

JavaFy: Media Player em Java

Rubens Matheus Venancio Melo Oliveira ¹, Wisla Alves Argolo²

¹Instituto Metr pole Digital – Universidade Federal do Rio Grande do Norte (UFRN)
59078-900 – Natal – RN – Brazil

rubensmatheusvenancio@gmail.com, wisla.argolo.133@ufrn.edu.br

Abstract. *This report describes the development of a Media Player in Java, which has regular and VIP users and allows management of songs, playlists, directories and the users themselves, as well as user authentication. To do this, screens were created for user interaction, text files for data persistence and DAOs to manage them, as well as the implementation of exception handling to ensure the stability of the system. It also explores the use of inheritance and polymorphism in the project structure.*

Resumo. *Este relat rio descreve o desenvolvimento de um Media Player em Java, que possui usu rios comuns e VIPs e permite gerenciamento de m sicas, playlists, diret rios e dos pr prios usu rios, al m de autentica  o de usu rio. Para isso, foram criadas telas de intera  o com o usu rio, arquivos de textos para a persist ncia dos dados e DAOs para administr -los, bem como a implementa  o de tratamento de exce  es para assegurar a estabilidade do sistema. Al m disso, explora o uso de heran a e polimorfismo na estrutura do projeto.*

1. Introdu  o

Este relat rio aborda o desenvolvimento do sistema *JavaFy*, um *Media Player* em Java, implementado como parte da disciplina de Linguagem de Program  o II.

Nesse sentido, o projeto visa aplicar conceitos chave da program  o orientada a objetos, como organiza  o de pacotes, interfaces gr ficas, uso de bibliotecas externas, documenta  o em JavaDOC, heran a, polimorfismo, classes abstratas e tratamento de exce  es.

A partir disso, o sistema possui diversas funcionalidades, incluindo a possibilidade de *login*, cria  o de playlists para usu rios VIP e adi  o/remo  o de m sicas, diret rios e playlists. Al m disso, o programa permite a reprodu  o de m sicas e controle dessa m sica. Para isso, o projeto utiliza arquivos de texto para manter os dados salvos, garantindo a persist ncia das informa  es de diret rios, m sicas e playlists entre as sess es.

Nesse sentido, este relat rio busca descrever a implementa  o do *Media Player* e as decis es de projeto tomadas.

2. Materiais

2.1. Tecnologias utilizadas

O sistema de *Media Player* foi implementado na linguagem de program  o Java e mais detalhes podem ser obtidos a partir da Tabela 1.

Tabela 1. Materiais utilizados

Versão Java	JavaFX	SceneBuilder	IDE
17.0.8	JavaFX SDK 17	21.0.0	Eclipse 2022-09 (4.25.0)

Além disso, arquivos CSS foram empregados para a estilização da interface de usuário.

3. Organização do projeto

3.1. Organização dos pacotes

O sistema foi projetado de modo a separar suas funcionalidades em pacotes.

O pacote **application** contém a classe **Main.java**, que é o ponto de entrada do aplicativo.

Em **modelo** existem as classes básicas do sistema, incluindo **Usuari**, **UsuarioVIP**, **Musica** e **Playlist**.

O pacote **dao** possui as classes que permitem o acesso aos dados, responsável pela persistência e recuperação das informações dos usuários, músicas, diretórios e playlists do sistema.

Para lidar com a interação entre interface do usuário e dados do sistema, existem as classes controladoras em **controle**.

Já **dados** é responsável por armazenar arquivos de texto que guardam dados persistentes. A aplicação foi organizada de forma que cada usuário possui um arquivo próprio para suas músicas e diretórios, garantindo a individualidade dos dados. Há também um arquivo central, **playlistGeral**, que contém os caminhos para as playlists de todos os usuários. Por fim, existe um arquivo **usuarios** que lista todos os usuários cadastrados na aplicação.

O pacote **excecoes** apresenta as classes de exceções personalizadas para lidar com situações específicas da aplicação. Neste projeto, apenas uma classe de exceção personalizada foi criada, estendendo *RuntimeException*.

