Universidade do Minho

Braga, Portugal

SPOTIFYUM

Relatório do Trabalho Prático

Programação Orientada aos Objetos

Departamento de Informática

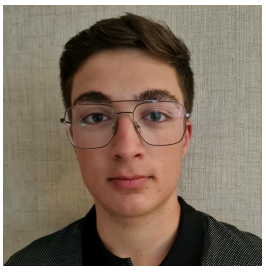
Engenharia Informática 2024/25

Grupo TP4:

A106936 - Duarte Escairo Brandão Reis Silva

A106932 - Luís António Peixoto Soares

A106856 - Tiago Silva Figueiredo



17 Maio 2025

Índice

1. Introdução

No âmbito da Unidade Curricular de Programação Orientada a Objetos, foi proposto um trabalho prático que visava desenvolver uma aplicação de gestão de utilizadores, músicas, álbuns e playlists.

O nosso grupo desenvolveu então essa aplicação, o SpotifyUM, que implementa dois modos de utilização distintos, o modo de utilizador, e o modo de administrador.

Um utilizador, caso esteja registado na aplicação, pode consultar as suas informações pessoais, ver o seu histórico, ouvir músicas, álbuns e playlists, alterar o seu plano de subscrição, e caso tenha as devidas permissões, pode adicionar playlists e álbuns à sua biblioteca pessoal, e ainda criar e gerar playlists pessoais.

Um administrador pode registar e remover as várias entidades existentes (utilizadores, músicas, álbuns e playlists) e ainda ver algumas estatísticas disponibilizadas pela aplicação.

Quanto aos planos de subscrição disponibilizados pela aplicação, estes foram implementados de forma a serem expansíveis, ou seja, no futuro, a criação de novos planos de subscrição seria relativamente simples, e não implicaria a mudança de muitas classes.

2. Descrição da Arquitetura

Nesta secção do relatório serão abordadas as várias classes que foram implementadas na nossa aplicação, bem como os seus atributos, e ainda algumas das decisões tomadas pelo grupo, no que diz respeito a essas mesmas classes.

2.1. Classes da Aplicação

O SpotifyUM tem cinco classes principais, sendo elas as classes `Utilizador`, `Musica`, `Album`, `Playlist` e `PlanoSubscricao`, estas representam as entidades que são o suporte de toda a aplicação.

A classe `Utilizador` define todas as entidades deste tipo, e ao contrário do que era referido no enunciado, optamos por criar um conceito de utilizador singular, ou seja, não existem subclasses do tipo `UtilizadorFree` ou `UtilizadorPremium`. O plano de subscrição atribuído a um utilizador tem o papel que teriam essas subclasses e ainda um pouco mais. Esta mudança relativa ao enunciado pode ser justificada com um simples exemplo. Suponhamos que temos um `UtilizadorPremium` com o plano de subscrição `PlanoPremiumBase`, caso esse utilizador alterasse o seu plano para `PlanoFree`, em teoria deveria continuar a ter acesso às funcionalidades *premium* porque é uma instância da classe `UtilizadorPremium`. O mesmo aconteceria num caso inverso, onde um `UtilizadorFree` fizesse um *upgrade* ao seu plano. Tem isto em conta, o grupo decidiu então ter um utilizador singular, e tudo o que se referisse a acessos e permissões seria tratado com o plano atribuído ao utilizador. Todos os utilizadores têm um campo *id* que se trata de um identificador único.

