UNIVERSIDADE D
COIMBRA

# Introdução À Programação em Java

# Sistema de gestão de conteúdo musical Rockstar Inc.

Tiago Sousa Caniceiro

Pedro Monteiro

Janeiro 2023

# Índice

# Introdução

Neste relatório, apresentado no âmbito da disciplina de introdução a Java do curso Acertar o Rumo da Universidade de Coimbra detalhamos o processo de desenvolvimento de um projeto que consiste no desenvolvimento de uma aplicação "standalone" direcionada a vários tipos de utilizador. Para além das competências em linguagem Java o projeto tem como objetivo a aplicação de conhecimentos em interface gráfica e armazenamento de dados.

Este documento aborda a criação de uma aplicação feita para a empresa fictícia Rockstar Inc. onde são incluídos a análise dos requisitos, uma visão geral da estrutura do sistema, a descrição de classes e métodos essenciais com diagrama de classes. Também é apresentada uma discussão dos desafios enfrentados e soluções encontradas e uma reflexão sobre as competências adquiridas durante o projeto.

# Descrição do problema e Análise de requisitos

O projeto pedido visa o desenvolvimento de uma aplicação que simula um ambiente de gestão de conteúdo musical. A versão desenvolvida neste trabalho foi a correspondente à de dois alunos sendo que o programa poderá ser acedido por dois tipos diferentes de utilizador: o utilizador Cliente (Client) e o utilizador músico (MusicCreator).

Neste sentido são desenvolvidas duas interfaces gráficas distintas para cada utilizador em que cada uma terá apenas as funcionalidades únicas de cada utilizador. As principais funções exigidas a um músico são a criação de novas músicas e álbuns, assim como a gestão e edição destas no sistema. Da mesma forma as principais funções de um cliente são a de comprar músicas e gerir playlists.

Os dois tipos de utilizador têm algumas funções partilhadas como a pesquisa, filtragem e ordenação de músicas, login e registo que são desenvolvidos em comum e posteriormente adaptados a cada utilizador.

Outros aspetos importantes no projeto foram a preocupação na criação de uma interface gráfica fácil de usar e intuitiva e de um sistema de armazenamento e recuperação de dados que atenda ao pedido para este projeto e que seja eficiente na utilização prática da aplicação.

# Estrutura da aplicação

## Aspetos gráficos:

Dados os requisitos do projeto foi feita a opção de dividir a estrutura gráfica em três elementos principias:

O primeiro elemento trata de todas as funcionalidades de registo e login. Este elemento é composto por três janelas que são apresentadas ao utilizador como diferentes opções para registo ou login tanto de cliente ou músicos.

O segundo elemento corresponde à janela que apresenta todas as funcionalidades a um cliente depois de um login bem sucedido e a terceira janela corresponde às funcionalidades atribuídas ao músico.

Este projeto foi desenvolvido com recurso à biblioteca gráfica Java Swing e nas primeiras fases do desenvolvimento foi entendido que para oferecer um ambiente gráfico intuitivo, fácil e ao qual o utilizador já estivesse habituado de outras aplicações. Desta forma, as primeiras fases do projeto foram dedicadas ao estudo de estruturas como as JList, JTables e JPopUpMenu.

As estruturas de JList e JTable são fundamentais neste projeto para a apresentação de informação ao utilizador assim como na ordenação de resultados. O JPopupMenu é uma estrutura fundamental para executar funções com determinado objeto (como a avaliação de uma música) ou mesmo funções que relacionem dois objetos (adição de uma música a uma playlist ou álbum).

O projeto foi pensado ao nível das classes de lógica de forma a que as estruturas da janela do cliente e do músico funcionem de forma muito semelhante apenas tendo sido adaptadas as funcionalidades únicas de cada utilizador.

## Aspetos lógicos:

Neste projeto existem dois sistemas de herança: tanto o músico como o cliente como classes herdam características da classe utilizador (User) e tanto os álbuns como as playlists como classes herdam características da classe correspondente a coleção de músicas (MusicCollection). Tanto a classe User como a MusicCollection são classes abstratas.

Este sistema duplo de herança permite que na classe gestora (RockstarManager) seja possível através do utilizador que tem o login feito (quer ele seja músico ou cliente) utilizar métodos polimórficos que respondem aos requisitos apresentados. Como por exemplo: para adicionar uma música a uma playlist (responsabilidade do cliente apenas) o método chamado na classe manager é o mesmo que o método para adicionar uma música a um álbum (responsabilidade do músico apenas).

Alguns métodos foram criados apesar de não existir nenhuma menção a estes no enunciado do projeto. Um exemplo é o método de remoção de músicas de uma playlist, o objetivo seria proporcionar uma experiência mais completa ao utilizador.

A solução apresentada para a pesquisa foi a criação de uma nova classe especifica (Search). A criação de um objeto temporário de pesquisa facilita a estruturação do código e permite uma boa organização e escalabilidade caso haja essa necessidade.

Cada vez que é feita uma pesquisa com um certo termo a classe gestora é responsável por criar este objeto que levará informações em formas de listas para a interface gráfica. Neste caso o objeto da classe Search criado para um cliente será mais completo pois terá a lista de pesquisa de música por nome de música, a lista de pesquisa de música por nome de artista e a pesquisa de coleções que terá álbuns e apenas as playlists públicas. O objeto Search criado para um músico terá apenas as músicas do próprio como pedido no enunciado do projeto.

Um dos grandes desafios foi a decisão de como tratar a problemática da criação de playlists aleatórias, isto pela variedade de soluções possíveis. A solução apresentada lida com este requisito da forma mais completa encontrada e faz a seleção de músicas a partir da lista global de músicas e não da lista de músicas que o utilizador possui.

Desta forma criamos uma nova forma de aquisição de músicas caso o utilizador não possua alguma das músicas selecionada aleatoriamente. Esta opção faz com que esta funcionalidade tenha sido construída com recurso a múltiplos métodos e várias opções que poderão ser dadas ao utilizador depois da decisão de construir uma lista aleatória com base no género musical, mas que responde ao requisito pedido de todas as formas.

Fluxo da criação de uma playlist aleatória:

A partir do momento em que o utilizador opta por construir uma nova playlist aleatória com determinado número músicas de determinado género musical é feita a verificação relativamente à quantidade de músicas disponíveis naquele género. Depois disto é feita uma lista com todas as músicas daquele género disponíveis desde que ativas.

Através de um método específico é criada uma lista de índices aleatórios que corresponderão às músicas selecionadas. Nesta fase poderemos ter todas as músicas gratuitas ou previamente adquiridas pelo utilizador nesta lista. Se isto acontecer a playlist é criada. Por outro lado, no caso de haver alguma ou mais músicas que são pagas e o utilizador não possui são dadas três opções ao utilizador:

-A primeira opção criará uma playlist com as músicas selecionas que sejam gratuitas ou previamente adquiridas e enviará para o cesto de compras todas as músicas que tenham de ser pagas. Isto permite ao utilizador optar posteriormente por comprar e adicionar à playlist manualmente e individualmente. Permite também uma opção caso o utilizador não tenha dinheiro suficiente naquele momento. Esta opção levará a que o número inicial de músicas pedidas pelo utilizador para criar aquela playlist não seja atendido.

-A segunda opção apenas estará disponível se o utilizador tiver dinheiro suficiente para fazer a aquisição de todas as músicas. Optando pela compra a playlist será feita automaticamente e a aquisição realizada tal e qual como se fosse um cesto de compras.

-A terceira opção permite ao utilizador escolher apenas músicas gratuitas ou previamente adquiridas. Desta forma é criada uma nova seleção e apresentada uma mensagem caso não haja músicas suficientes naquele género que correspondam ao pedido.

Relativamente à atribuição de um rating numérico a primeira avaliação de uma música por parte de um cliente específico resulta na criação de um novo objeto avaliação. A segunda vez que o cliente fizer essa avaliação apenas estará a mudar o valor da avaliação anterior. Isto permite que não haja avaliações repetidas por parte de um cliente à mesma música. A classe música (Music) tem a responsabilidade de guardar todas as avaliações próprias e calcular a classificação com base na média dessas avaliações.

Todo o sistema de compras e adição de saldo é feito com base nas classes Music Aquisition, Music e Client. Não foi considerado necessário criar uma classe de cesto/carrinho de compras, a opção escolhida foi criar uma lista de músicas a comprar na classe Client.

