Implementação de player de música utilizando JavaFX

Ian Gabriel S. Dias

[1]Instituto Metrópole Digital - Universidade Federal do Rio Grande do Norte (UFRN)

Resumo. *Este relatório descreve uma implementação de um player de música utilizando a linguagem Java. O projeto foi realizado para a disciplina de Linguagem de Programação II, valendo nota para a terceira unidade. O projeto trabalhou com os principais conceitos de Programação Orientada a Objetos e possui interface gráfica. A interface gráfica foi feita com a biblioteca JavaFX.*

1. Introdução

Este trabalho fala sobre a implementação de um tocador de músicas utilizando a linguagem de programação Java. Este projeto foi realizado como avaliação na disciplina de Linguagem de Programação II. O objetivo deste trabalho era implementar um tocador de músicas funcional utilizando conceitos de POO trabalhados nas aulas.

O player de música seguiu as seguintes especificações:

- O player deveria possuir autenticação e autorização de usuários, separando-os em dois grupos: Usuários VIP e usuários comuns.
- Os usuários poderiam adicionar músicas e diretórios dos seus arquivos pessoais para serem tocadas pelo player posteriormente.
- Os usuários VIP poderiam criar playlists com as músicas adicionadas.
- O player deveria salvar todas as informações em arquivos para que elas persistissem quando o player fosse aberto novamente.
- O player deveria possuir uma interface gráfica para enriquecer a experiência do usuário.

2. Metodologia

2.1. Visão geral

O projeto seguiu o padrão MVC (Model - View - Control) [wag 2020]. A primeira etapa da construção do tocador foi decidir quais seriam as classes de modelo e suas relações. Após a tomada das decisões, foi construído um diagrama de classes para facilitar a visalização da organização da camada de modelo. O diagrama de classes está representado na Figura 1.

Além das camadas de modelo, visão e controle, foi implementada uma quarta camada DAO (Data Access Object). Esta camada tem a responsabilidade de pegar os dados da camada de modelo e salvar no banco de dados (neste caso, arquivos .txt). Após a construção do diagrama de classes, as camadas Modelo e DAO foram implementadas simultaneamente.

Na camada DAO, todas as classes utilizaram o padrão Singleton [gur]. Esta decisão foi tomada pois era importante que em toda a aplicação houvesse apenas uma única instância de cada classe DAO ativa. Isso foi necessário para que não houvesse perda de dados. Assim, os mesmos dados seriam usados por todas as outras camadas da aplicação.

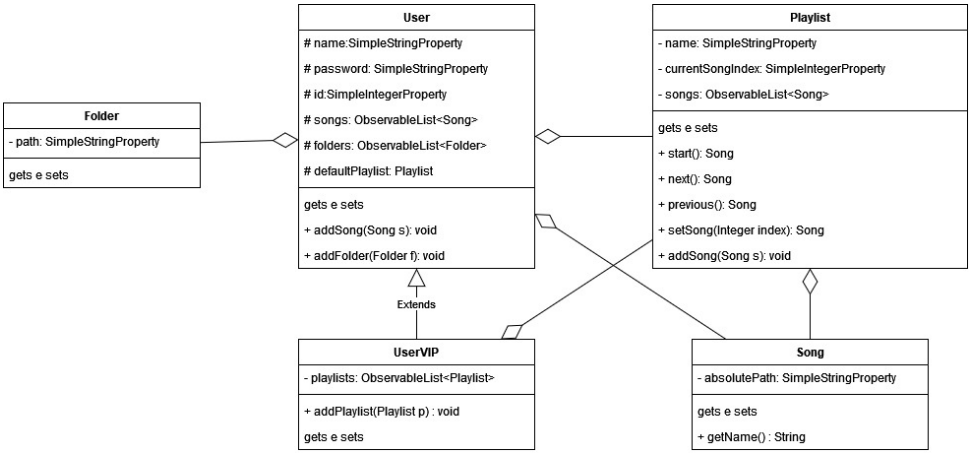

Figura 1. Diagrama de classes do projeto

Em seguida foi construída a interface gráfica do tocador. A interface foi feita utilizando a biblioteca JavaFX [jav]. O desenho das telas foi feito com o software SceneBuilder. Durante a construção da interface foram implementadas as camadas de Controle e Visão, responsáveis pela lógica da resposta do tocador com a interação do usuário.

Para os atributos das classes de modelo, foi decidido que seria utilizado as classes `Properties` do JavaFX. São exemplos de `Properties` as classes `SimpleStringProperty` e `SimpleIntegerProperty`. A vantagem de utilizar estes tipos é que a interface do JavaFX pode escutar por mudanças nos valores dos atributos, atualizando automaticamente.