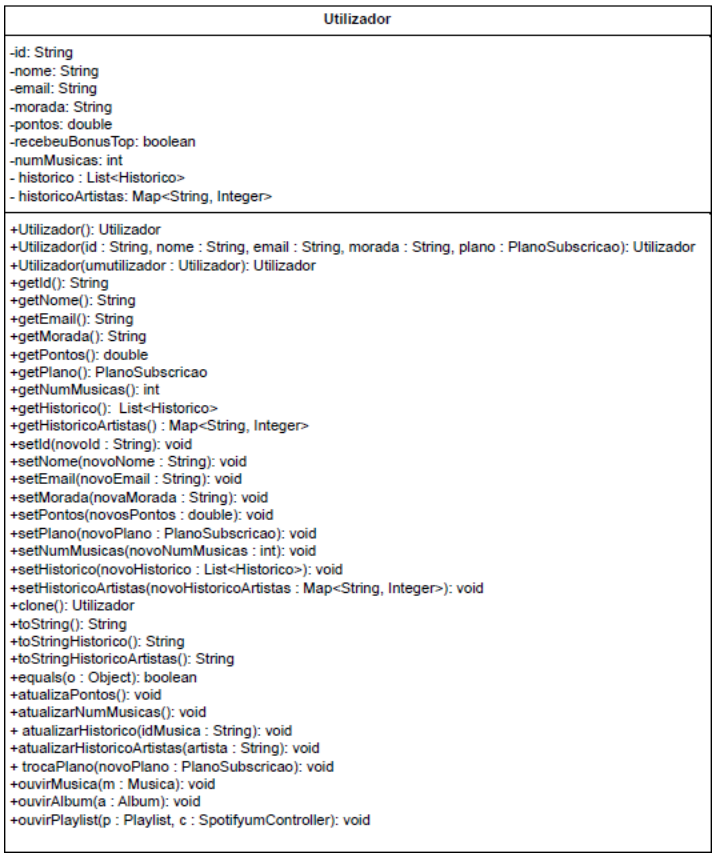


Figura 1: Definição do `Utilizador` no diagrama UML

A classe `Musica` define todas as entidades deste tipo. Tal como é referido no enunciado existem três tipos de músicas, as "normais", as explícitas e as multimédia. Para representar as últimas duas criamos duas subclasses da classe `Musica`, a classe `MusicaExplicita` e a classe `MusicaMultimedia`. O que as distingue relativamente à sua superclasse é a sua reprodução, pois estas vêm com uma identificação diferente devido a uma variável de instância, `public static final String AVISO`, no caso das explícitas e `public static final String VIDEO` no caso das multimédia. Todas as músicas têm um *id* que é um identificador único.

A classe `Album` define todas as entidades deste tipo, e trata-se de uma coleção de músicas. Cada álbum contém um título e uma lista de músicas. A associação entre as classes `Musica` e `Album` dá-se por agregação já que na lista de músicas estão guardados os apontadores das músicas passados como parâmetro.

A classe `Playlist` é uma classe abstrata que generaliza a entidade playlist, é composta por um nome, uma lista de músicas, um `boolean` para caracterizar se a playlist é pública ou não e ainda o autor responsável por criar a playlist.

Conforme o enunciado, foi necessário criar quatro subclasses para a entidade playlist:

- `PlaylistAleatoria`: esta subclasse refere-se ao único tipo de playlists que pode ser reproduzido por utilizador com o plano `PlanoFree`. Estas playlists são todas públicas e o seu autor é o Sistema, ou seja o SpotifyUM. Uma das suas características é o facto da sua reprodução ser aleatória.
- `PlaylistTempoGenero`: esta classe refere-se às playlists que são criadas com um determinado tempo máximo e com músicas de apenas um género. Estas playlists são todas públicas e o seu autor é novamente o Sistema. Qualquer plano de subscrição permite reproduzir este tipo de playlists.
- `PlaylistPersonalizada`: esta subclasse refere-se às playlists que podem ser criadas apenas por utilizadores com planos *premium* (`PlanoPremiumBase` ou `PlanoPremiumTop`). Neste caso são os utilizadores que as criam que definem se a playlist será pública ou não. Caso as playlists deste tipo não sejam públicas, os outros utilizadores (que têm permissões para tal) não as poderão reproduzir.
- `ListaFavoritos`: esta subclasse refere-se às playlists que são geradas apenas para os utilizadores que tenham o plano `PlanoPremiumTop`. Estas são geradas a partir do histórico de reproduções do utilizador e só podem ser reproduzidas pelo próprio utilizador à qual elas se referem, já que nenhuma delas é pública.

A associação entre as classes `Musica` e `Playlist` dá-se por agregação já que na lista de músicas estão guardados os apontadores das músicas passados como parâmetro.