# Diagrama De Classes

# Descrição de classes e métodos

## Classe SaveFileManager

Nesta classe encontramos métodos associados aos processos de serialização e desserialização. Este projeto tem como requisito a persistência de dados após o encerramento do programa através de ficheiro de objetos. Devido à dimensão do projeto e sendo este um projeto curricular a opção foi a da criação de um ficheiro de objetos único. Este ficheiro contém o resultado da serialização do objeto da classe RockstarIncManager.

Métodos:

-run(): Este método inicia o processo de carregamento do ficheiro, inicia também o processo de guardar os dados automaticamente com uma certa frequência de tempo e o processo de guardar o ficheiro no encerramento do programa.

-updateDataFile(): Este método é responsável por reescrever o ficheiro de objetos efetivando a gravação.

-loadFile(): Método responsável por carregar o ficheiro e ler o seu conteúdo. Responsável também por chamar o método updateDataFile() no caso de não ser encontrado o ficheiro. Isto vai fazer com que seja criado um ficheiro novo.

-autoSave(): Com recurso a um Timer é responsável pela gravação automática com uma certa frequência de tempo.

-saveOnshutdown(): Método responsável pela gravação quando o programa encerra.

-getGc(): Este método retorna a instância criada de RockstarIncManager.

## Classe Main

A classe Main serve como ponto de entrada do programa esta tem a responsabilidade de iniciar a gestão do ficheiro de gravação através do método SaveFileManager.run();

Obtem a instância RockstarManager criada e inicia a execução em run() da classe RockstarManager.

## Classe RockstarIncManager

Esta é a classe central da aplicação. Esta é a classe que contêm as listas de todos os utilizadores. Para além disso gerência todos os aspetos lógicos e relações entre as restantes classes e objetos que não pertencem à parte gráfica do programa. Esta é a classe responsável

por comunicar com a parte gráfica da aplicação (GUIManager). Esta classe tem também como parâmetro o utilizador atual ao qual serão aplicados os diferentes métodos (currentUser).

Métodos:

-RockstarIncManager(): o construtor é responsável por inicializar as listas principais do sistema.

-run(): Inicia o método startGUI() e permite a criação de objetos numa fase de desenvolvimento e testagem da aplicação. Foi feita a opção de manter este método mas ele não é estritamente necessário uma vez que o método startGUI() pode ser executado diretamente da classe Main.

-startGUI(): cria uma nova instância da classe GUIManager e inicializa a parte gráfica da aplicação.

-loginAttempt(): Método responsável pela verificação e validação do processo de login dos dois tipos de utilizador, tem também a responsabilidade de atualizar a componente gráfica dependendo do sucesso ou não do processo de login.

-newUserAttempt(): Método responsável por gerir e validar o processo de registo dos dois tipos de utilizador. Verifica e não permite a criação de utilizadores com o mesmo username ou email. Permite a criação de contas de diferentes tipos com username e email iguais. Isto faz com que um músico possa criar uma conta de ouvinte com o mesmo username e email e vice-versa.

-termValidationOnNewRegistration(): Este é um método auxiliar do método de registo newUserAttempt() e permite a validação de todos os termos inseridos pelo utilizador numa tentativa de registo.

-search(): Este método tem a responsabilidade de criação de objetos temporários da classe Search. Este método cria várias listas de pesquisa tendo em conta um termo pedido pelo utilizador. O método entrega o resultado da pesquisa para utilizadores clientes e para utilizadores músicos.

-newRandomPlaylistAttempt(): este método inicializa o processo de criação de uma nova playlist aleatória. Neste método é criada a lista de todas as músicas ativas e de um certo género disponíveis no sistema. De seguida é avaliada a possibilidade de criação comparando o número de músicas pedidas pelo utilizador com a quantidade de músicas encontradas. Caso não seja possível é enviada uma mensagem ao utilizador com o número de músicas disponíveis para o género pretendido. No caso de ser possível fazer uma playlist aleatória é chamado o método responsável pela criação da playlist randomPlaylistCreation().

-randomPlaylistCreation(): Este método é responsável por chamar o método de seleção de músicas para a playlist randomMusicSelectio() que fornece duas listas (a lista de musicas gratuitas ou adquiridas) e a lista de músicas que têm de ser adquiridas.

Este método cria automaticamente a playlist aleatória no caso de todas as músicas selecionadas sejam gratuitas ou o utilizador já as tenha adquirido. No caso de haver pelo menos uma música selecionada que o utilizador possua ou não seja gratuita é iniciado um novo método que trata desta situação processorOnRandomToPayMusic();

-randomMusicSelection(): Este método realiza a seleção de músicas de forma aleatória. Para este efeito é utilizado um método acessório randomIndexVector() que cria uma lista de índices de forma aleatória que corresponderá aos índices da lista de músicas do género selecionado anteriormente criada. Todas as músicas pagas e não adquiridas pelo utilizador irão para uma segunda lista.

-processorOnRandomToPayMusic(): Este é o método responsável por lidar com a situação de criação de uma nova lista aleatória tendo sido selecionadas músicas pagas e não adquiridas pelo utilizador. Este método chama um método acessório que calcula o preço de uma lista de músicas para depois apresentar a opção de compra ao utilizador. O método é responsável pela comunicação com a parte gráfica e envia informações como a lista de músicas a serem adquiridas para a constituição da playlist, os respetivos preços e um booleano que confirma a possibilidade do utilizador adquirir as músicas. Tendo em conta a resposta do utilizador o método utiliza um "switch" de forma a enviar essas músicas a adquirir para a cesto de compras, permite e executa a compra de todas as músicas automaticamente ou redireciona para um novo método de criação de playlists aleatórias com musicas adquiridas ou gratuitas newRandomPlaylistOnlyFree().

-musicListPriceCalculator(): método que calcula o preço de uma lista de músicas.

-newRandomPLaylistOnlyFree(): método que reinicializa o processo de criação de playlist aleatória pois apenas são selecionadas músicas adquiridas ou gratuitas do género pedido.

-randomIndexVector(): método que cia uma lista de índices de forma aleatória.

-newMusic(): método que lida com a validação de termo para nome de música e para o preço de uma música e que tem a responsabilidade de criar uma nova música no sistema. Isto consiste na adição a lista de músicas do sistema, adição à lista de músicas do respetivo utilizador e envio de mensagem ao utilizador através da classe GUIManager.

-musicEditionAttempt(): método que lida com a validação dos parâmetros de uma música, efetua a lógica de edição e comunica com o gestor gráfico com base no sucesso da tarefa.

-musicNameValidation(): método responsável pela validação do nome de uma música.

-musicPriceValidation(): método responsável pela validação do preço de uma música.

-logout(): método que iguala o "currentUser" a "null "em caso de "logout".

-Conjunto de métodos getters: corresponde a um conjunto de métodos que permitem levar informações simples à classe gráfica ou a outros métodos: getCurrentUserAllMusic(), getCurrentUserAllCollections(), getCurrentUserBalance, getUserBasketList. Os métodos getClientAllMusicAsCollection() e getMusicCreatorAllMusicAsCollection() retornam um novo objeto Playlist ou Álbum que permitem formar na interface gráfica o primeiro elemento na lista de playlists ou álbuns. De notar que não corresponde a uma playlist ou álbum conceptualmente. São apenas objetos criados de forma temporária de forma a apresentar todas as músicas adquiridas no caso do cliente e todas as músicas criadas no caso do músico na interface gráfica.

-Conjunto de métodos polimórficos: este conjunto de métodos chama os métodos polimórficos do sistema de classes com herança User/Client/MusicCreator. Os métodos são: removeMusicFromCollection(), addMusicToCollection(), newCollection(), removeMusicCollection(), removeMusicCollection().

-Conjunto de métodos de utilizador específicos. Estes métodos lidam com funcionalidades especificas de cada tipo de utilizador havendo para isso necessidade de realizar cast do tipo de utilizador (Client ou MusicCreator): evaluateMusic(), validationOfAquisition(), addMoney(), addMusicTOMusicToBuy().

-Conjunto de métodos de estatística. A maior parte das estatísticas são obtidas através de getters ou ciclos simples: musicTotalPriceValue(), totalSongs(), totalAlbumsByGenre(), totalSalesValue(), salesCurrentUser(), currentUserTotalMusicCreated(). Os métodos finais getStatistics() e getAlbumTypeStatistics() são responsável por reunir todas estas estatísticas numa arraylist em que cada índice corresponde ao valor de uma estatística. No caso das estatísticas de álbuns o primeiro valor corresponde à totalidade de álbuns.