Uma outra estrutura utilizada foi a `ObservableList`. Esta estrutura representa uma lista que pode ter suas mudanças observadas pela interface, que pode atualizar de acordo. Esta estrutura é especialmente útil nas listas da aplicação, que precisam refletir as mudanças assim que o usuário adiciona uma nova música ou pasta.

2.2. Classes de Modelo

2.2.1. User

Esta classe representa o usuário da aplicação. As credenciais do usuário são armazenadas nos atributos `name` e `password`. O usuário também possui um `id` único para identificação, que é gerado automaticamente pela aplicação.

Além disso cada usuário também possui a sua própria lista de músicas e pastas adicionadas. Quando o usuário adiciona um arquivo de música pelo método `addSong`, esta música é colocada na playlist padrão. Quando o usuário adiciona uma nova pasta pelo método `addFolder`, todos os arquivos de música contidos nela são adicionados para a playlist padrão.

A playlist padrão é a única que esta classe possui. Apenas uma instância da classe `UserVIP` pode possuir mais playlists.

2.2.2. UserVIP

Esta é uma classe que herda de `User` que representa um usuário com o plano VIP. Este usuário possui a mais um atributo que é uma lista de playlists. Estas playlists são adicionadas pelo método `addPlaylist`

2.2.3. Folder

Esta classe representa um diretório adicionado pelo usuário. Esta classe contém um único atributo que representa o caminho absoluto até a pasta adicionada.

2.2.4. Song

Esta classe representa uma música que pode ser tocada pela aplicação. Esta classe possui um atributo que representa o caminho absoluto até o arquivo de música. Além deste atributo e de seus gets e sets, a classe também possui um método `getNome` retorna o nome do arquivo através do seu caminho absoluto.

2.2.5. Playlist

Esta classe representa uma lista de músicas customizada pelo usuário. A lista das músicas da playlist fica representada pelo atributo `songs`. A classe também possui o atributo `currentSongIndex` que controla qual o índice da música que está sendo tocada.

Os métodos `start`, `next`, `previous` e `setSong` alteram o valor do atributo `currentSongIndex`. Assim estes métodos controlam qual música está sendo tocada no momento.

2.3. Descrição das telas

2.3.1. Tela de login

A primeira etapa da construção da interface foi a tela de login. A tela possui um formulário com 2 campos: usuário e senha, um botão para submeter as informações (fazer login) e outro que abre a tela de cadastro de usuário. A validação dos campos foi feita pela camada de controle. A busca pelo usuário a partir das credenciais nos arquivos foi feita pela camada DAO.

A figura 2 mostra como ficou a aparência da tela de login da aplicação.

2.3.2. Tela de cadastro

A tela de cadastro possui um formulário com dois campos de texto, usuário e senha, e uma CheckBox perguntando se o usuário é VIP. Além disso a tela possuia dois botões, um para cancelar o cadastro e outro para cadastrar o usuário. Ambos os botões fechavam a janela, mas o botão de cadastro chamava a camada DAO para salvar o novo usuário em um arquivo.

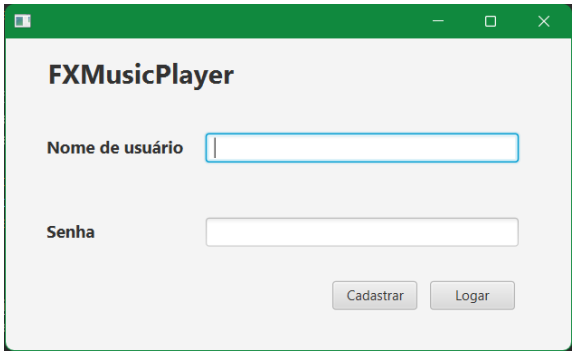

Figura 2. Tela de login da aplicação

A camada de controle também validava os dados e mostrava um alerta para o usuário quando ele fosse cadastrado com sucesso ou quando houvesse campos inválidos. A Figura 3 mostra o alerta em ação.

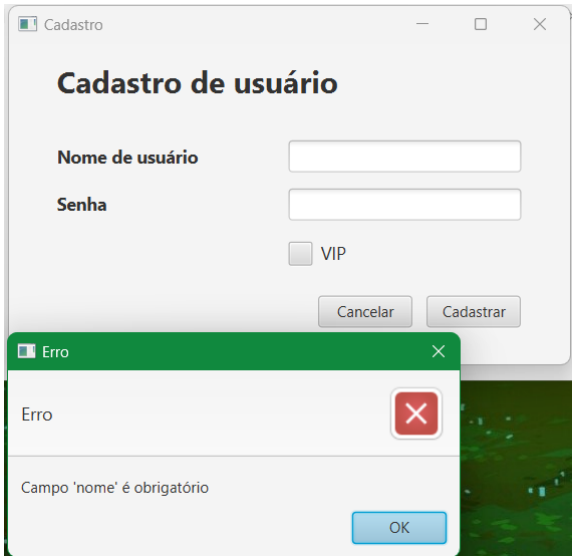

Figura 3. Tela de cadastro da aplicação e alerta de erro

2.3.3. Tela principal

O usuário é direcionado para esta tela após logar com sucesso no sistema. Nela é possível visualizar todas as músicas adicionadas pelo usuário. Também é possível adicionar novas músicas ou pastas.

