UFES

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CAMPUS GOIABEIRAS

CIÊNCIA DA COMPUTAÇÃO

ESTRUTURAS DE DADOS I

Relatório sobre o desenvolvimento do programa PlayED

Bruno Menegaz Acerbi
Matheus Adami Bernardes Moreira

VITÓRIA-ES 2021

Introdução

Tomando como início a popularização dos aplicativos de streaming musical e suas principais funcionalidades, foi proposta a aplicação dos conhecimentos passados sobre a manipulação de estruturas de dados encadeadas em um ambiente similar. A partir de uma coletânea de informações práticas dos usuários, propôs-se elaborar um programa capaz de armazenar tais informações de forma eficiente para que, em sequência, seja possível realizar alterações e análises de interesse, tal programa foi elaborado a partir da linguagem estruturada C.

O trabalho iniciava-se com o objetivo de estruturalizar uma série de dados externos de clientes do nosso aplicativo, essas informações eram fornecidas por meio de arquivos de entrada contendo os nomes dos clientes, as suas relações de amizades, a quantidade de playlists, suas intitulações e as informações das músicas que as construíram (nome da banda e da própria música), após essa leitura era necessário registrar essas informações dentro de diferentes listas elaboradas para cada tipo de dado, possibilitando o acesso e as alterações que foram exigidas em sequência.

Após a construção de toda a estrutura de dados, foi proposto a refatoração a julgar pelas informações armazenadas, uma vez que fez-se necessário recriar as listas de playlists de cada usuário com base em cada compositor contido nas listas antigas, ou seja, cada autor deveria ter sua própria playlist com as suas respectivas músicas, para que assim tal conjunto pudesse substituir as playlists antigas do cliente.

Em sequência, pede-se que se faça um cálculo das similaridades dos gostos musicais entre os usuários que apresentam uma relação de amizade dentro do ambiente do playED.

Ao término dessa refatoração e desse cálculo, a nova relação de dados deve ser exibida em diferentes arquivos de texto, um deles armazena a quantidade de similaridades musicais entre determinados amigos, já outro é responsável por conter a nova quantidade de playlists que cada cliente possui seguida de seus respectivos nomes. Por fim deve ser criado diretórios para cada usuário, sendo estes responsáveis por armazenar arquivos de textos com as respectivas listas musicais de cada playlist do cliente em questão.

Implementação

O projeto iniciou-se, de fato, após a preparação do ambiente de desenvolvimento. Houve uma reunião onde definiram-se as ferramentas que seriam utilizadas, como a IDE e o plugin de colaboração (para que ambos pudessem ver e editar os arquivos da aplicação de forma simultânea). Além disso, foi nesta etapa que decidiu-se pela utilização de um controlador de versões (Github) para que houvesse uma maior organização nas mudanças e comentários que viriam a ser adicionados ao decorrer do projeto.

A segunda etapa consistiu em observar o enunciado, que foi passado no classroom, e elaborar o mapeamento do projeto. Ao analisar as necessidades do aplicativo, contidas no escopo, tomamos a decisão de separar o projeto em diversas pastas para facilitar a organização, em sequência decidimos por utilizar um makefile previamente elaborado na disciplina de Tópicos Especiais em Programação, o qual possibilita a compilação dos arquivos em seus respectivos diretórios. Além disso, optamos por trabalhar com as estruturas de listas simples com sentinela, pois percebemos que a necessidade de adicionar novos elementos ocorreria sempre ao final da lista. Dando continuidade, iniciou-se a fase de estruturação das relações de subordinação das respectivas listas para com seus dados existentes no projeto, com o objetivo de clarear essa etapa, decidimos por elaborar o seguinte diagrama (imagem 1) que ilustra a ideia geral do programa. Nele podemos ver a existência de uma lista de usuários, onde cada usuário (cliente) aponta para as suas respectivas listas de playlists e amigos. Em sequência, a lista de amigos aponta para um usuário o qual nosso cliente primordial possui uma relação de amizade. Agora em relação a lista de playlists, temos que cada elemento dessa estrutura deve apontar para a sua respectiva lista de músicas o que finaliza o mapeamento geral da estruturação do programa.