## Classes User/Client/MusicCreator

Estas três classes correspondem à entidade de utilizador. No sistema implementado existem dois tipos de utilizador o cliente e o músico. Desta forma, a solução apresentada consiste na construção de 3 classes: A classe User é uma classe abstrata que contem parâmetros como nome, “username” e “password” e anuncia um conjunto de métodos polimórficos utilizados nas duas classes descendentes. A classe Client tem como parâmetros adicionais o saldo, uma lista que contém todas as aquisições realizadas (como que recibos de compras), uma lista com o histórico de depósitos, e uma lista com as músicas que o cliente pretende comprar (cesto de compras). A classe MusicCreator contém os parâmetros adicionais: pin e o valor total de vendas.

Métodos:

Nestas classes temos os métodos polimórficos: newCollection(), addMusicToCollection(), newMusicToAllMusicCOllection(), removeMusicFromCollection() e removeMusicCollection().

Estes métodos permitem que o Sistema execute estas tarefas com qualquer utilizador. No caso as tarefas executadas pelo cliente estarão a ser realizadas sempre sobre uma playlist e as tarefas executadas sobre um músico sempre sobre um álbum.

-validationOfAquisition() Este é um método especifica da classe Client, onde é feita uma validação de uma eventual compra. É criado um novo objeto da classe MusicAquisition que guarda informações da compra.

-Existem nesta classe um conjunto de guetters e setters estritamente necessários ao bom funcionamento do programa.

## Classes MusicCollection/Playlist/Album

Conjunto de classes onde está presente o conceito de herança. A classe abstrata MusicCollection contêm atributos comuns como o nome, lista de músicas e data de criação. A classe Playlist tem como parâmetros únicos um booleano que permite aferir se a playlist é ou não pública. E um parâmetro que permite gravar o cliente que criou essa playlist. A classe álbum tem o parâmetro adicional que guarda o género principal que é calculado em vários momentos.

Métodos:

Nestas classes existem dois construtores, um dos construtores permite a criação de coleções de música vazias apenas com o nome, o outro construtor permite a criação de coleções com uma lista de músicas já definida.

Nestas classes existem os getters e setters estritamente necessários ao funcionamento do programa e dois métodos polimórficos: addMusicToCollection() e removeMusicFromCollection();

Na classe Álbum existe um método adicional que permite o cálculo do género musical principal de um álbum de forma a atribuir um género sempre que é adicionada ou removida uma música, ou sempre que uma música é editada. Sempre que existe um empate a decisão tomada foi tornar o género nulo. De qualquer forma este método já cria uma lista de géneros em caso de empate.

## Classe Search

A classe de pesquisa permite a criação de um objeto que contém diversas listas atribuídas pelo método de pesquisa na classe RockstarIncManager. Isto permite que a pesquisa seja facilmente personalizável e atualizável.

Nesta classe existem dois construtores diferentes, um que cria uma pesquisa para o cliente e outro que cria a pesquisa para um músico. A pesquisa para um músico é bastante mais simples pois apenas permite a pesquisa das próprias músicas. No caso do cliente permite a pesquisa de músicas pelo nome, de músicas pelo nome do artista e de coleções.

Esta estratégia permite que assim que o termo seja pesquisado toda a pesquisa de todos os critérios seja feita em bloco.

## Classe Música

A classe música contém diversos atributos: nome, género, criador de música associado, lista de avaliações, preço, lista de preços históricos, booleano que difine se a música está ou não ativa, classificação atribuída, album associado.

Métodos:

Esta classe tem diversos getters e setters necessários ao funcionamento do programa.

-addEvaluation(): este método é responsável por criar uma nova avaliação no caso do cliente nunca ter feito uma avaliação à música em questão. No caso de o cliente já ter uma avaliação anterior é feita apenas uma atualização à avaliação anterior.

-calculateClassification(): sempre que é adicionada uma avaliação ou editada é feito um recalculo da classificação global dessa música tendo como base a média de todas as avaliações.

## Classes Acessórias: MusicAquisition, BalanceDeposit, MusicEvaluation.

Estas classes têm como principal propósito a gravação de dados associados a operações feitas com músicas pelo cliente. Elas têm um papel ativo no funcionamento do programa de forma a corresponder a todos os requisitos apresentados no enunciado e permitir expansibilidade futura da aplicação caso seja necessário.

A classe Genre guarda a lista de géneros em forma de “enum”, já que esta informação é acedida por várias classes da aplicação.

## Classes Gráficas

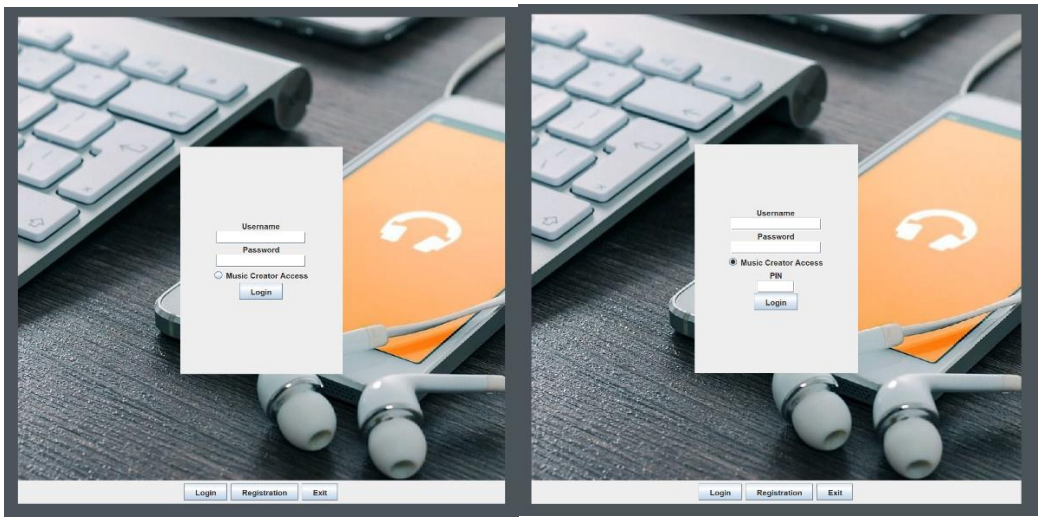
As principais classes associadas à interface gráfica do programa são:

-GUIManager, permite toda a gestão das restantes JFrams, JDialogs e avisos ao utilizador como toda a comunicação com a classe RockstarManager.

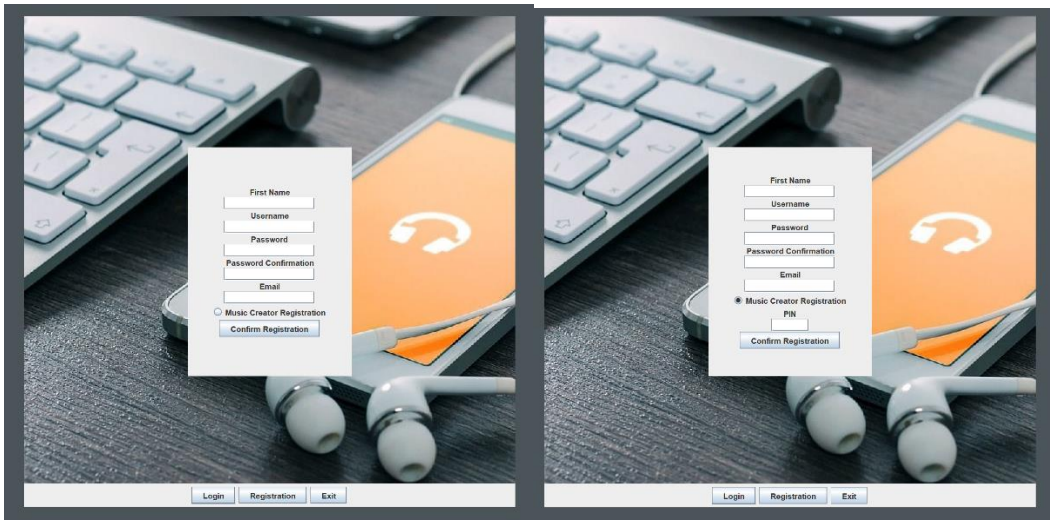
-As classes ClientGUI / MusicCreatorGUI / LoginRegistrationGUI são os principais três elementos que constituiem a parte visual da aplicação.

-As classes EditMusicDialog / ImagePaths / LogRegFrame / RandomPlaylistSelectionDialog / RetangleBarCharComp  entre outras, são classes acessórias que complementam todas as funcionalidades implementadas.