Ademais, as classes auxiliares foram armazenadas em **util**, incluindo classes para alertas e gerenciadores da mudança de interface gráfica.

3.2. Herança e Polimorfismo

Neste projeto, os conceitos de herança e polimorfismo foram aplicados para promover uma arquitetura flexível.

No sistema, o uso de herança esteve presente nas classes **Usuario** e **UsuarioVIP**, uma vez que **UsuarioVIP** herda as propriedades de **Usuario**, mas se diferencia pela habilidade exclusiva de criar playlists, uma funcionalidade que não disponível para os usuários comuns.

Em relação ao polimorfismo, ele foi utilizado na gestão de usuários. Por exemplo, quando o método **autenticar** em **TelaLoginController** verifica as informações de um

usuário, ele trata esse usuário como um tipo genérico de *Usuario*, assim como as classes DAOs em métodos como carregar, adicionar e remover. Além disso, no método *adicionarPlaylistAcao* em *TelaAdicionarPlaylistController*, o usuário atual - isto é, logado - como um *UsuarioVIP*.

3.3. Classes Abstratas, Interface e Tratamento de Exceções

Neste projeto, não foram utilizadas classes abstratas ou interface, enquanto o tratamento de exceções atua assegurando a robustez e a confiabilidade do aplicativo.

Para isso, nas classes DAOs, quando uma operação de leitura ou escrita em arquivo falha, o sistema lança uma exceção de I/O. Mais especificamente, em *UsuarioDAO*, os métodos carregar, adicionar e remover podem lançar exceções de I/O, mas também faz uso de exceções personalizadas, como *ExcecaoPersonalizada*, para lidar com situações específicas, como a tentativa de adicionar um usuário que já existe ou remover um que não existe para que a aplicação responda de forma apropriada a esses casos.

Para prevenir falhas na aplicação, essas exceções são capturadas pelas classes controladoras que invocam esses métodos, onde os erros são tratados através da exibição de alertas personalizados, que fornecem informações sobre o erro ou alerta para o usuário.

3.4. Documentação Javadoc

A utilização do Javadoc para documentar o código-fonte é uma prática importante para compreensão do sistema.

Nesse sentido, neste projeto cada classe e método possui comentários Javadoc detalhados, descrevendo sua funcionalidade, parâmetros, valores de retorno e exceções lançadas. Isso permite o entendimento do funcionamento do sistema, bem como lógica de negócios utilizadas - como as operações em DAO - e organização estabelecida.

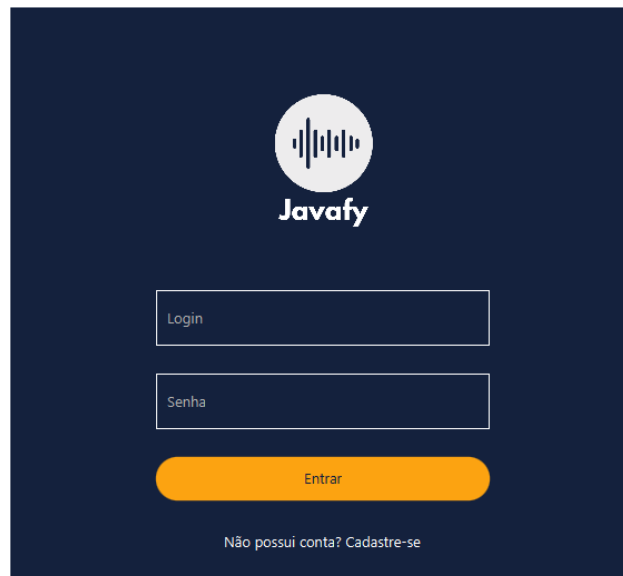
3.5. Interface Gráfica

As interfaces gráficas do *Media Player* foram criadas utilizando a tecnologia JavaFX, complementadas pelo SceneBuilder e arquivos de estilo CSS.