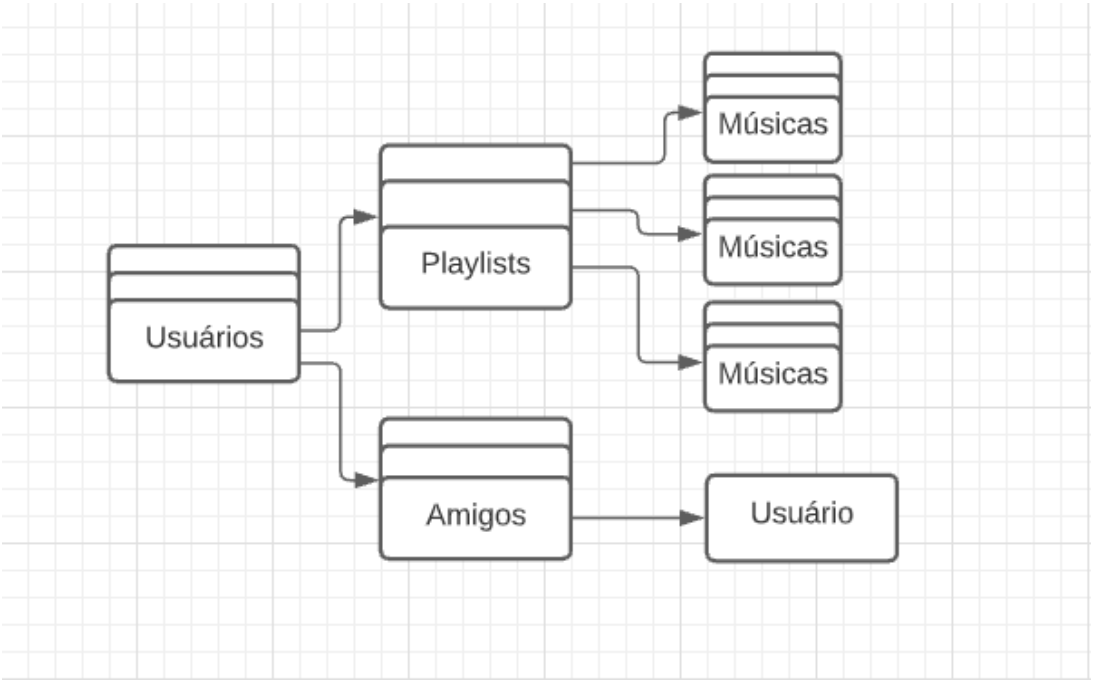

Imagem 1 - Diagrama da estruturação dos dados

Após alinhar o projeto, o próximo passo foi iniciar com a programação em si. O ponto de partida foi a criação dos arquivos de TAD's que julgamos serem necessários para prosseguirmos com o desenvolvimento do projeto. Os primeiros a serem criados foram tMusica, tPlaylist e tUsuario, ambos com estruturas bastante similares, onde colocamos as funções que lidariam com: inicialização, preenchimento, manipulação, retorno e liberação de cada uma das estruturas simples e seus respectivos dados. Vale destacar que, devido à necessidade de algumas dessas estruturas (tPlaylist e tUsuario) conterem estruturas de listas dentro delas, foram adicionadas sentinelas com a tipagem de suas respectivas estruturas de interesse.

Em sequência, houve a criação das estruturas que lidariam com as listas encadeadas. Para isso, foram criados os TAD's: tListaAmizade, tListaUsuario, tListaPlaylist e tListaMusica. Estes, como dito anteriormente, foram feitos para lidar com listas simplesmente encadeadas com sentinela. Dentre as funções recorrentes nestes TAD's, pode-se destacar as que lidam com: inicialização da sentinela, criação da lista, inserção de um elemento na lista e liberação da lista e de seus elementos. É válido destacar que, assim como nos TAD's anteriores, nestas estruturas repetiu-se a ideia de similaridade entre algumas das funções contidas nos arquivos .C e .H, pois todos estes, apesar de abordarem propósitos diferentes, seguem uma lógica parecida.