Existem também algumas classes auxiliares, como é o caso do `Historico` e das classes referentes aos menus.

A classe `Historico` contém um identificador de uma música e a data que essa música foi reproduzida. A data é registada como uma variável de instância do tipo `LocalDate`. Esta classe está associada aos utilizadores na medida em que a classe `Utilizador`

detêm uma lista de históricos. Esta lista é utilizada no momento de geração de listas de favoritos.

As classes `Menu` e `MenuOpcao` são responsáveis por gerir os menus, ou seja, por construí-los, e mostrar as suas opções.

2.2. Exceções

Para um melhor tratamento de erros e situações inesperadas, definimos vários tipos de exceções:

- `EntidadeExisteException`: exceção lançada quando uma entidade já existe (utilizador, musica, álbum ou playlist). Utilizada principalmente em métodos de adição.
- `EntidadeNaoExisteException`: exceção lançada quando uma entidade não existe (utilizador, música, álbum ou playlist). Utilizada principalmente em métodos de remoção.
- `OpcaoException`: exceção lançada quando uma opção escolhida é inválida. Por exemplo, no momento da escolha do plano de subscrição, se a opção escolhida for diferente de `PlanoFree`, `PlanoPremiumBase` ou `PlanoPremiumTop`, esta exceção é lançada.
- `PlanoException`: exceção lançada quando é feito um acesso que o plano do utilizador não permite. Por exemplo, se um utilizador tem o `PlanoFree` e reproduzir uma playlist aleatória esta exceção será lançada.
- `PublicException`: exceção lançada quando um utilizador quer reproduzir uma playlist que não é pública.

2.3. Model, View, Controller

A arquitetura implementada para a interação com o utilizador foi a abordada nas aulas teóricas, ou seja, o MVC (*Model*, *View*, *Controller*).

Como é sabido, a *view* é responsável por interagir diretamente com o utilizador da aplicação, o *controller* é a ponte de ligação entre a *view* e o *model*, e por fim o *model* é responsável por toda a atividade computacional propriamente dita do programa.

No nosso caso, a view é bastante abstrata, por isso limita-se a mostrar os menus e as suas opções e a pedir *input*. O *controller* é de facto a ponte de ligação entre os dois outros objetos, mas também faz algum processamento. O *model* tem quatro coleções do tipo `Map` que armazenam objetos do tipo `Utilizador`, com o *id* como chave; `Musica`, com o *id* como chave; `Album`, com o título como chave e `Playlist`, com o nome como chave. Apesar de todas as músicas pertencerem a um álbum, e parecer redundante ter uma coleção de músicas, decidimos na mesma ter esta coleção presente no *model*, primeiro, porque acaba por atuar como uma espécie de *database* das músicas, mas também porque, quando o utilizador pretende reproduzir só uma música, apenas é necessário procurar o seu *id* na coleção de músicas, ao invés de procurar a música dentro da coleção dos álbuns. Tendo isto em conta o *model* processa todas as ações requeridas pelos utilizadores através dos métodos implementados. Como o *model* atua

sobre as diversas entidades presentes na aplicação, e acaba por agrega-las numa única classe, este funciona também como uma **Facade**.

2.4. Estatísticas

No *model* estão também implementados os métodos que calculam as estatísticas pedidas no enunciado do trabalho prático. Como todas as estatísticas necessitam dos dados presentes nas várias coleções do *model*, o mais lógico a se fazer foi de facto manter essas implementações nesta classe, para evitar passar as referências das coleções para fora da classe *model*, preservando assim a sua integridade e respeitando o encapsulamento. As estatísticas implementadas foram as seguintes:

- `musicaMaisReproduzida()`: calcula qual a música mais reproduzida da aplicação
- `artistaMaisEscutado()`: calcula qual o artista mais ouvido da aplicação
- `utilizadorMaisMusicaOuvidas()`: calcula qual o utilizador que mais músicas ouviu desde sempre
- `utilizadorMaisMusicasOuvidasIntervalo()`: calcula qual foi o utilizador que mais músicas ouviu num determinado intervalo
- `utilizadorMaisPontos()`: calcula qual o utilizador com mais pontos
- `tipoMusicaMaisReproduzido()`: calcula qual o género de música mais reproduzido da aplicação
- `numeroPlaylistsPublicas()`: calcula o número de playlists públicas existentes
- `utilizadorMaisPlaylists()`: calcula qual o utilizador com mais playlists na sua biblioteca pessoal

```
public void utilizadorMaisPontos(){
    String id = null;
    double max = 0;

    for (Utilizador u : this.utilizadores.values()) {
        if (u.getPontos() > max) {
            max = u.getPontos();
            id = u.getId();
        }
    }

    if (id != null) {
        System.out.println("O utilizador com mais pontos é: " + id
        + " (" + max + " pontos)");
    } else {
        System.out.println("Nenhum utilizador encontrado.");
    }
}
```

Implementação de uma das estatísticas.

3. Descrição da Aplicação

Nesta secção serão mostrados o funcionamento e as funcionalidades da aplicação desenvolvida.

3.1. Opções dos Menus

Quando corremos a aplicação com o comando

```
./gradlew run -console=plain
```

é mostrado no terminal o menu principal. Neste é possível entrar em modo de administrador, entrar em modo de utilizador, guardar o estado atual da aplicação, ou carregar um estado já previamente guardado.

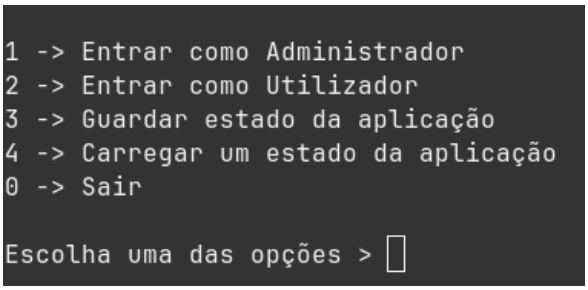


Figura 2: Menu principal da aplicação

Se escolhermos entrar em modo de administrador teremos acesso ao menu do administrador que possibilita a adição, remoção e consulta das várias entidades existentes na aplicação, e ainda podemos ver as estatísticas fornecidas pela mesma.

Figura 3: Menu do administrador

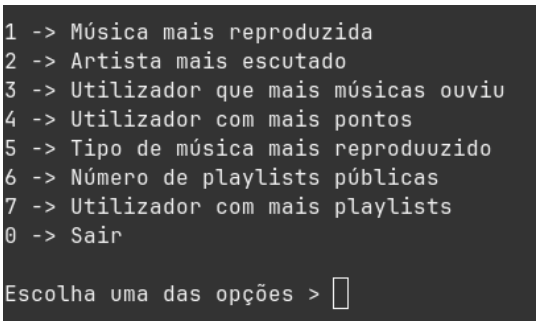


Figura 4: Menu das estatísticas

Por outro lado, se optarmos por entrar em modo de utilizador, teremos primeiro de colocar o nosso identificador, e caso este se encontre dentro da aplicação, temos então acesso ao menu do utilizador. Aqui podemos consultar as nossas informações pessoais, ver os nossos históricos, reproduzir músicas, álbuns e playlists, e ainda trocar o nosso plano de subscrição.

```
1 -> Entrar como Administrador
2 -> Entrar como Utilizador
3 -> Guardar estado da aplicação
4 -> Carregar um estado da aplicação
0 -> Sair

Escolha uma das opções > 2
Introduza o seu ID de utilizador > U35
```