A Figura 1 mostra a tela de login, que é a tela principal do sistema, já que representa o ponto inicial de interação com o usuário. Nela, são solicitados os dados do usuário - *login* e senha - para acesso à conta. Alternativamente, para novos usuários, um link direciona para a tela de cadastro, facilitando a navegação no sistema.

A partir da tela de cadastro (Figura 2), o usuário pode adicionar suas informações - nome, *login* e senha -, bem como em que tipo de usuário ele se enquadra, sendo VIP ou comum. O sistema assegura que todos os campos sejam preenchidos, emitindo alertas para quaisquer campos deixados em branco (Figura 3). Uma vez que o usuário conclui o preenchimento e clica no botão **Cadastrar**, suas informações são armazenadas no arquivo *usuarios.txt*, de modo que esse usuário pode acessar o sistema posteriormente ao fechar a janela de cadastro.

Ao entrar no sistema, a tela do usuário comum é exibida na Figura 4. A partir dos botões na parte lateral esquerda, o usuário é capaz de adicionar ou remover diretórios e, consequentemente, as músicas deles. Na parte central, são listadas as músicas desse



The login screen features a dark blue background. At the top center is the Javafy logo, which consists of a white circle containing a black waveform icon, with the word "Javafy" in white text below it. Below the logo are two white rectangular input fields: the first is labeled "Login" and the second is labeled "Senha". Underneath these fields is a prominent orange rounded rectangular button labeled "Entrar". At the bottom center, there is a small white text link that reads "Não possui conta? Cadastre-se".

Figura 1. Tela de Login



The registration screen has a dark blue background. At the top center, the word "Cadastro" is displayed in white, underlined with a thin orange line. Below this are four white input fields: "Nome", "Login", "Senha", and a dropdown menu currently showing "Comum" with a small downward arrow. At the bottom center is a white rounded rectangular button labeled "Cadastrar".

Figura 2. Tela de Cadastro

usuário. Ao clicar no botão acima da listagem, ele é capaz de adicionar novas músicas individualmente, mas para remover é preciso selecionar uma música e clicar com mouse.

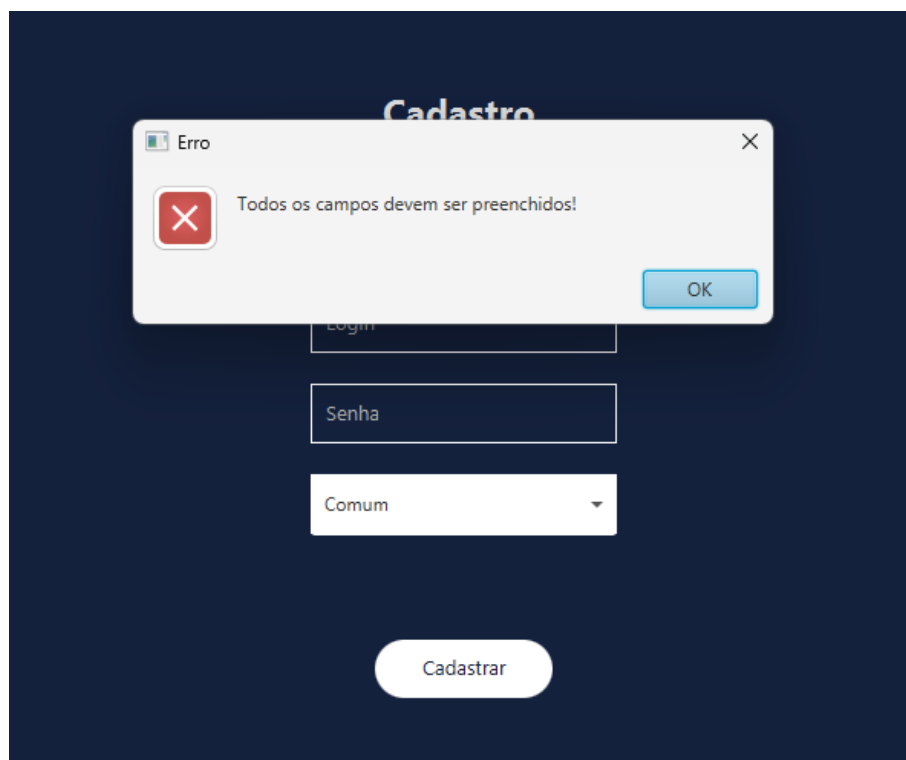


Figura 3. Tela de Erro em Cadastro

Na parte inferior dessa tela, é possível reproduzir uma música, bem como pausar, parar, ir para posterior e retroceder, bem como mutar a música.

