PlayED

Trabalho em dupla apresentado na disciplina de Estrutura de Dados no período de 2024/01 para obtenção de nota parcial.

Alunas: Bárbara Alencar e Maria Eduarda Noia

Introdução

O presente trabalho emula um serviço de streaming de músicas fictício chamado PlayED, cujo principal objetivo objeto é organizar as músicas dos usuários em playlists e montar redes de amizades e de compartilhamento de músicas entre os usuários do streaming. O desafio deste trabalho consiste em aplicar os conhecimentos adquiridos na disciplina de Estruturas de Dados para construir esse pequeno sistema e resolver o problema proposto.

O trabalho pode ser definido em três etapas principais: a primeira é a construção do sistema de conexões entre músicas, playlists e usuários; a segunda é a realização da refatoração de playlists, que nesta etapa deixam de ser divididas por gênero e passam a ser divididas por artistas; e a terceira etapa é o *merge,* onde são localizadas as músicas similares dentro de uma rede de amigos e fundidas numa mesma playlist.

Em um panorama geral, a construção do PlayED aqui apresentada foi feita utilizando listas encadeadas. Essas listas foram usadas para conectar os usuários na rede de amigos, para conectar as playlists de um usuário e para conectar as músicas dentro de uma mesma playlist, tudo feito de forma sequencial.

Essas diferentes listas estão organizadas em uma hierarquia, onde a lista de músicas está contida dentro da lista de playlists, a lista de playlists está contida dentro da lista de usuários, e a lista de usuários está contida dentro do sistema.

Para a implementação dessas listas, optou-se por não utilizar um formato de listas genéricas pelo nível mais elevado de implementação e uma maior propensão ao erro. Portanto, não existe uma biblioteca específica para as funções de lista dentro dos arquivos do trabalho - foram feitas as funções de manipulação de lista específica para cada tipo definido, que se encontram dentro da biblioteca com o nome de cada tipo. Por exemplo, a função que retira uma música de uma lista de músicas é diferente da função que retira uma playlist de uma lista de playlists, apesar da tarefa ser a mesma. Essa foi uma decisão tomada no início do projeto que guiou toda a implementação.

Em suma, cada tipo definido no trabalho tem as suas próprias funções de lista, e feito isso as tarefas conseguintes se resumiram a organizar essas funções para construir os algoritmos de refatoração, de busca de similaridades e de *merge*, que serão detalhados mais à frente.

Implementação

Para gerenciar essas listas e os objetos contidos nelas, foram criadas as principais estruturas:

- Musica, que armazena as informações das músicas, como nome da música e do artista
- ListaDeMusica, que é o sentinela da lista de músicas, que armazena a localização da primeira e última música.
- Playlist, que armazena o nome da playlist e a lista de música dentro dela.
- ListaDePlaylist, que armazena determinadas playlists dentro de uma lista.
- ListaDeusuario, que armazena uma sequência de usuários.
- Usuario, que armazena o nome do usuário, a lista de playlists dele e a lista de amigos (usuários) dentro dele dentro dele.
- PlayED, que armazena uma lista com todos os usuários do programa, e todas as playlists.

As funções de listas definidas para cada tipo num geral englobam: cria lista, insere na lista, retira da lista, retorna próximo da lista e busca na lista. Devido a hierarquia de estruturas construídas juntamente ao também ao fato de existirem funções de lista para cada tipo diferente, existem funções como “retiraMusicaDaPlaylist(playlist)” que possuem a função apenas de chamar a função “retiraMusicaDaListaDeMusica(playlist->listaDeMusica)” para propiciar o encapsulamento das funções e abstração de alguns conceitos mais básicos no topo da hierarquia.

As principais funcionalidades do programa estão na biblioteca do PlayED, sendo elas:

- Inicialização e finalização do programa

Engloba as funções: criaPlayED, leListaDeUsuario e lePlaylistsUsuarios. Essas funções fazem primeiramente a alocação de memória necessária para armazenar as estruturas. As funções de leitura recolhem os dados dos arquivos e os colocam dentro de suas respectivas estruturas alocadas. A função leListaDeUsuario faz isso de forma mais simples, devido aos usuários estarem localizados de forma mais

externa no programa. Já o leListaDePlaylist precisa acessar cada usuário individualmente, criar e inserir essas playlists dentro deles.

A finalização do programa é feito através da função liberaPlayED, que chama dentro de si as funções destroiListaEMusicas, destroiListaEPlaylists e destroiListaEUsuarios. Essas funções liberam a memória não só da lista mas também das estruturas alocadas para Música, Playlist e Usuário, que estão armazenadas na estrutura PlayED. Cabe aqui pontuar a diferença entre essas funções para aquelas similares que são encontradas dentro do programa: destroiListaDePlaylist, destroiListaDeMusica e destroiListaDeUsuario. Essas últimas, fazem a liberação somente das listas, e não das estruturas, sendo usadas principalmente na etapa de refatoração das playlists, onde é necessário quebrar vínculos de lista e manter o conteúdo original salvo.

