Resumo Projeto Prático I - MC 322 - Profa. Dra. Esther Luna Colombini

Luiza Viana Souza - 241126

Maria Cecília Torres Vianna - 260230

Sofia Valverde Villas Bôas - 252974

Echo - Sistema de Avaliação Musical

INTRODUÇÃO

A partir da ideia da possibilidade de uma plataforma digital que facilitaria usuários a publicar e compartilhar opiniões sobre conteúdos musicais, nasceu o Echo, um MVP[1] que permitiria armazenar informações sobre usuário, álbuns, artistas, músicas, publicações e todos os outros componentes existentes dentro desse meio artístico. Nessa versão, o produto funciona como um "diário" musical, em que o usuário pode explorar, salvar, fazer reviews e curtir os conteúdos relacionados à discografia da cantora Taylor Swift.

DESENVOLVIMENTO

Com o intuito de permitir principalmente a elaboração de *reviews*[2] sobre um fonograma musical, o sistema foi dividido em diversas classes Java, todas seguindo princípios da programação orientada a objetos. Entre as funções das classes, temos:

- **Classe Álbum:**
  - addMusica(Musica musica): adiciona uma música ao álbum;
  - getMusicabyFaixa(int faixa): retorna uma música pelo seu número de ordenação;
  - mostrarMusicas(): lista todas as músicas do álbum;
  - getters, setters e toString().
- **Classe Artista:**
  - addSingle(Musica musica): adiciona uma única música ao vetor de singles;
  - addAlbum(Album album): adiciona um álbum à discografia do artista;
- **Classe Conteudo:**
  - addReview(Review review): adiciona a review na lista de reviews do conteudo e atualiza sua nota baseado na avaliação dada;

[1] Minimum Viable Product
[2] resenhas, críticas

    - removeReview(Review review): remove a review na lista de reviews do conteudo e atualiza sua nota;
    - getters, setters e toString().
- **Classe Gravadora:**
    - popularidade(Conteudo c): calcula a popularidade de um conteúdo, baseado na média aritmética das suas reviews;
    - addArtista(Artista artista): adiciona um artista à lista da gravadora;
    - addAlbum(Album album): adiciona um álbum à lista da gravadora;
    - getters, setters e toString().
- **Classe Lista:**
    - criarLista(): interage com o usuário para criar uma nova lista;
    - adicionarConteudo(Conteudo c): adiciona um conteúdo à lista;
    - removeConteudo(Conteudo c): remove um conteúdo da lista;
    - getters, setters e toString().
- **Classe Música:**
    - getters, setters e toString().
- **Classe Review**
    - criarReview(): método abstrato que interage com o usuário para a criação de uma review de um certo conteúdo (mais detalhes em cada implementação das classes filhas);
    - getters, setters e toString().
- **Classe ReviewAlbum:**
    - criarReview(): coleta informações do usuário para criar uma avaliação de um álbum e permite a avaliação de músicas individuais contidas no álbum;
    - getters, setters e toString().
- **Classe ReviewMusica:**
    - criarReview(): coleta informações do usuário para criar uma avaliação de uma música;
    - getters, setters e toString().
- **Classe Usuário:**
    - publicarReviewMusica(Musica m): publica uma nova review de uma música e a adiciona à lista de reviews do usuário;
    - publicarReviewAlbum(Album a): publica uma nova review de um álbum e a adiciona à lista de reviews do usuário;
    - curtirReview(Review review): adiciona uma curtida à review passada como parâmetro;
    - addAmigo(Usuario amigo): adiciona um usuário à lista de amigos;
    - removeAmigo(Usuario amigo): remove um usuário à lista de amigos;
    - publicarLista(): cria uma nova lista e a adiciona à lista de listas;