# Implementação e funcionalidades



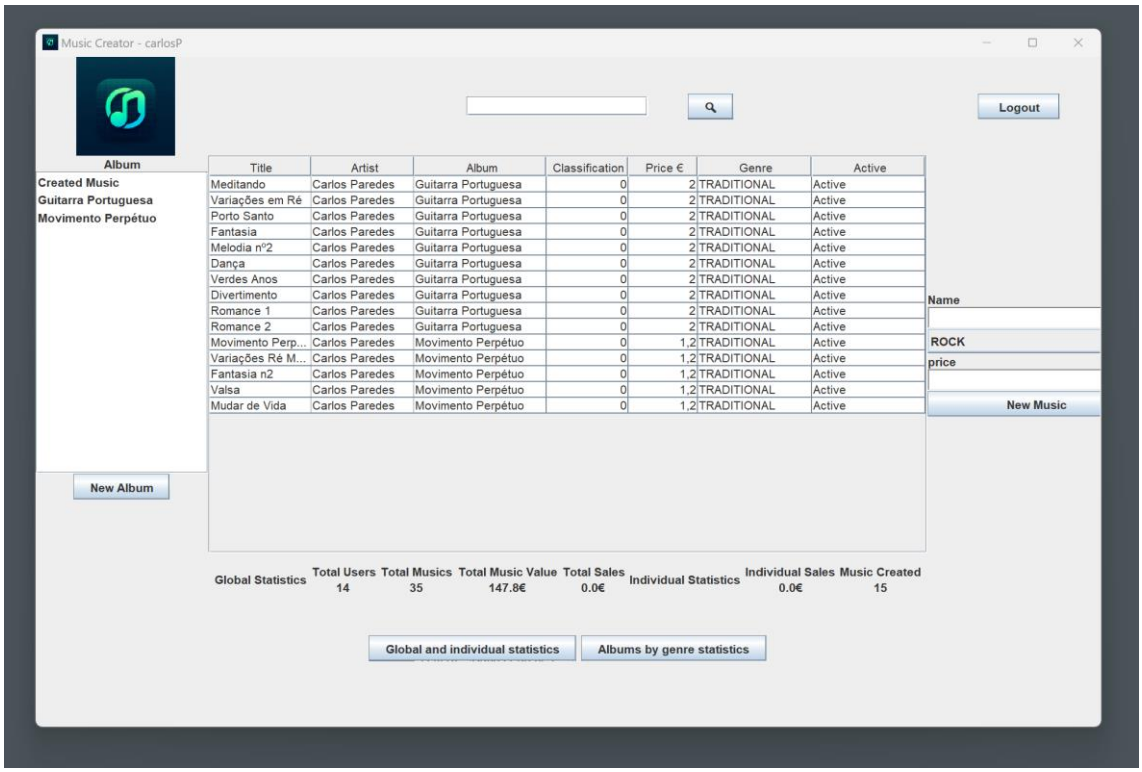

No Painel de Login é possível selecionar a opção de login para Artista ou para cliente.




No Painel de registo é igualmente possível fazer esta seleção.

Um dos requisitos deste projeto é um sistema de login e registo de novos utilizadores. Durante o registo são implementadas diversas validações. Uma das características do registo é permitir que um músico repita as suas credenciais para criar uma conta sua de cliente e vice-versa.

## Funcionalidades do Artista (MusicCreator)



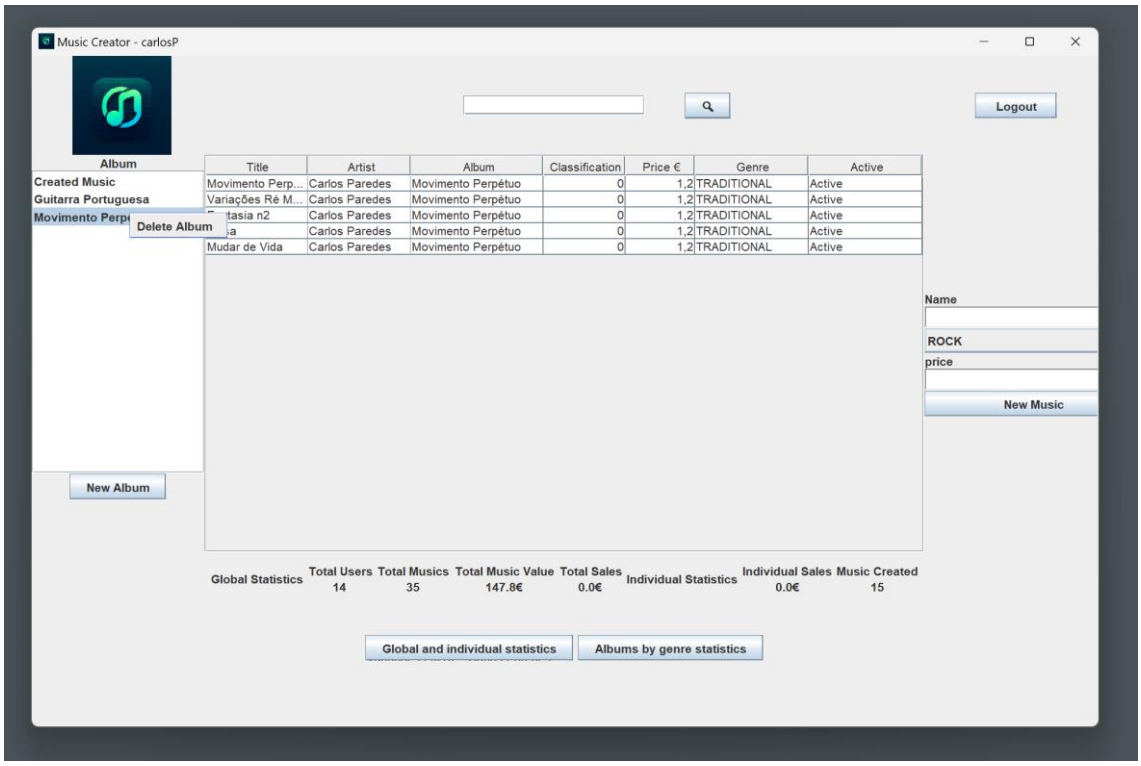

A janela correspondente ao artista é gerida através de um BorderLayout. De notar que todas as colunas das tabelas apresentadas são ordenáveis.

- A oeste está associado um painel que contém uma lista de Álbuns. O primeiro elemento da lista não é um álbum verdadeiro mas sim uma compilação de todas as músicas criadas pelo artista. Esta lista é selecionável e faz atualizar a tabela central tendo em conta o álbum selecionado.

-A norte o “TextField” de pesquisa e o botão de “logout”.

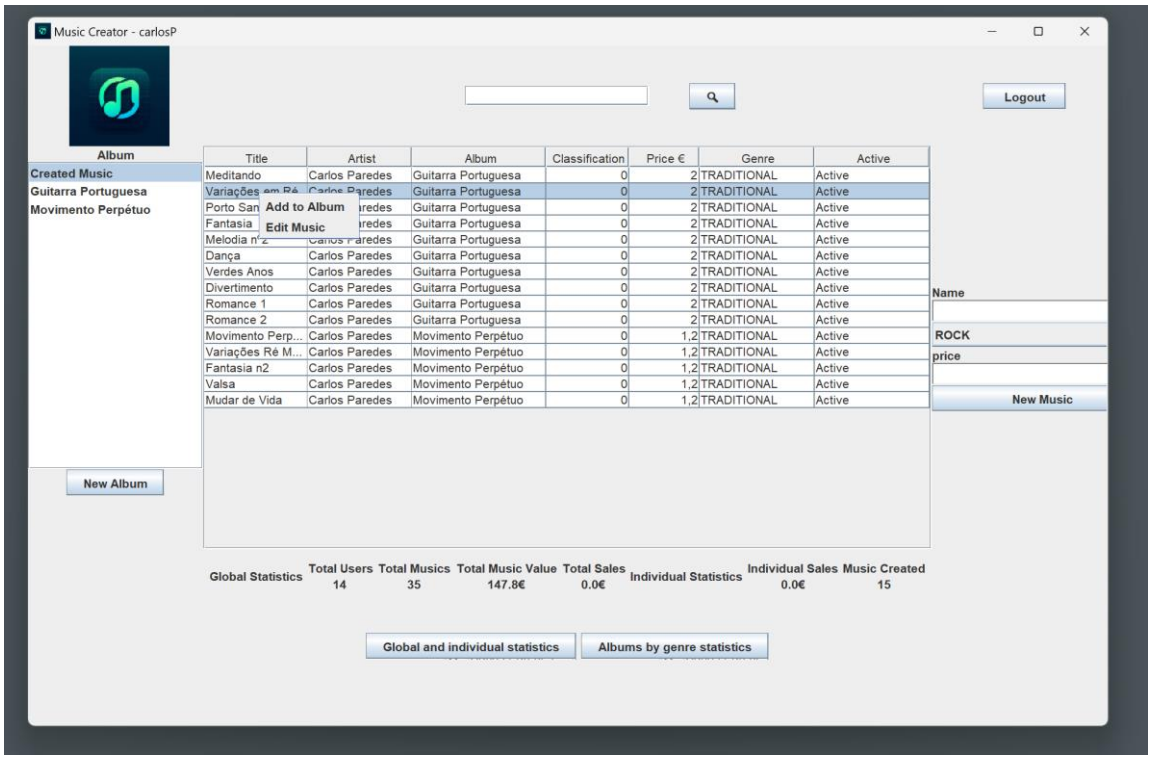
-A este o painel responsável pela criação de nova música.