- Refatoração das playlists

A refatoração é feita através da função refatoraPlayED e funciona segundo o esquema da figura 1 da lista de figuras, e gera os arquivos de saída através da função imprimePlayEDRefatorada.

- Busca de similaridades entre as playlists dos amigos

O algoritmo de busca de similaridade entre as playlists dos amigos é feito somente através da função imprimeSimilaridades, onde a realização da busca de similaridades e a impressão do documento é feita em conjunto. Essa função tem algumas peculiaridades: foi necessário criar uma flag e adicioná-la ao usuário para que não ocorra a impressão repetida de dois amigos, por exemplo “João;Maria;4” e “Maria;João;4”. Toda vez que o usuário passa pela verificação de similaridade, ele possui a sua flag marcada similaridade como 1. Então, nos loops internos do algoritmo, quando a flag do usuário é igual a 1, ele não é impresso novamente. A contagem de similaridades entre músicas dos usuários é feita através de um contador, chamado similaridades (no plural). O algoritmo de busca de similaridades está esquematizado na figura 2 da lista de figuras.

- Merge de playlists

O algoritmo de merge de playlists é feito de forma similar aos anteriores, feito pelas funções mergePlayED e imprimePlayEDMerged. O algoritmo de merge de playlists está esquematizado na figura 3 da lista de figuras.

Conclusão

Uma das dificuldades encontradas durante a realização do projeto foi a quantidade de listas implementadas, em oposição ao uso de apenas uma lista genérica. Tivemos que implementar diversas funções, muitas vezes análogas, o que não seria necessário no outro caso. Por outro lado, a diversidade de funções permitiu que o código fosse mais fácil de compreender devido à maior especificidade.

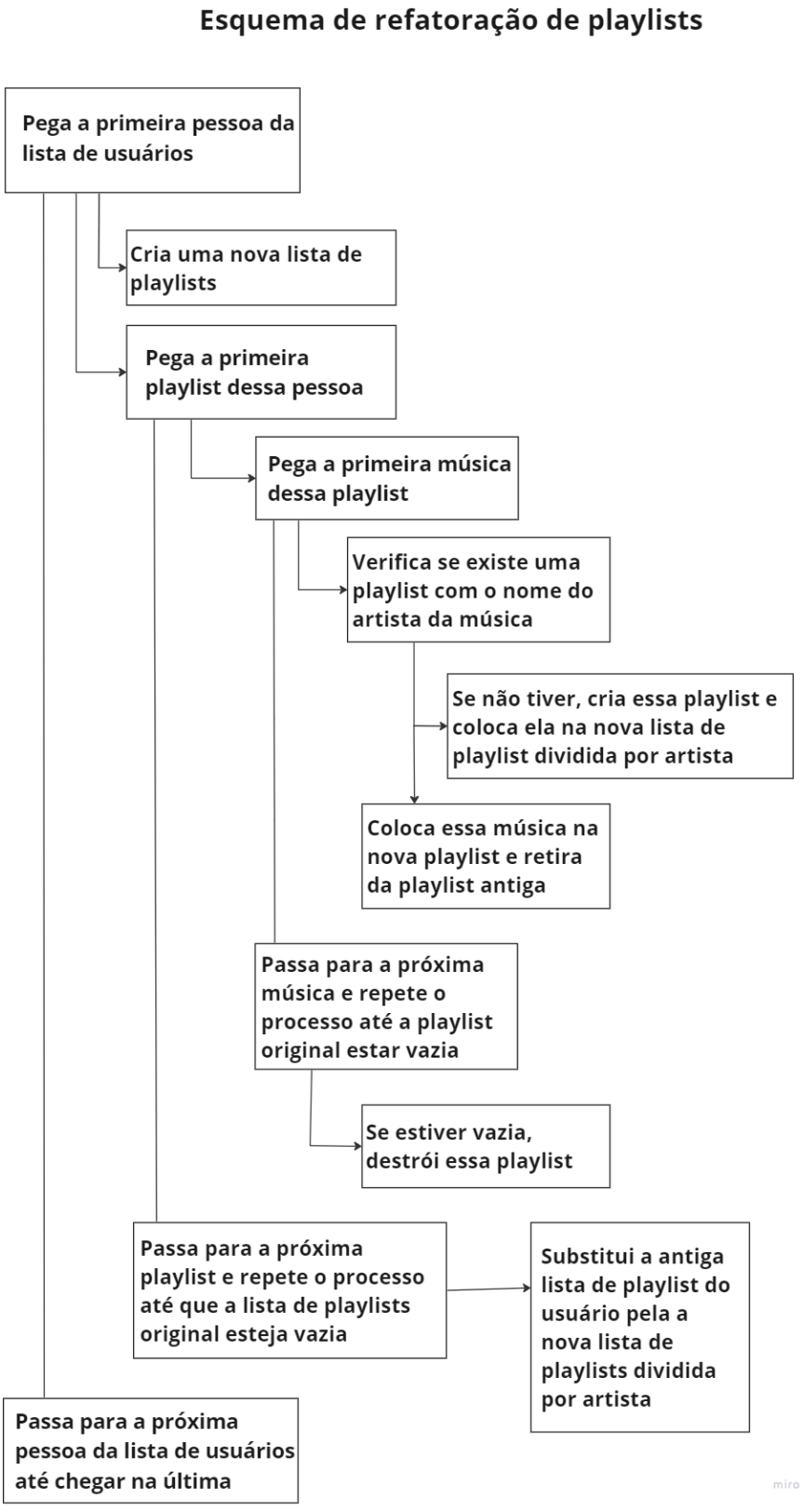
Os algoritmos para Refatoração, Merge e Similaridade também exigiram bastante planejamento e depuração. Por iterarem em muitas listas e acessarem diversas camadas, essas foram certamente as partes mais desafiadoras em termos de lógica e organização

