Relatório de Desenvolvimento de um MP3 Player em Java com Framework JavaFX

Gabriel Rocha dos Santos

Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN)

gabrielrochasant@gmail.com

Abstract. *This report describes the planning process and the data structures involved in the development of a Media Player in the Java language, utilizing the JavaFX application framework.*

Resumo. *O relatório descreve o processo de planejamento e ilustra as estruturas de dados envolvidas no desenvolvimento de um Media Player em Java com o auxílio do framework JavaFX.*

1. Características da Aplicação

Na aplicação, cada usuário terá uma conta associada. Essa conta pode ter um status *premium* ou básico. Para contas premium, estão disponíveis as funcionalidades de criação, armazenamento e reprodução de Playlists personalizadas, contendo músicas de diferentes diretórios. Para usuários básicos, apenas músicas de um mesmo diretório podem ser reproduzidas na mesma fila de reprodução. Qualquer usuário pode adicionar um novo diretório ao sistema, e esse diretório estará disponível para todos os usuários da aplicação, independente do seu status.

O player, portanto, será dividido em duas telas principais: uma tela de login, que irá lidar com o manejamento de usuários do sistema, e a tela principal, responsável pela reprodução de músicas a partir de diretórios ou playlists. A figura 1 representa o desenho inicial das telas da aplicação. Nele, é possível perceber a utilização de uma única tela central para o acesso à todas músicas presentes no sistema, tanto de diretórios quanto de playlists. Também se destaca uma área horizontal dedicada às funcionalidades do player, bem como a indicação do título e artista da música atual.

2. Organização do Projeto

JavaFX é um framework que habilita o desenvolvimento de interfaces gráficas em Java, com elementos de XML. Aplicações escritas com o auxílio dessa ferramenta devem seguir o modelo MVC de organização de projetos, que divide as classes em pacotes de Modelo, Visão e Controle. O pacote de visão é composto por arquivos .FXML, que define a organização dos componentes gráficos das telas da aplicação. O pacote de controle possui métodos que integram os componentes gráficos da aplicação com as classes criadas no pacote de modelo.

Nesse sentido, o tocador de mídia será dividido em classes de modelo: representações de músicas, usuários e estruturas de dados encapsulando objetos de música. Classes de controle: inicialização de componentes gráficos e controle das ações de usuário. E classes de visão: organização estética dos componentes visuais.

Figura 1. Arte conceitual da tela inicial da aplicação.

3. Estruturas de Dados

As estruturas, suas características e as relações entre elas serão descritas por pacote. Cada pacote será analisado e explicado separadamente.

3.1. Classes de Modelo

As classes do pacote modelo são responsáveis pela manipulação dos dados do usuário e das músicas do tocador. A figura 3 contém o diagrama das classes do pacote modelo.