-No sul o painel de estatísticas.

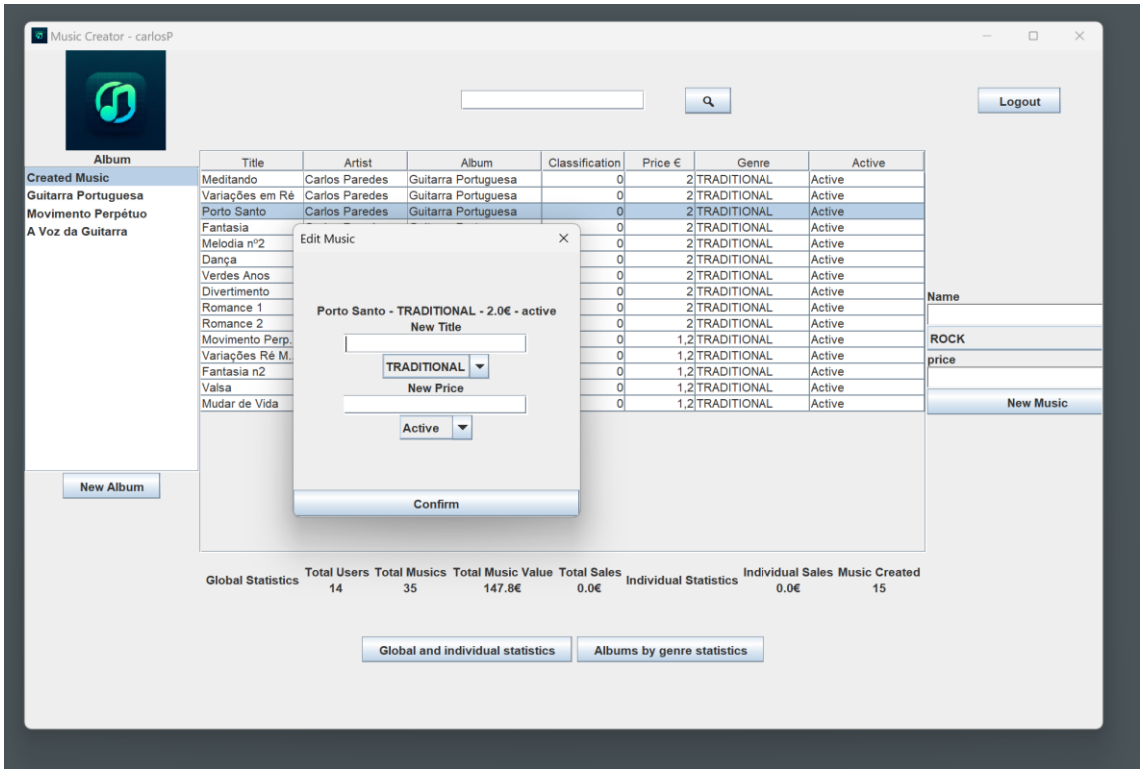



Uma das características do aplicação criada é a possibilidade do utilizador interagir com base em "JPopUpMenus" criados em situações especificas do programa.

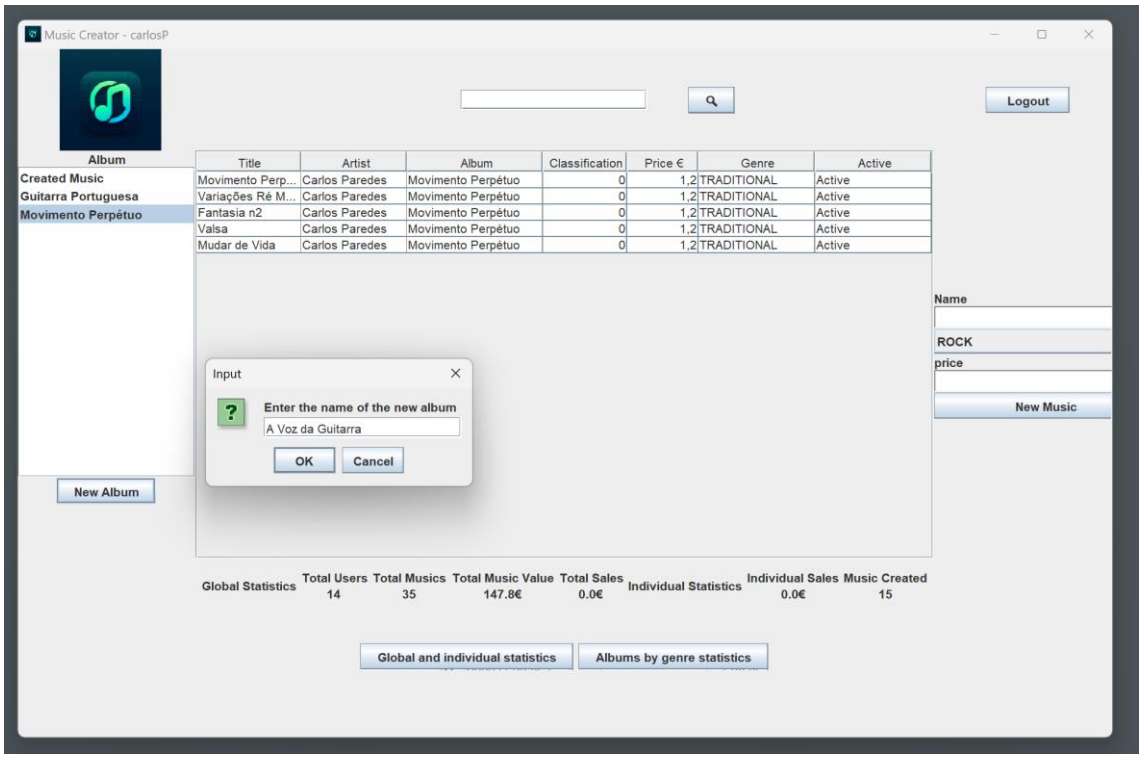
Neste caso a seleção de um Álbum permite a sua eliminação.



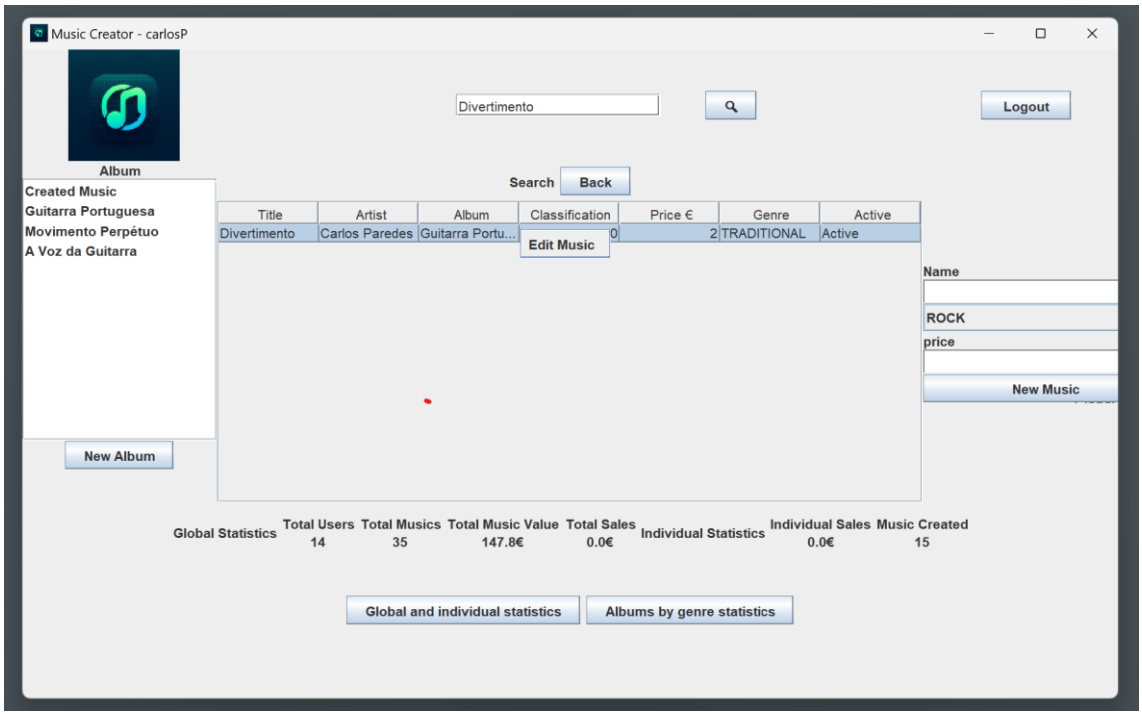
O clique com o botão do rato direito numa música com a seleção prévia de “Created Music” irá gerar um menu que permite adicionar essa música a um álbum específico ou ainda selecionar a opção de edição da música.