Bibliografia

Não foi utilizado nenhuma bibliografia externa neste trabalho além dos conhecimentos passados em sala de aula e dos materiais disponibilizados pela professora da disciplina.

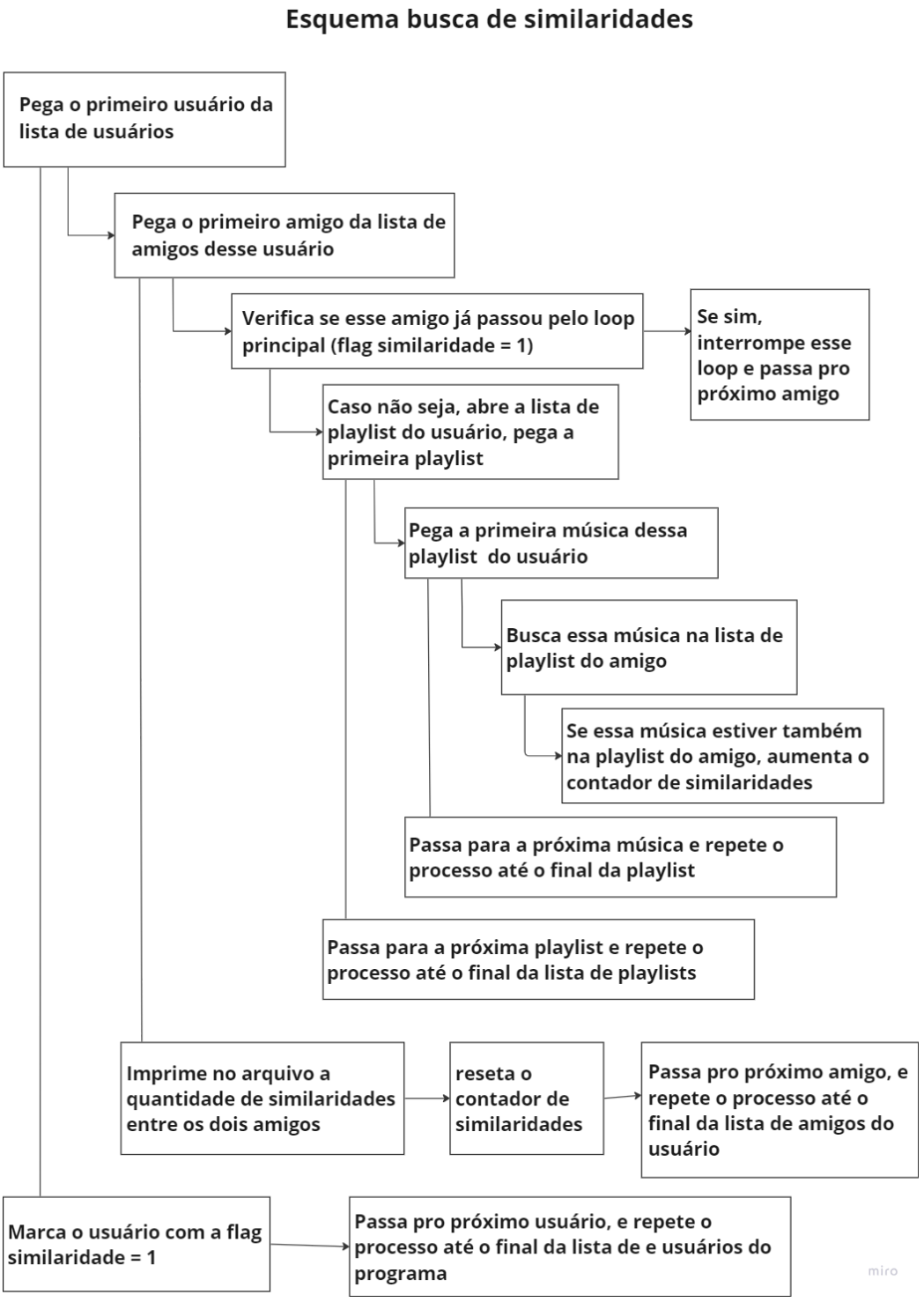
Lista de figuras