Figura 5: *Login* de um utilizador

Como os diferentes planos de subscrição, `PlanoFree`, `PlanoPremiumBase` e `PlanoPremiumTop`, fornecem diferentes permissões aos utilizadores, o menu limitará algumas funcionalidades consoante o plano do utilizador que fizer o *login*.

```
1 -> Informações do Utilizador
2 -> Ver histórico
3 -> Ver histórico de Artistas
4 -> Ouvir Música
5 -> Ouvir Álbum
6 -> Ouvir Playlist
7 -> (indisponível)
8 -> (indisponível)
9 -> (indisponível)
10 -> (indisponível)
11 -> (indisponível)
12 -> (indisponível)
13 -> Alterar o plano de subscrição
0 -> Sair
```

```
1 -> Informações do Utilizador
2 -> Ver histórico
3 -> Ver histórico de Artistas
4 -> Ouvir Música
5 -> Ouvir Álbum
6 -> Ouvir Playlist
7 -> Criar Playlist Personalizada
8 -> (indisponível)
9 -> Adicionar Álbum à Biblioteca
10 -> Adicionar Playlist à Biblioteca
11 -> Ver as minhhas playlists
12 -> Ver biblioteca de albuns
13 -> Alterar o plano de subscrição
0 -> Sair
```

```
1 -> Informações do Utilizador
2 -> Ver histórico
3 -> Ver histórico de Artistas
4 -> Ouvir Música
5 -> Ouvir Álbum
6 -> Ouvir Playlist
7 -> Criar Playlist Personalizada
8 -> Gerar Lista de Favoritos
9 -> Adicionar Álbum à Biblioteca
10 -> Adicionar Playlist à Biblioteca
11 -> Ver as minhhas playlists
12 -> Ver biblioteca de albuns
13 -> Alterar o plano de subscrição
0 -> Sair
```

Figura 6: Menus dos utilizadores com diferentes planos de subscrição

Como pode ser visto nas imagens acima, os utilizadores que possuem o plano `PlanoFree`, não têm acesso às opções de criação e de geração de playlists, os utilizadores que possuem o plano `PlanoPremiumBase` não podem gerar listas de favoritos, e os utilizadores com o plano `PlanoPremiumTop` têm total acesso.

3.2. Funcionalidades da Aplicação

Tal como já foi referido, quem entrar em modo de administrador poderá adicionar entidades, sendo os campos preenchidos pelo *input* pedido pela aplicação, remover entidades através do identificador único, e ainda listar todos os tipos de entidades.

Tal como é demonstrado na figura abaixo, para um administrador criar um utilizador é necessário preencher os campos, *id*, nome, email, morada e plano de subscrição.

```
Escolha uma das opções > 1
ID > U100
Nome > Cristiano Remelhe
Email > cris.remelhe@sapo.pt
Morada > Barcelos
Plano (PlanoFree | PlanoPremiumBase | PlanoPremiumTop) > PlanoFree
```

Figura 7: Adição de um utilizador

Para remover um utilizador do SpotifyUM, basta saber qual o identificador do utilizador que se pretende remover.

```
Escolha uma das opções > 5
ID do utilizador a remover > U100
```

Figura 8: Remoção de um utilizador

Um administrador apenas pode adicionar playlists do tipo `PlaylistAleatoria` e `PlaylistTempoGenero` ao sistema, e as músicas selecionadas para cada uma delas são escolhidas de forma aleatória. No caso das playlists aleatórias basta escolhermos um nome para a playlist, e o número de músicas que a playlist deve ter, já no caso das playlists com tempo e género, é escolhido o nome, o tempo máximo (em minutos), e o género pretendido.

```
Escolha uma das opções > 4

1 -> Gerar Playlist Aleatória
2 -> Gerar Playlist por Tempo e Genero
0 -> Sair

Escolha uma das opções >
```

Figura 9: Menu de escolha das playlists do administrador

```
Escolha uma das opções > 4

1 -> Gerar Playlist Aleatória
2 -> Gerar Playlist por Tempo e Genero
0 -> Sair

Escolha uma das opções > 1
Nome da playlist > aleatoria
Número de músicas > 10
```

Figura 10: Criação de uma playlist aleatória

```
1 -> Gerar Playlist Aleatória
2 -> Gerar Playlist por Tempo e Genero
0 -> Sair

Escolha uma das opções > 2
Nome da playlist > tempo
Tempo máximo (em minutos) > 25
Género > Pop
```

Figura 11: Criação de uma playlist com tempo e género

Um administrador pode também ver as estatísticas disponibilizadas pela aplicação, como por exemplo qual o utilizador com mais pontos.

```
Escolha uma das opções > 4
O utilizador com mais pontos é: U35 (606.5158763770156 pontos)
```