Por outro lado, as diferenças nos arquivos de listas começaram a ocorrer com o desenvolvimento de algumas funções que não seriam genéricas, ou seja, que possuíam finalidades específicas para o escopo deste projeto. Nesse sentido, pode-se dizer que destacaram-se as funções que lidam com a leitura dos arquivos de entrada, a refatoração das listas e as verificações de similaridades. De tal forma que, inicialmente, precisou-se armazenar os dados de input dentro da estrutura já pré estabelecida, tal armazenamento foi possível com a devida busca dos dados em seus respectivos arquivos, busca essa que exigia um grau de cuidado para que o registro ocorresse da forma apropriada. Em sequência, as funções de refatoração (presentes em tListaMusica, tListaPlaylist e tListaUsuario) tiveram por propósito utilizar os dados de input, já armazenados, para lidar com a lógica necessária para preencher as novas listas com o padrão passado no enunciado do trabalho. Já as funções de similaridade, (em tListaPlaylist, tListaMusica e tListaAmizade) basicamente possuem uma lógica de percorrer as suas respectivas listas calculando o quão parecidas são duas listas de músicas, playlists, ou em última instância, o quão parecidos são dois amigos quando trata-se do gosto musical em questão.

Por fim, foram criadas as funções necessárias para a construção das saídas, que no geral apresentam como propósito o de exibir os resultados da refatoração e dos cálculos das similaridades. Elas por sua vez progridem de uma maneira bem linear, ou seja, a partir da lista de usuários, foi-se criado um diretório para esse cliente utilizando da funcionalidade "mkdir" que logo em sequência é preenchido com as respectivas playlists pós refatoração deste usuário. Da mesma forma é exibido à medida que a lista de usuários proguide, a quantidade de playlists com seus respectivos nomes e as similaridade dos gostos musicais, porém nesse caso vale destacar a preocupação para a não repetição de similaridades já previamente calculadas e exibidas no arquivo de saída, para isso realizamos uma verificação pré cálculo que procura os nomes dos usuários, e caso os encontre, o programa avança para o próximo da lista de amigos, evitando assim a reincidência.

Conclusão

A fim de simular a lógica dos algoritmos contidos “por baixo dos panos” dos grandes aplicativos de streaming, o projeto, intitulado “PlayED”, cumpre muito bem o seu papel, visto que aplica de forma clara a necessidade da utilização de estruturas encadeadas para o registro de maneira proveitosa das informaçõe de cada cliente e põem em prática a realidade de dados sendo agrupados de forma encapsulada, uma vez que se torna perceptível as relações de dependência entre os mesmos.

Abordando agora as dificuldade enfrentadas, a primeira a ser observada foi o nível de encapsulamento das informações, uma vez que existe nesse algoritmo a incidência de listas englobadas por outras listas, provocando assim o surgimento de tad’s internamente dependentes a outras estruturas, tal complexidade acabaria por resultar em alguns erros caracterizados por loopings de inclusão que graças a colaboração da monitoria, puderam ser resolvidos com uma simples mudança no local de definição das estruturas.

A contraponto, algo que facilitou bastante o desenvolvimento do código foi a similaridade entre as estruturas encadeadas, uma vez que todas seguiam o padrão de listas simplesmente encadeadas com sentinela, o que por sua vez resultou na aplicação de funções semelhantes. Vale destacar que ao lidarmos com tal característica, uma implementação de listas genéricas mostra-se mais satisfatória, pois evita a repetição de códigos. Todavia, ao desenrolar do desenvolvimento percebemos que não haveria a necessidade de colocar em prática esta implementação, tal ideia poderá vir a ser realizada no futuro com o estudo deste tema.

Referências

1. https://devdocs.io/c/
2. http://www.inf.ufes.br/~pdcosta/ensino/2012-1-estruturas-de-dados/slides/Aula9%28listas%29.pdf
3. http://www.inf.ufes.br/~pdcosta/ensino/2012-1-estruturas-de-dados/slides/Aula10_2%28listas3%29.pdf
4. https://pt.stackoverflow.com/
5. https://www.ime.usp.br/~pf/algoritmos/aulas/lista.html#:~:text=Uma%20lista%20encadeada%20%C3%A9%20uma,segunda%2C%20e%20assim%20por%20diante.
6. https://docs.microsoft.com/pt-br/cpp/c-language/?view=msvc-160