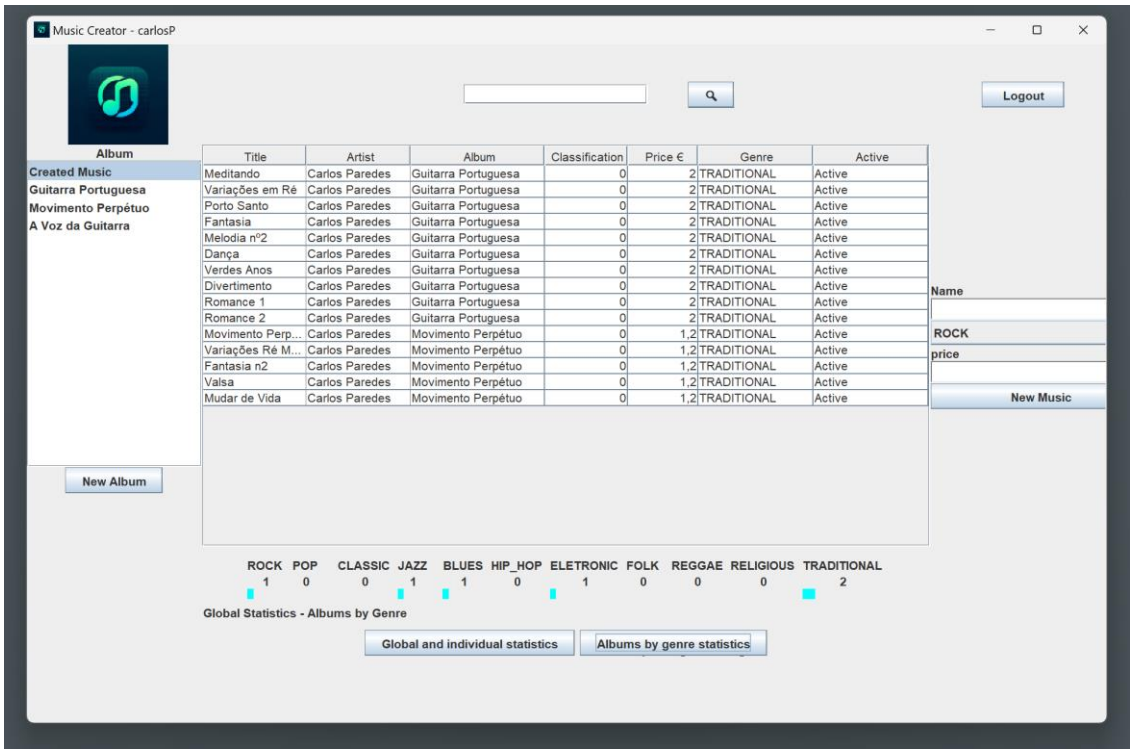

A selecionar a edição é possível alterar o nome, o género, o preço e o estado de ativo ou inativo. (Para alterar um dos valores não é necessário realizar qualquer ação nos outros parâmetros)




Para criar um álbum basta clicar no botão “New Album” o álbum criado será sempre vazio.



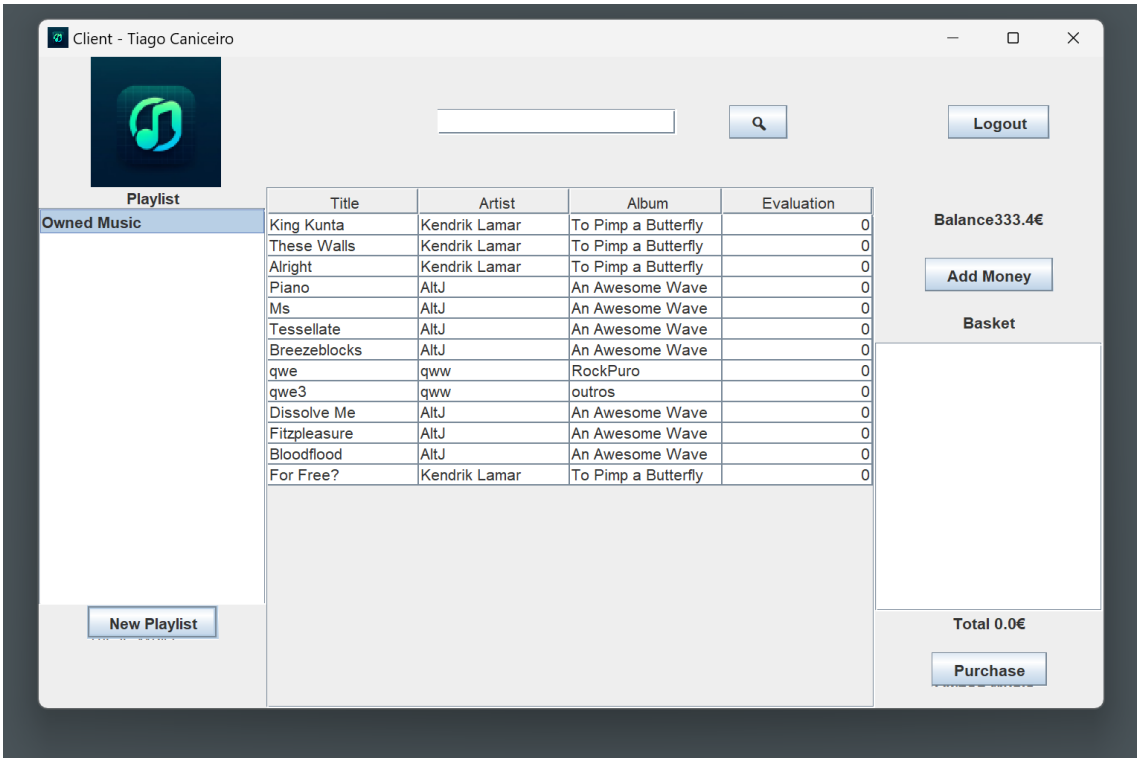

Nesta imagem é possível observar a função de pesquisa. No caso do Artista apenas é possível pesquisar sobre as músicas do próprio. É também possível editar a música a partir deste painel.




No Painel Sul é possível observar e alternar entre diversos tipos de estatísticas.

O pequeno gráfico azul apenas demonstra a proporção da quantidade de álbuns de certo género musical relativamente ao total de álbuns.

## Funcionalidades do Cliente (Client)

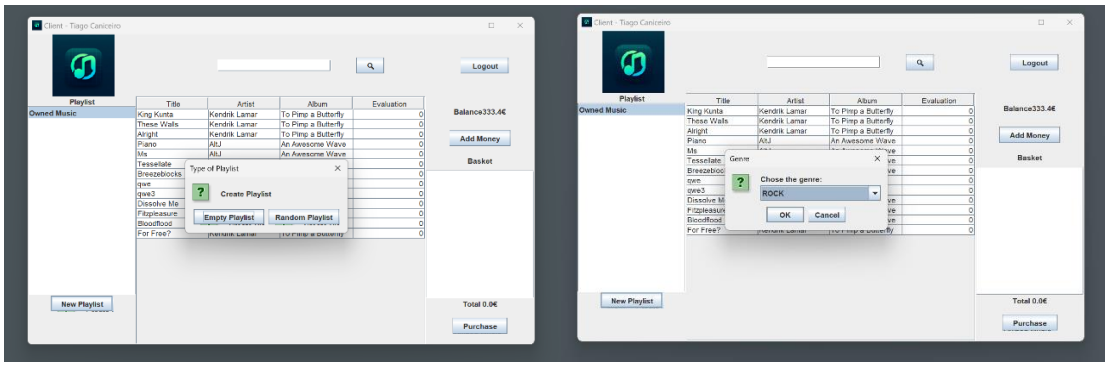


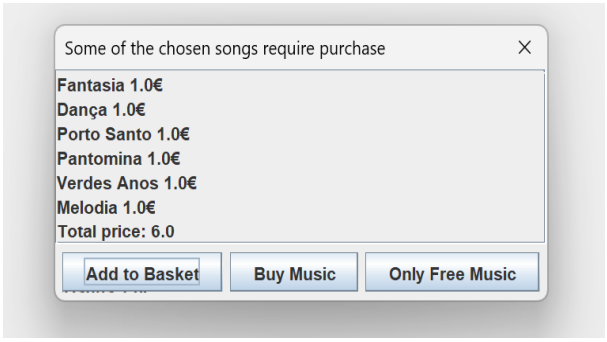

À semelhança da janela anterior a janela do cliente é gerida através de um BorderLayout.