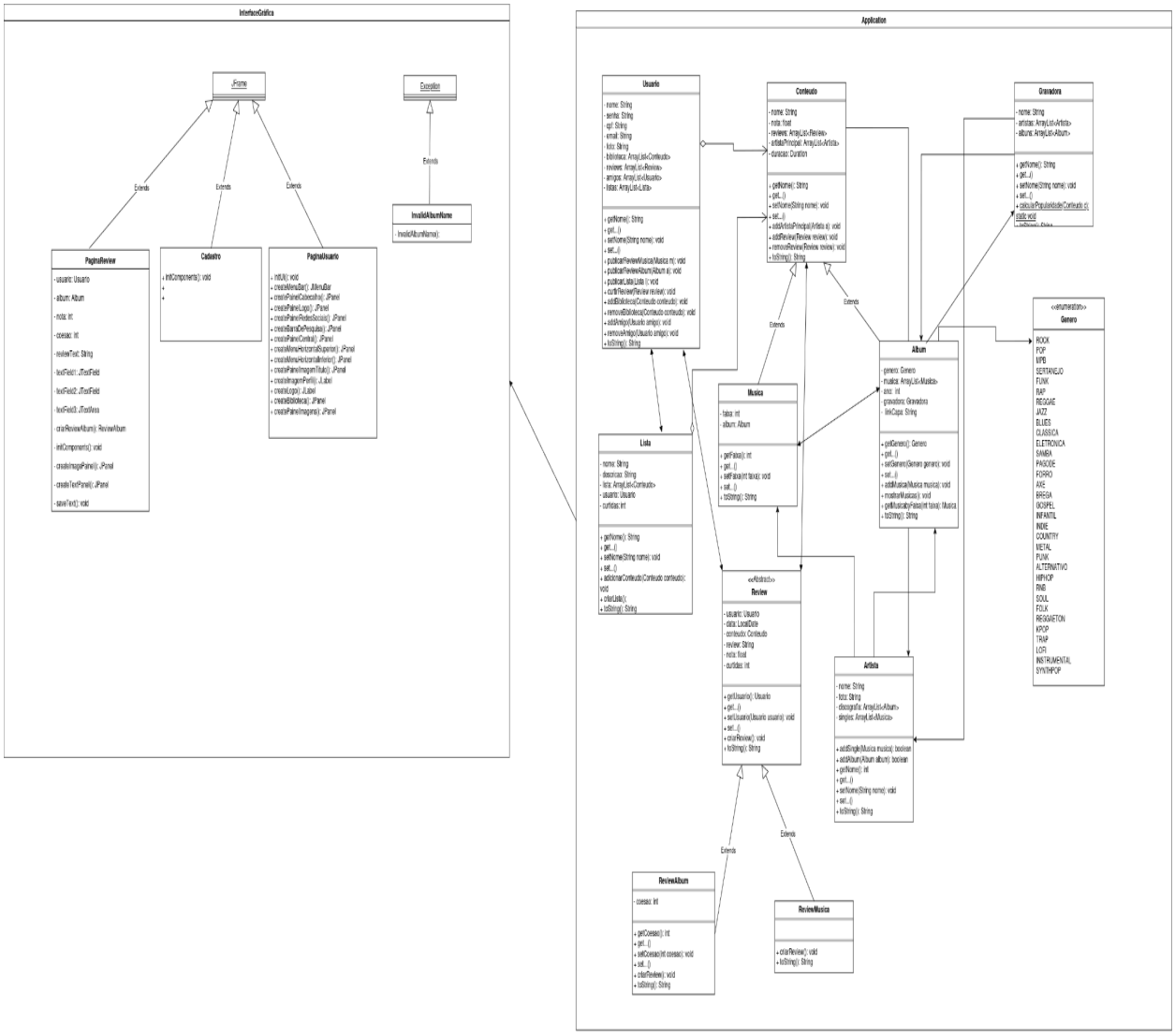
    - addBiblioteca(Conteudo conteudo): adiciona um conteúdo à biblioteca;
    - removeBiblioteca(Conteudo conteudo): remove um conteúdo da biblioteca;
    - getters, setters e toString()
- **Classe InvalidAlbumName:**
    - InvalidAlbumName(): construtor sem parâmetros que chama o construtor da superclasse (Exception) com uma mensagem de erro específica: "O álbum não foi encontrado! Por favor, digite um nome válido."
- **Classe JSONParser:**
    - parse(String json): recebe uma string com o que foi lido de um arquivo JSON e o devolve como um objeto do tipo JsonNode, da biblioteca jackson-databind;
    - albumFromJson(JsonNode node, Artista artista): lê todos os detalhes e informações inseridas no nó formado pela leitura anterior do arquivo JSON, o converte em objetos da classe Album e retorna um vetor com todos;
    - tracksFromJson(JsonNode node, ArrayList<Album> albums): lê os atributos de cada música, cria objetos da classe Musica e as insere na tracklist do seu respectivo álbum.
- **Classe InterfaceGrafica**
    - chamarCadastro(): inicia a interface gráfica
- **Classe PaginaUsuario**
    - initUI(): Inicializa a interface do usuário: título, operações de fechamento, tamanho, cor de fundo e layout. Configura a barra de menus e adiciona painéis de cabeçalho e central. Centraliza e exibe a janela.
    - createMenuBar(): Cria uma barra de menus com itens "Feed", "Settings" e "Log out".
    - createPainelCabecalho(): Cria um painel de cabeçalho com sub-painéis para logo, ícones de redes sociais e barra de pesquisa.
    - createPainelLogo(): Cria um painel para o logo da aplicação.
    - createPainelRedesSociais(): Cria um painel para ícones de redes sociais.
    - createBarraDePesquisa(): Cria um painel para a barra de pesquisa com um campo de texto e um botão.
    - createPainelCentral(): Cria um painel central com menus horizontais, imagem e título do usuário, e biblioteca de álbuns.
    - createMenuHorizontalSuperior(): Cria um painel para o menu horizontal superior com botões como "Discover", "Lists", "Genres", etc.
    - procurarAlbum(ArrayList<Album> albums, String nome): Procura um álbum pelo nome e retorna o álbum ou lança uma exceção se não encontrado.
    - createMenuHorizontalInferior(): Cria um painel para o menu horizontal inferior com botões como "Ratings", "Reviews", "Likes", etc. Define ação para o botão "MAKE A REVIEW" abrir a página de revisão.