No canto superior direito da interface, um ícone de usuário. Ao clicar neste ícone, o usuário encontra as opções "Conta", que o direciona para a tela com detalhes da sua conta, e "Sair", que encerra a sessão atual e redireciona para a tela de login.

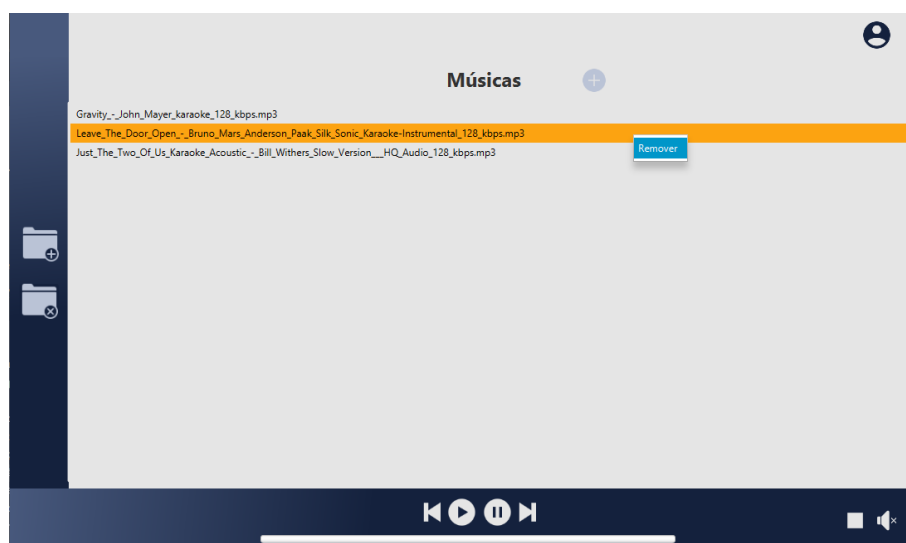


Figura 4. Tela Usuário Comum

A tela de Conta do usuário comum (Figura 5) exibe o *login* e nome do usuário, bem como as opções de excluir e sair da conta. Ao clicar em qualquer uma dessas opções,

a aplicação é redirecionada para tela principal. Ademais, também é possível retornar para a tela anterior a partir do botão no canto superior esquerdo.