Na parte inferior da tela há os controles do tocador de músicas. Os controles permitem pausar, retomar ou navegar entre as músicas. O usuário também pode focar na lista de músicas e selecionar a música a ser tocada utilizando as setas.

No canto direito há um resumo do perfil do usuário e uma lista de suas playlists. Por lá o usuário pode clicar em um botão que o direcionará para a tela de criação de playlists. O usuário também pode selecionar a playlist a ser tocada focando na lista de playlists e utilizando as setas.

Na figura 4 podemos ver a estrutura da tela principal. Segue abaixo uma descrição dos seus elementos:

1. Lista com músicas adicionadas pelo usuário
2. Lista com músicas da playlist selecionada
3. Botões para adicionar novas músicas e diretórios
4. Controles do player
5. Informações do usuário
6. Lista de playlists do usuário

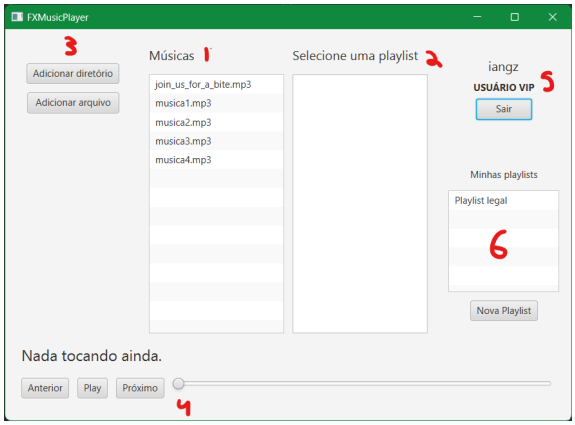

Figura 4. Tela Principal da aplicação

2.3.4. Tela de criação de playlist

Esta tela possui um campo de texto que recebe o nome da playlist. Na parte central da tela, há duas listas e dois botões. Uma lista contém as músicas disponíveis, enquanto a outra lista as músicas já selecionadas que farão parte da playlist. Os botões do meio transferem as músicas selecionadas de uma lista para outra. Quando o usuário criar a playlist, apenas as músicas da lista da direita farão parte dela.

A figura 5 mostra a aparência final da tela de criação de playlists.

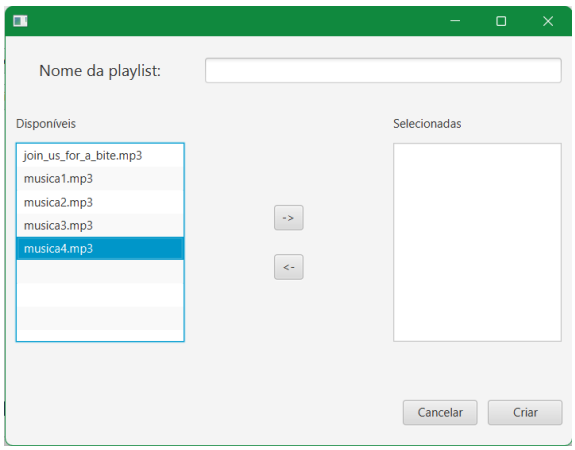

Figura 5. Tela de criação de playlists

3. Conclusão

Podemos concluir que projeto do player de música agregou o conhecimento do autor sobre os conceitos da Programação Orientada a Objetos. No projeto foram trabalhados conceitos de herança, interface, composição e tratamento de erros. Isto ajudou o autor não só a fixar os conceitos mas também o ensinou a como aplicá-los em uma aplicação real.

O trabalho também envolveu tecnologias atualizadas, como a biblioteca JavaFX, e padrões de projeto amplamente utilizados como MVC e Singleton. O uso destes padrões é certamente útil para o mercado de trabalho. Logo, este projeto ajudou a preparar o autor para sua vida profissional.

Referências

Java fx website. https://openjfx.io/index.html. Acessado em: 02-12-2023.

Singleton design pattern - refactoring guru. https://refactoring.guru/design-patterns/singleton. Acessado em: 02-12-2023.

(2020). O que é padrão mvc? entenda arquitetura de softwares! https://blog.lewagon.com/pt-br/skills/o-que-e-padrao-mvc/#:~:text=O%20MVC%20%C3%A9%20uma%20sigla,sejam%20mais%20r%C3%A1pidas%20e%20din%C3%A2micas. Acessado em: 01-12-2023.