- createPainelImagemTitulo(): Cria um painel para imagem e título do usuário.
- createImagemPerfil(): Carrega a imagem de perfil do usuário a partir de uma URL e retorna um JLabel com a imagem.
- createLogo(): Carrega o logo da aplicação e retorna um JLabel com a imagem.
- createBiblioteca(): Cria um painel para a biblioteca de álbuns com título e imagens dos álbuns.
- createImage(Album album): Carrega a imagem da capa do álbum a partir de uma URL e retorna um ImageIcon.
- createPainelImagens(): Cria um painel para as imagens de álbuns, adicionando botões com as imagens e ações de clique.

- **Classe PaginaReview**
  - criarReviewAlbum(): Cria um objeto ReviewAlbum com usuário e álbum. Configura a nota e o texto da review. Define a coesão e retorna o objeto ReviewAlbum.
  - initComponents(): Configurações gerais da janela: título, tamanho, cor de fundo, operação de fechamento e layout. Adiciona o painel de cabeçalho e o painel central. Centraliza a janela.
  - createPainelCabecalho(): Cria um painel de cabeçalho com o logo da aplicação.
  - createPainelLogo(): Cria um painel para o logo da aplicação.
  - createLogo(): Carrega o logo da aplicação e retorna um JLabel com a imagem.
  - createPainelCentral(): Cria um painel central com um painel de imagem e um painel de texto.
  - createImagePanel(): Cria um painel para exibir a imagem do álbum e seu título. Configura a fonte e cor do título.
  - createImage(): Carrega a imagem da capa do álbum a partir de uma URL e retorna um JLabel com a imagem.
  - createTextPanel(): Cria um painel para os campos de texto onde o usuário pode inserir a nota, coesão e a review. Adiciona os campos de texto, labels e um botão de salvar.
  - saveText(): Obtém os valores inseridos nos campos de texto. Cria um objeto ReviewAlbum e publica a review. Fecha a janela atual e abre a PaginaUsuario.
- **Classe Cadastro**
  - initComponents(): Configura a janela (título, tamanho, operação de fechamento). Cria e configura um painel com layout GridBagLayout. Adiciona labels, campos de texto e botão ao painel. Define cores e adiciona um ouvinte de ação ao botão para salvar dados e criar um objeto Usuario.
  - createPainelLogo(): Cria um painel para o logo da aplicação.
  - createLogo(): Carrega e redimensiona o logo da aplicação, retornando um JLabel com a imagem.

- **Classe Application**
    - main(String[] args): Carrega os álbuns e adiciona as faixas a partir de arquivos JSON. Inicializa a interface gráfica e chama o método para exibir a tela de cadastro.
    - addAlbunsJson(): Cria um objeto Artista (Taylor Swift). Lê o conteúdo do arquivo albums.json. Converte o conteúdo JSON em uma lista de álbuns associados ao artista. Retorna a lista de álbuns.
    - addTracksJson(ArrayList<Album> albums): Lê o conteúdo do arquivo tracks.json. Converte o conteúdo JSON em faixas e as adiciona aos álbuns fornecidos.

Mais detalhes sobre as classes e seus respectivos atributos estão abaixo, note que relacionamentos estão indicados e estruturas hierárquicas também estão destacadas no diagrama UML:



Essas classes e o enum foram pensadas de modo a abordar os pontos chaves do universo da indústria da música.