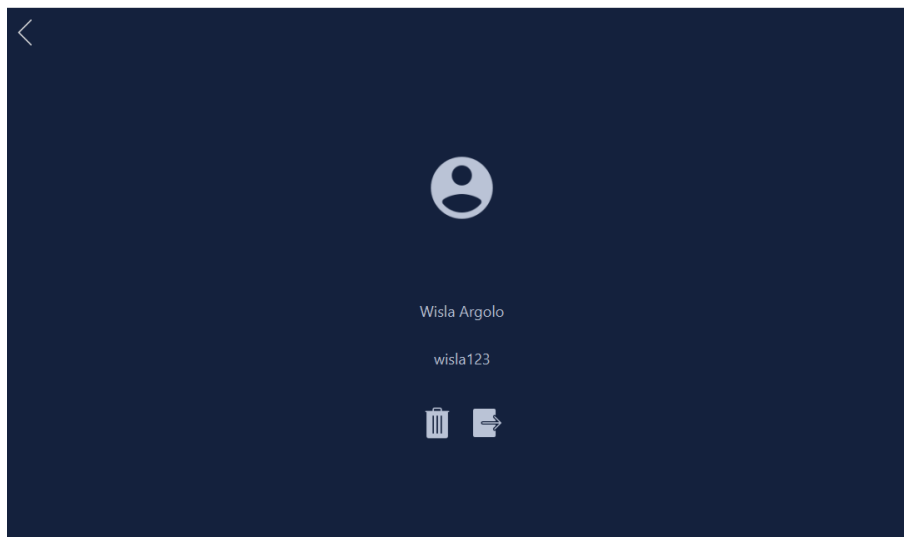


Figura 5. Tela Conta Usuário Comum

Para o usuário VIP, quando ele entra no sistema, a tela exibida (Figura 6) é semelhante ao do usuário comum (Figura 4) com a adição da listagem de playlists. Nesta área, o usuário VIP tem a opção de adicionar novas playlists através de um botão acima da lista, bem como remover playlists ao clicar com mouse sobre uma playlist da lista.

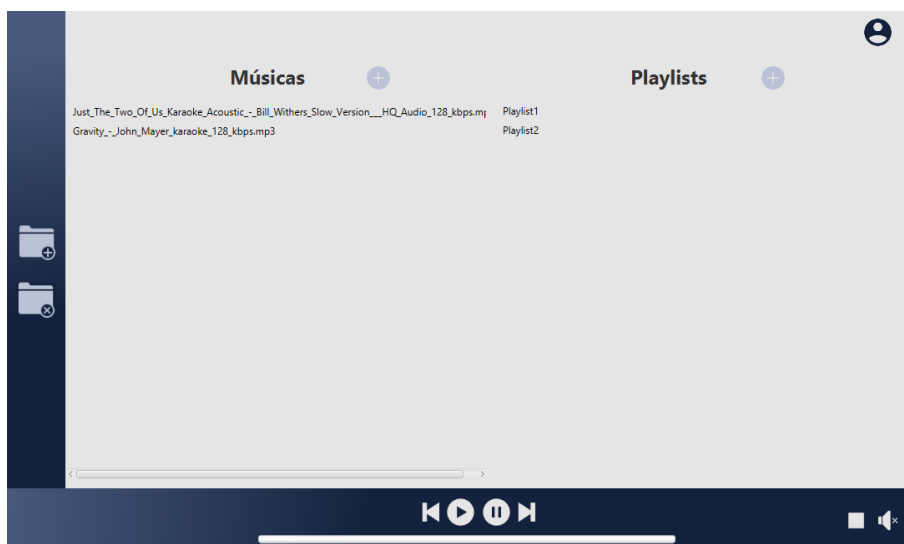


Figura 6. Tela Usuário VIP

Ao clicar nesse botão, uma janela é aberta (Figura 7) e o usuário pode escolher um nome para sua nova playlist.

Além disso, ao clicar duas vezes em alguma das playlists listadas na tela da Figura 6, o usuário é redirecionado para a tela de uma playlist específica (Figura 8). Nela, é possível adicionar e remover músicas de uma playlist específica a partir da lista das