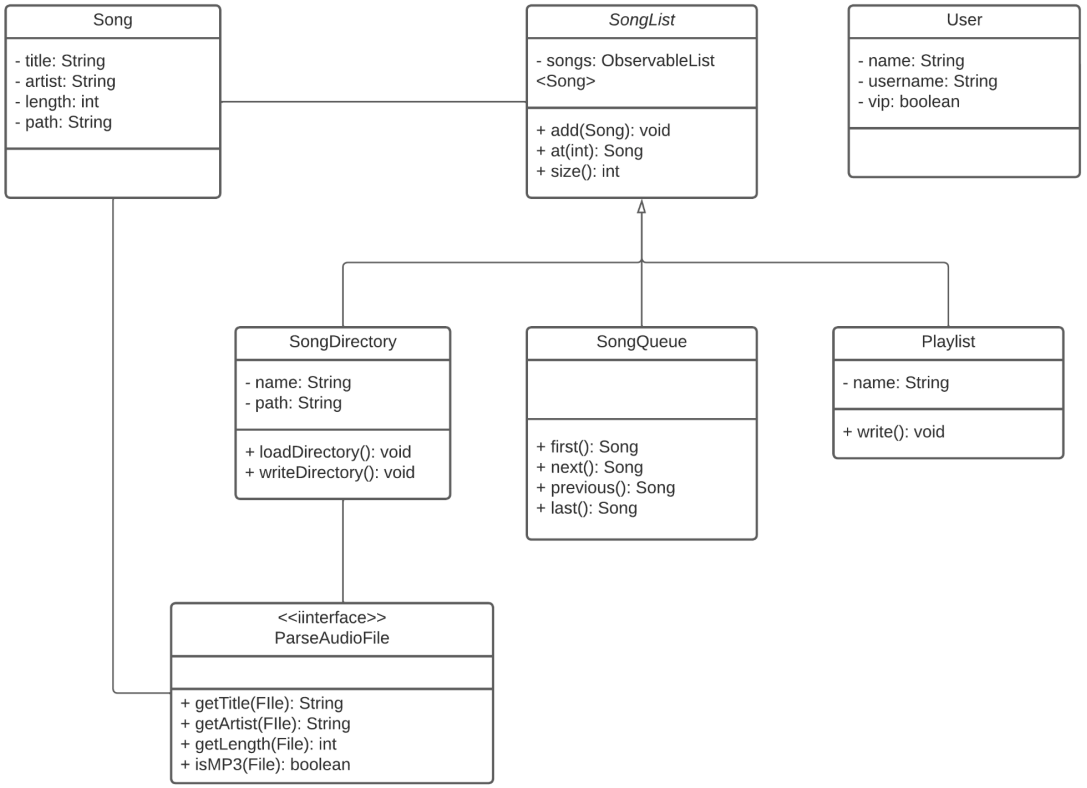


Figura 3. Diagrama de Classes do Pacote Modelo

3.1.1. Classe User

A classe User é composta pelos atributos *name*, *username* e *vip*, que armazenam características dos usuários do sistema. Os objetos User são utilizados no login, no registro (para verificar a existência de um *username* no sistema) e nas funcionalidades premium do sistema. Todas as contas criadas no sistema possuem seus campos *name*, *username* e *vip* escritos no arquivo *accounts.dat*.

3.1.2. Classe Song

A classe Song é responsável por armazenar objetos de música no sistema. Ela possui os campos *title*, *artist* e *length*, que guarda informações sobre o título, autor e duração da música, em segundos. A classe pode ser inicializada a partir de um construtor padrão, que recebe todos os campos, ou um construtor alternativo que recebe um arquivo MP3 e

checa as suas etiquetas para obter as informações necessárias. O processo de etiquetamento será explicado na interface ParseAudioFile.

3.1.3. Classe SongList

Como o nome sugere, a classe SongList armazena uma Lista Observável de objetos Song e implementa funcionalidades de adição e acesso à essa lista. A Lista Observável é uma estrutura de dados específica do framework JavaFX e é utilizada para povoar as tabelas de músicas da aplicação. SongList é uma classe abstrata e possui três herdeiros: SongDirectory, SongQueue e Playlist.

3.1.4. Classe SongDirectory

SongDirectory armazena um diretório de músicas presente no computador do usuário dentro da aplicação. Ela implementa funcionalidades de carregamento, leitura e escrita de diretórios. A leitura é feita a partir de um caminho absoluto fornecido pelo usuário através de um explorador de arquivos. A escrita é feita no arquivo *directory.dat*, para que a aplicação mantenha o diretório no sistema mesmo após a troca de usuários/fechamento do programa. SongDirectory também implementa a interface ParseAudioFile para verificar a extensão de cada arquivo do diretório.

3.1.5. Classe SongQueue

A classe SongQueue representa uma fila de reprodução. Ela implementa funcionalidades de avançar, retroceder ou ir ao início ou ao fim da fila. Ela possibilita que o MediaPlayer possa tocar músicas de forma indeterminada, circulando entre as músicas da fila sempre que a música atual se encerra.

3.1.6. Classe Playlist

Playlist implementa a funcionalidade premium de criação de listas de reprodução únicas para cada usuário. Ao contrário da classe SongDirectory, a classe Playlist não constrói sua Lista Observável música a música. Portanto, ela requer que a lista pronta seja passada em seu construtor. A escrita da Playlist no sistema também ocorre de forma diferente: na primeira linha é escrito o nome da playlist, e nas linhas subsequentes são escritos os caminhos absolutos para cada música presente na playlist. Nesse sentido, cada usuário VIP possui um diretório próprio na pasta *playlist*, nomeado com o seu *username*, e esse diretório contém todos os arquivos de playlists associados ao usuário.

3.1.7. Interface ParseAudioFile

A interface ParseAudioFile implementa as funcionalidades de etiquetamento utilizadas nas classes Song e SongDirectory. Os métodos, através da biblioteca *JAudioTagger*, verificam as etiquetas dos arquivos MP3 para obter informações sobre o título, artista e duração da música. Caso as etiquetas associadas ao título e artista estejam vazias, ao título é atribuído o nome do arquivo e o artista recebe o valor "Artista Desconhecido". Ela também implementa a checagem de extensão de um arquivo, para verificar se ele é do tipo .MP3.