- A oeste está associado um painel que contém uma lista de Playlists. O primeiro elemento da lista não é uma playlist verdadeira mas sim uma compilação de todas as músicas criadas pelo artista. Esta lista é selecionável e faz atualizar a tabela central tendo em conta a playlist selecionada de forma semelhante à janela anterior.

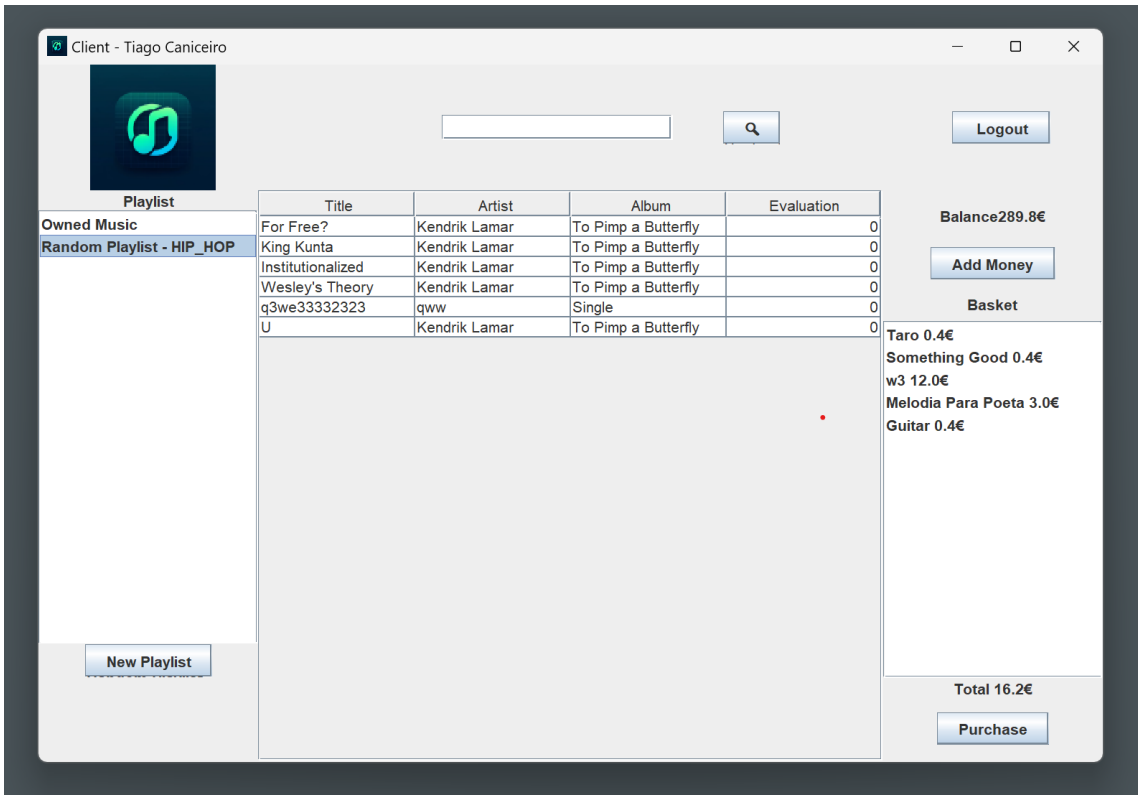
-A norte o “TextField” de pesquisa e o botão de “logout”.

-A este o painel responsável pelas compras de música e saldo do utilizador.

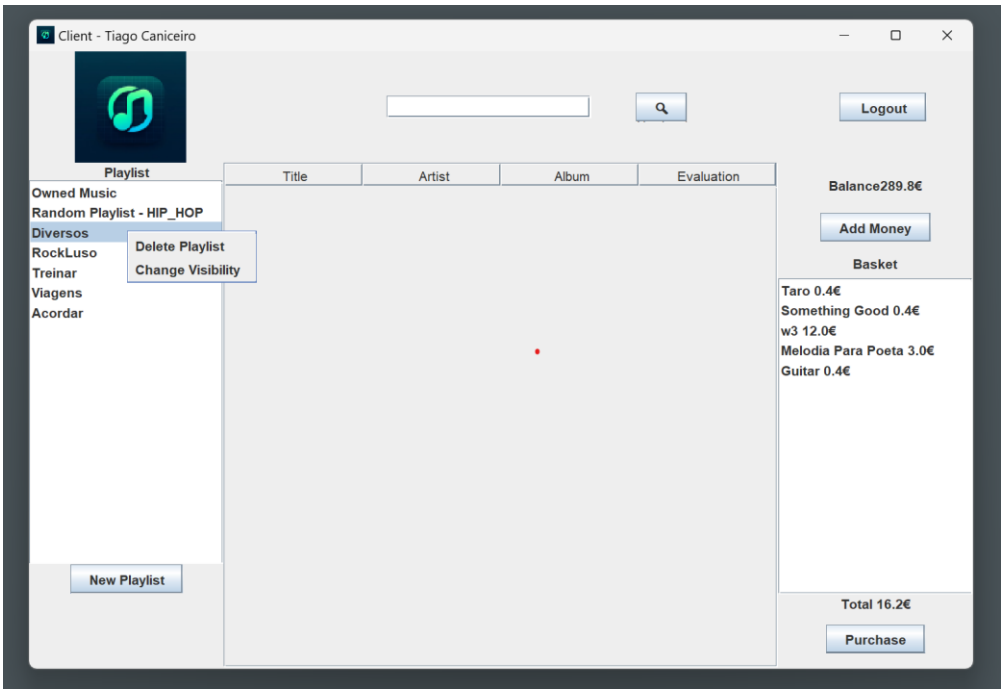




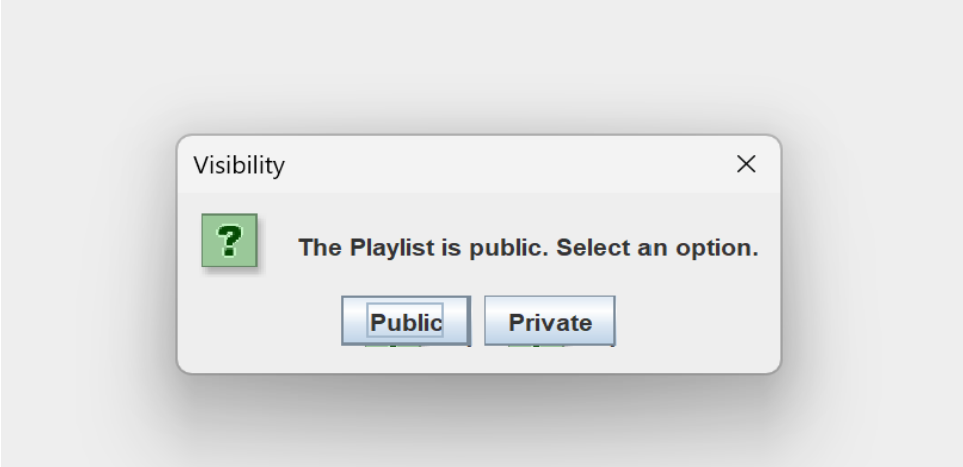

O conjunto de imagens acima demonstra a possibilidade de utilizador escolher várias opções no caso de algumas músicas escolhidas aleatoriamente serem pagas e o utilizador não as possuir na sua coleção.



