice Hall, 2010.

ANEXO I – DIAGRAMA DE CLASSES DO PLAYER MR