FUNCIONALIDADES

A versão 1 do Echo atualmente permite:

- **Cadastro de Usuário:** via interface gráfica é possível, cadastrar um novo objeto da classe Usuario, adicionando todos seus atributos;
- **Visualização dos Álbuns Cadastrados:** por ser uma plataforma de publicação de avaliações, e não de obras em si, não é possível cadastrar novos álbuns e músicas. Mas é possível visualizá-los e escolher qual deles o usuário gostaria de tomar como fonte de opinião. Atualmente, retiramos as informações da discografia da

cantora e compositora Taylor Swift de um *dataset*[3] em JSON, cujos dados foram retirados da própria API do Spotify por um script em Python, algo que despertou interesse do grupo além do projeto.

- **Escrita de Resenha de Álbum e/ou Música:** a principal funcionalidade do projeto também está disponível. Ao escrever uma *review*, o usuário automaticamente a tem conectada ao seu perfil, onde pode acessá-la mais tarde, caso queira. Além disso, as informações da resenha feita ficam registradas em um artigo .txt.

Nessa versão, o grupo decidiu alterar a proposta do Echo para um "diário musical" englobando a discografia da cantora Taylor Swift. Dessa forma, no atual momento, não é possível haver integração entre usuários, apenas o uso individual. Tal interação fugiria muito do escopo do projeto agora, de modo que será necessário uma fase de estudo mais aprofundada para essa implementação, a qual pretendemos prosseguir após essa entrega.

INTERFACE GRÁFICA

A interface gráfica do Echo foi feita utilizando Java Swing Jframe. Nessa primeira versão, para a interface gráfica foram feitas três janelas.

A primeira janela é a janela de cadastro do usuário. Nela, é solicitado ao usuário que adicione suas informações para a criação de seu perfil. Tais informações são nome de usuário, cpf, email, senha e url da foto de perfil. Além disso, foi adicionado um botão "Cadastrar-me", que quando o usuário clica, suas informações são armazenadas e uma caixa de mensagem aparece com o texto "Informações salvas!". Ao clicar em "Ok", o usuário é direcionado para a segunda janela.

A segunda janela é a página do usuário. Nela aparece a foto de perfil, o nome e a biblioteca do usuário. Quando o usuário clica no botão "Fazer Review" uma inputbox aparece na tela solicitando o nome do álbum que o usuário deseja avaliar. Tratamento de exceção foi implementado caso o usuário não digite nenhum nome e caso ele digite o nome de um álbum que não está no catálogo do Echo. Caso não ocorra nenhuma exceção, o usuário é direcionado para a terceira janela.

A terceira janela é a página de review de um álbum. Nela aparecem o título e a capa do álbum que está sendo avaliado. Além disso, apresenta campos para o usuário digitar a nota geral que avalia o álbum, a nota de coesão e também uma review escrita sobre a obra. Por fim, ao clicar no botão "SALVAR", as informações digitadas pelo usuário são salvas e o usuário é direcionado de volta para a sua página.

[3] uma coleção de dados usada para análise e modelagem, normalmente organizada em um formato estruturado, como uma planilha Excel, um .csv ou um arquivo JSON

DATASET

Para esta entrega do Echo, optamos por criar um *dataset* dos álbuns e músicas da cantora Taylor Swift. Para isso, o grupo consumiu a API do Spotify utilizando um *script*[4] feito pelas integrantes em python. Com os arquivos "tracks.json" e "albums.json", foi possível instanciar facilmente os objetos necessários para o projeto. Nesse processo, o grupo utilizou as bibliotecas jackson-core e jackson-databind, facilitando a manipulação dos dados armazenados em JSON para um vetor de objetos Java, cuja informação realmente poderia ser utilizada dentro do projeto.

Vale destacar que, para esse processo ser funcional, foi necessário refatorar a estrutura do Echo para um projeto Maven, adicionando um arquivo "pom.xml" onde estão as dependências do código. Para instalá-las em seu computador, é muito simples, basta baixar o Maven, através das instruções desse link, depois, basta rodar o comando *mvn clean install* na raiz do projeto.

CONCLUSÃO

Nesse protótipo do Echo, o usuário consegue ter uma interação facilitada através de uma interface gráfica, podendo entrar na aplicação e logar reviews de quaisquer conteúdos da discografia da cantora Taylor Swift. Além disso, foi feito o tratamento de exceções, de modo a permitir um uso mais robusto da aplicação. Esta versão, apesar de fornecer as principais funcionalidades, ainda está longe do que o time pretende alcançar com a implementação da interação entre usuários, possibilitando a adição de amigos, ver resenhas feitas por outros perfis, às curtir e comentar.

[4] uma série de instruções programadas para realizar determinada tarefa em um computador