3.2. Classes de Controle

O sistema possui duas classes de controle: LoginController, que lida com o login e cadastro de novos usuários do sistema, e MainController, que gerencia todos os outros aspectos da aplicação. A figura 4 exibe o diagrama de classes desse pacote.

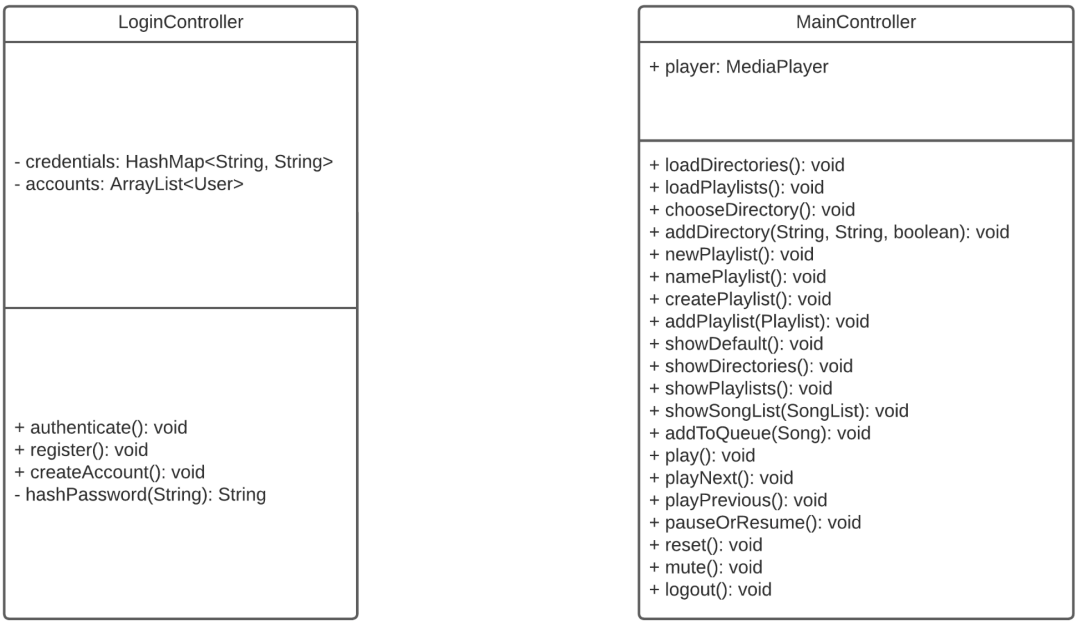


Figura 4. Diagrama de Classes do Pacote Controle.

3.2.1 Classe LoginController

LoginController implementa os métodos associados ao login e registro de usuários no sistema. Ela possui o atributo *credentials*, um HashMap que associa *usernames* registrados no sistema com as suas respectivas senhas. As senhas dos usuários são armazenadas criptografadas no arquivo accounts.dat. Além disso, a classe também

possui uma ArrayList de usuários, que armazena o *name* e status *vip* de cada *username* registrado.

Ao logar, a combinação de *username* + senha criptografada será procurada no HashMap. Caso seja encontrada, o username é buscado no ArrayList e o *name* e status *vip* associados são armazenados no objeto global *user*, que representa o usuário atual do sistema.

Ao criar uma nova conta, o *username* fornecido é checado no ArrayList. Caso esteja disponível, uma nova combinação *username* + senha criptografada é adicionada ao HashMap, e um novo objeto User é armazenado no ArrayList. As novas informações também são escritas no arquivo accounts.dat.

3.2.2. Classe MainController

A classe MainController implementa as funcionalidades do MediaPlayer, através do campo *player*. Esse campo é responsável por abstrair o processo de reproduzir uma faixa de música e é constantemente utilizado dentro dos métodos da classe. Além disso, o MainController implementa todas as funcionalidades da tela principal da aplicação. Entre elas:

- Criação, adição e visualização de diretórios e playlists;
- Visualização de músicas em tabelas;
- Escolha de músicas para criação de playlists;
- Criação de filas de reprodução;
- Tocar ou pausar a música atual;
- Pular para a próxima música ou para a música anterior;
- Trocar de conta.