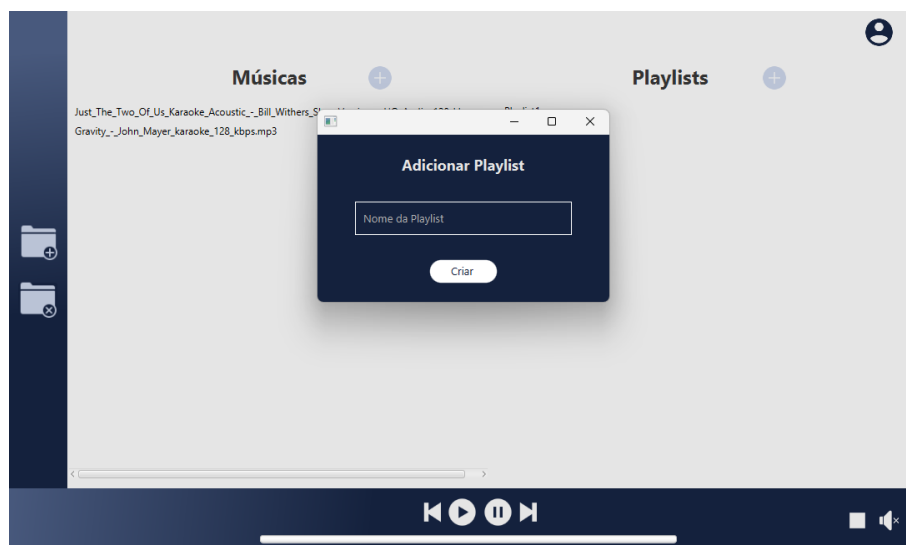


Figura 7. Janela Adicionar Playlist

músicas disponíveis ao lado, bem como reproduzir as músicas dessa playlists. O usuário também pode retornar para a tela da Figura 7 pelo botão no canto superior esquerdo.

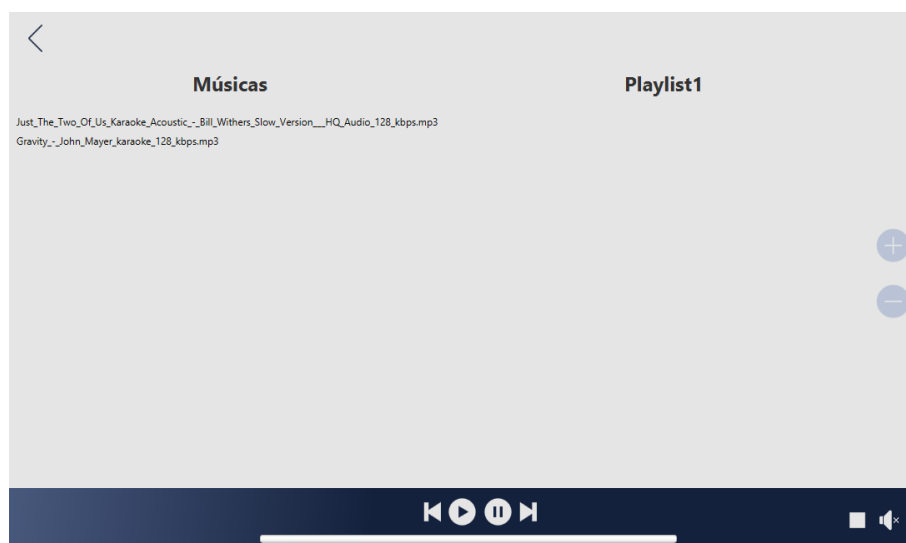


Figura 8. Tela Playlist

Por fim, para o usuário VIP a tela de conta (Figura 9) exibe um ícone indicando que ele é VIP.

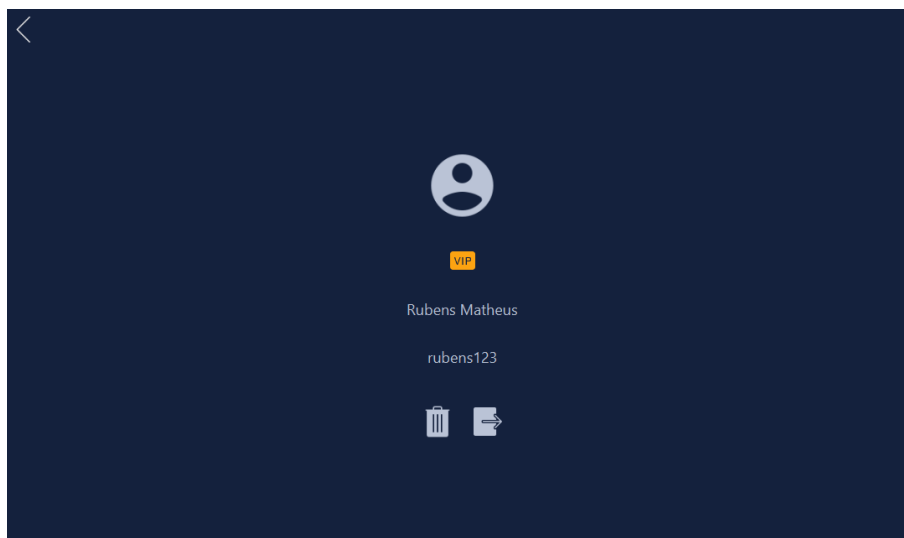


Figura 9. Tela Conta VIP

3.6. Controle de Usuário

Neste projeto, o controle de usuários é baseado na autenticação por *login* e senha. Uma vez autenticado, o sistema reconhece o tipo do usuário — VIP ou comum — e concede as permissões correspondentes. Usuários comuns têm capacidade para gerir suas coleções de músicas e diretórios, enquanto os VIPs possuem funcionalidades adicionais, como a criação e gerenciamento de playlists.