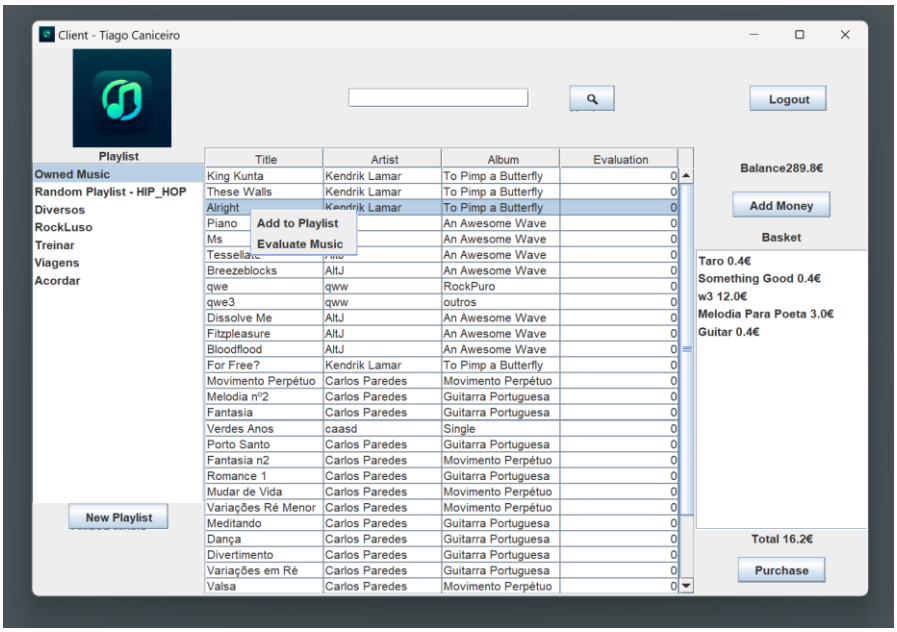

A imagem anterior demonstra a seleção musical de uma playlist criada aleatoriamente, as músicas pagas neste caso foram adicionadas ao cesto.



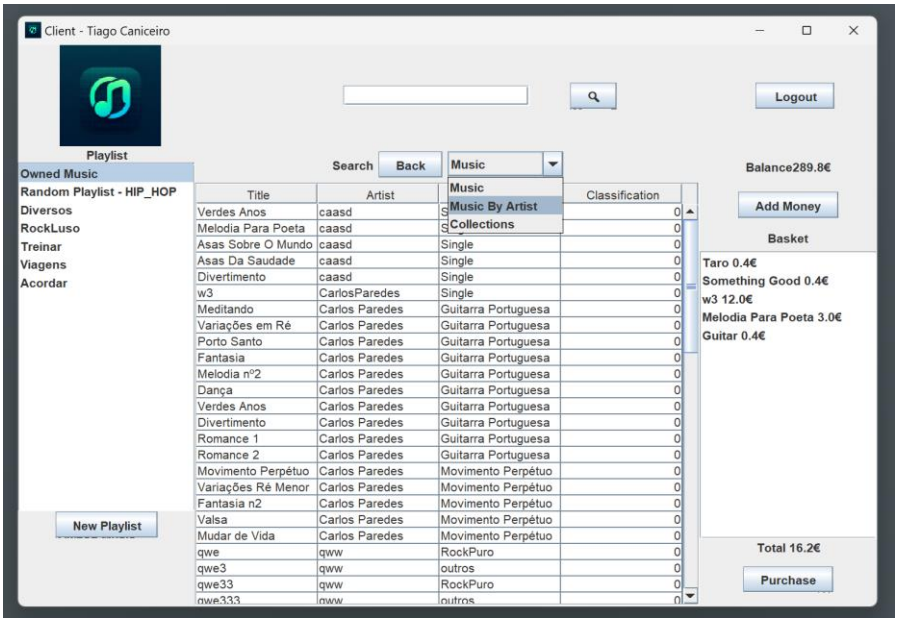

Demonstração do menu no caso de clique com o botão direito numa playlist.



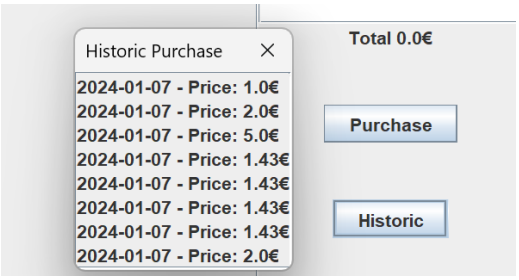

O utilizador poderá optar por tornar a sua playlist pública ou privada. A atribuição de visibilidade como privada numa playlist fará com que esta não seja apresentada na pesquisa de outros utilizadores.



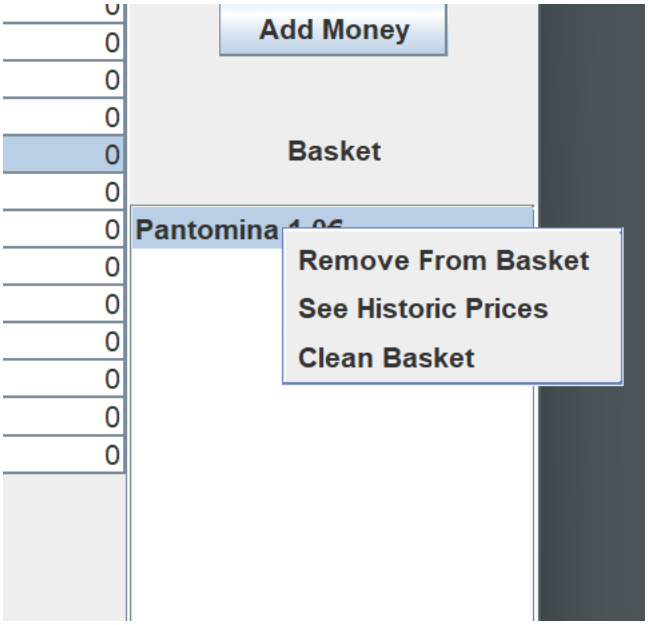

Demonstração da utilização de clique com o botão direito do rato numa música. Isto permite adicionar a uma playlist e fazer uma avaliação da música. Nesta tabela aparecerá o a avaliação que o utilizador atribui à música. Na tabela de pesquisa aparecerá a média de avaliações de todos os clientes. Todas as músicas que não têm álbum associado são consideradas “Single”.




Na funcionalidade de pesquisa é possível realizar a pesquisa de músicas por um termo. A pesquisa do termo é possível ser aplicada ao nome da música ou ao nome do artista (mostrará as músicas do artista). Ainda é possível realizar a pesquisa por coleções musicais em que serão mostrados todos os álbuns e todas as playlists públicas com base no termo pesquisado.




São ainda apresentadas funcionalidades adicionais como a da eliminação de músicas do cesto de compras e a apresentação de um histórico de compras.



## Desafios e soluções

Um dos principais desafios deste projeto foi a decisão relativa à implementação de playlists aleatórias. A solução encontrada escolhe músicas de forma aleatória na lista global. Isto faz com que o utilizador possa não ter algumas das músicas selecionadas. A solução apresentada permite que sejam apresentadas ao utilizador músicas que eventualmente sejam menos compradas ou que num primeiro momento o utilizador não estaria a pensar ouvir o pode ser visto como um incentivo à aquisição de tais músicas. Consideramos a solução apresentada apropriada respondendo aos requisitos da aplicação.

Outro dos desafios encontrados foi a funcionalidade de pesquisa. De forma a responder a este requisito foi criada uma classe de pesquisa que trata uma pesquisa como um objeto que é construída através de diversas listas. Isto permite fazer a pesquisa de um termo em bloco e faz com que seja fácil personalizar a pesquisa adicionando ou retirando listas.

Outro desafio encontrado foi a atribuição de um género a um álbum de forma a fazer as estatísticas pedidas. Na implementação escolhida as músicas são criadas com um género específico o qual pode ser editado. Na necessidade de atribuir um género a um álbum e dando a possibilidade de um álbum poder ter mais que um género musical a classe álbum tem a responsabilidade de calcular o género mais comum. No caso de empate a opção escolhida foi o de atribuir ao género como “null” por uma questão de simplificação. Fica em aberto a possibilidade da atribuição de vários géneros a um álbum e até mesmo a uma música.

Este projeto realiza a gestão de dados criando um ficheiro de objetos único. A gravação é feita quando o utilizador fecha o programa e de forma automática de 2 em 2 minutos. O ficheiro de objetos é lido cada vez que a aplicação é iniciada e se este ficheiro não existir é criado um novo. O projeto foi realizado de forma a que cada classe proteja os seus dados.

# Conclusão

Este projeto foi essencial como trabalho curricular para a consolidação de conhecimentos adquiridos durante o primeiro trimestre do curso acertar o rumo. Outro aspeto importante a realçar durante a execução do trabalho foi a aprendizagem continua de outros aspetos importantes e de tecnologias como Swing para estruturar o projeto.