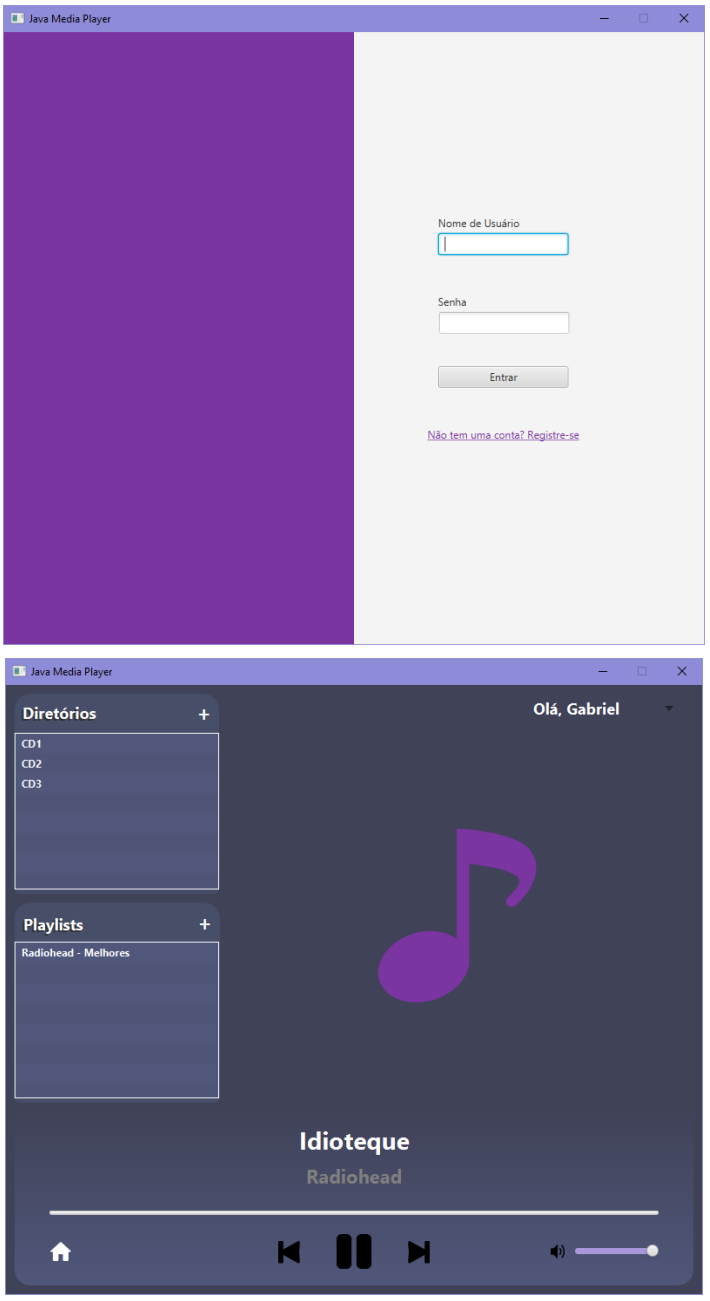
Quanto a campos específicos, o MainController armazena apenas inteiros auxiliares utilizados na construção da visualização em cadeia dos diretórios e playlists.

3.3. Classes de Visão

Em JavaFX, elementos gráficos são dispostos na tela por meio de um arquivo .FXML, e estilizados por um arquivo .CSS. No projeto, dois arquivos .FXML foram criados, Login.fxml e Main.fxml. Eles constituem os elementos gráficos da tela de login e da tela principal da aplicação, respectivamente.

4. Conclusão do Projeto

Todos os métodos descritos acima foram implementados nas suas respectivas classes, e o projeto foi organizado nos pacotes mencionados. Além disso, as telas foram construídas em arquivos .FXML e estilizadas através de um arquivo .CSS. O resultado do processo de estilização está disposto nas figuras 6 e 7.



Figuras 6 e 7. Telas Principais da Aplicação.

Referências

JavaFX Documentation Project. Projeto Open-Source do Github. Colaboradores: (https://github.com/FXDocs/docs/graphs/contributors). Fonte: (https://fxdocs.github.io/docs/html5/) Acessado entre 12 de Novembro e 4 de Dezembro de 2022.

JavaFX Documentation Home. Oracle Help Center. Fonte: (https://docs.oracle.com/javafx/2/). Acessado entre 12 de Novembro e 4 de Dezembro de 2022.