A autenticação é feita a partir da leitura do arquivo *usuarios.txt* que contém as informações de todos os usuários cadastrados, incluindo o tipo de cada um. Se o usuário for VIP, a tela de músicas do *Media Player* também vai exibir uma listagem de playlists e uma nova tela para gerenciá-las será liberada.

3.7. Armazenamento dos Dados

3.7.1. Diretório de Músicas

Ao iniciar o player o programa vai carregar as músicas disponíveis para cada usuário, essas músicas estarão salvas em um diretório, cada usuário terá seu próprio diretório e ao utilizar o player pela primeira vez depois de fazer o cadastro, os clientes não terão músicas cadastrar. Para fazer esse controle utilizamos de um arquivos chamado *diretorios_login.txt*, sendo *login* substituído pelo real login do usuário, cada cliente terá o seu arquivo, o conteúdo dos arquivos será o caminho para as pastas que contém as músicas. Os *.txt's* de todos usuários estarão na pasta *diretorios*. A classe *DiretorioDAO* lida com esses arquivos carregando e realizando operações de adição e remoção conforme necessário.

3.7.2. Arquivos de Músicas

O usuário também tem a possibilidade de incluir músicas individualmente para fazer isso criamos a classe *musicaDAO* que é responsável primeiramente por carregar essas músicas assim que o usuário fizer login. As músicas são armazenadas em um arquivo

musicas_login.txt contendo nome e caminho para música adicionada individualmente do usuário, esse arquivos ficam armazenados um diretório chamado *musicas*. Após o carregamento é possível fazer o gerenciamento delas, podendo adicionar mais ou remover.

3.7.3. Arquivo de Usuários

Para armazenar todos os usuários do sistema, é utilizado um arquivo de texto *usuarios.txt*. Nele, cada linha representa um usuário e possui suas informações, incluindo nome, *login*, senha e tipo (VIP ou comum). A classe *UsuarioDAO* acessa esse arquivo, realizando a autenticação dos usuários ao consultar suas informações e adicionando novos usuários ao arquivo quando eles são cadastrados.

3.7.4. Diretório de Playlists

Em relação ao gerenciamento de playlists para os usuários VIPs, existe um arquivo *playlistsGeral.txt* que contém as informações de todas as playlists do sistema, isto é, em cada linha desse arquivo tem o nome e o caminho para o arquivo de texto de uma playlist específica.

Além disso, cada playlist possui um arquivo com o título seguindo o modelo *playlist_login_nome.da.playlist.txt*. Esse arquivo contém o nome e *login* do usuário na primeira linha e o nome da playlist na segunda. As linhas subsequentes listam as músicas da playlist, incluindo seus nomes e caminhos.

Ao inicializar o sistema, o *playlistDAO* lê o arquivo *playlistsGeral.txt* e procura as playlists do usuário atual e carrega as músicas contidas nelas. Após o carregamento é possível fazer operações como adição de playlist, adição de músicas a playlist, remoção de músicas de playlist e deletar playlists.

4. Conclusão

A implementação do *Media Player*, o *JavaFy*, envolveu a criação de DAOs, a persistência de dados a partir de arquivos de texto, herança, polimorfismo, tratamento de exceções e organização de pacotes, além da construção de uma interface de fácil uso para o usuário.

À vista disso, o desenvolvimento e resultados do trabalho atenderam os objetivos propostos, dado que proporcionaram uma compreensão mais aprofundada sobre conceitos importantes da programação orientada a objetos.