Figura 1: Esquema de refatoração de playlists



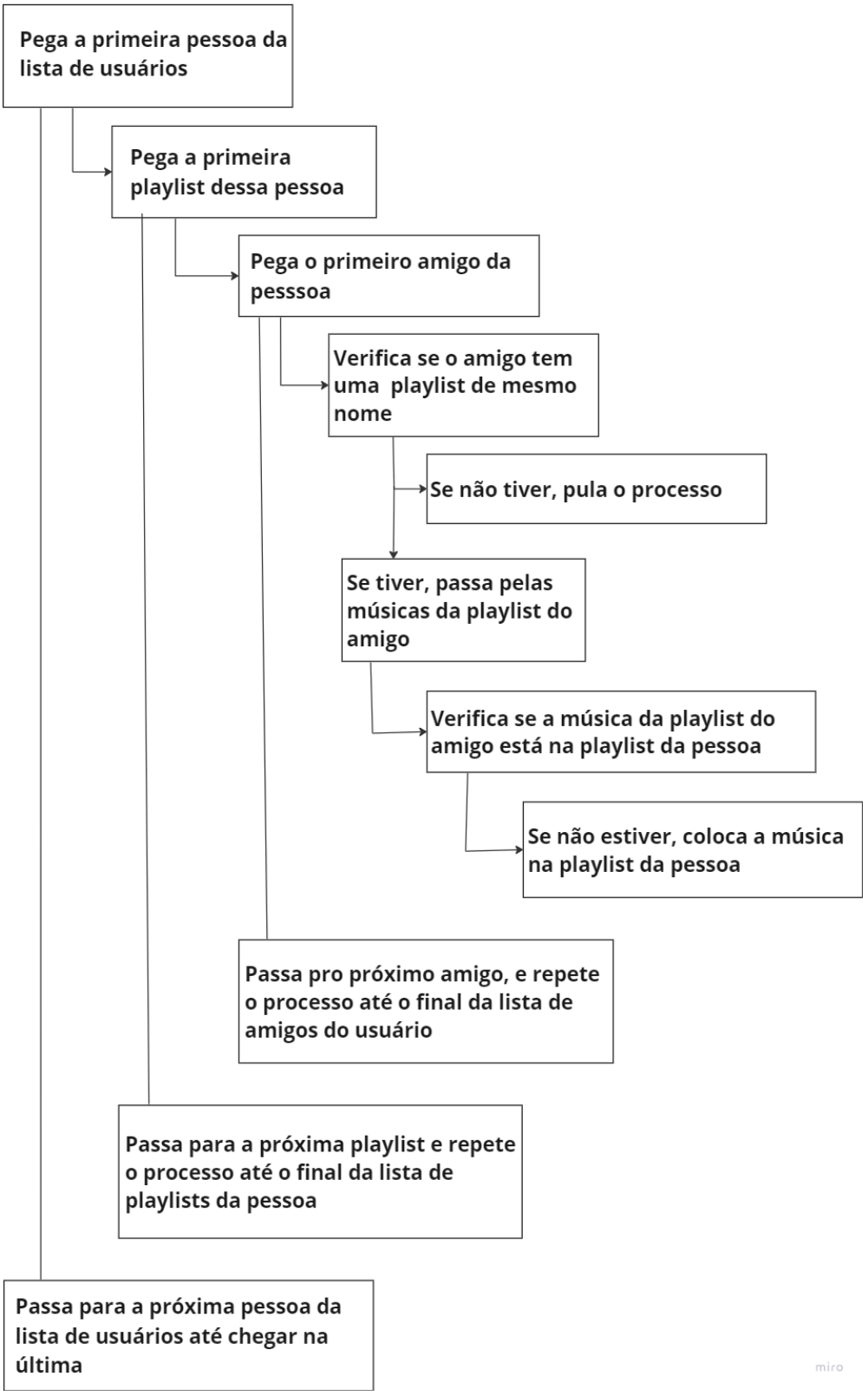
Fonte: arquivo dos autores

Figura 2: esquema de busca de similaridades



Fonte: arquivo dos autores

Figura 3: esquema de merge de playlists



Fonte: arquivo dos autores