Figura 12: Exemplo de uma das estatísticas disponíveis

Já um utilizador pode ver as suas informações pessoais, e consultar os seus históricos.

```
[UTILIZADOR]
Id: U35
Nome: Tiago Figueiredo
Email: tiago.figueiredo@exemplo.pt
Morada: Barcelos
Pontos: 606.5158763770156
Plano: Plano Premium Top
```

Figura 13: Informações do utilizador U35

```
Música: M10 -> Data: 2025-05-10
Música: M82 -> Data: 2025-05-10
Música: M43 -> Data: 2025-05-10
Música: M65 -> Data: 2025-05-10
Música: M1 -> Data: 2025-05-10
Música: M28 -> Data: 2025-05-10
Música: M3 -> Data: 2025-05-10
Música: M17 -> Data: 2025-05-10
Música: M62 -> Data: 2025-05-10
Música: M14 -> Data: 2025-05-10
```

Figura 14: Excerto do histórico de reproduções do utilizador U35

```
Escolha uma das opções > 3
Artista: "Billie Eilish" -> NumReproducoes: "3"
Artista: "Ed Sheeran" -> NumReproducoes: "14"
Artista: "The Weeknd" -> NumReproducoes: "26"
Artista: "Post Malone" -> NumReproducoes: "3"
Artista: "Taylor Swift" -> NumReproducoes: "6"
Artista: "Ariana Grande" -> NumReproducoes: "4"
Artista: "Justin Bieber" -> NumReproducoes: "5"
Artista: "Dua Lipa" -> NumReproducoes: "5"
Artista: "Drake" -> NumReproducoes: "3"
Artista: "Bruno Mars" -> NumReproducoes: "4"
```

Figura 15: Histórico de artistas do utilizador U35

Pode também reproduzir músicas, quer seja individualmente, mas também álbuns e playlists, para isso basta indicar o identificador dá música, o título do álbum, ou ainda o nome da playlist.

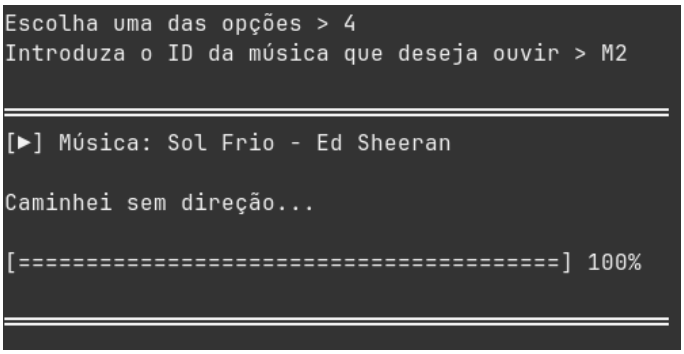


Figura 16: Exemplo de reprodução de uma música

Caso o plano do utilizador o permita, ele poderá também criar playlists personalizadas, gerar uma lista de favoritos pessoal, baseada nos seus históricos, e ainda adicionar álbuns e playlists à sua biblioteca pessoal. Nas playlists personalizadas é necessário escolher o nome, indicar os *ids* das músicas que se pretende adicionar, e indicar se a playlist será pública ou não.

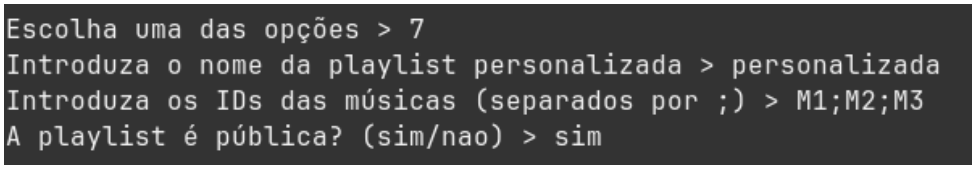


Figura 17: Criação de uma playlist personalizada

Quanto às listas de favoritos, elas podem ser "normais", ou seja, um conjunto de dez músicas baseadas nos seus históricos, ou ainda limitadas por um tempo máximo, ou um conjunto de músicas do tipo `MusicaExplicita`.

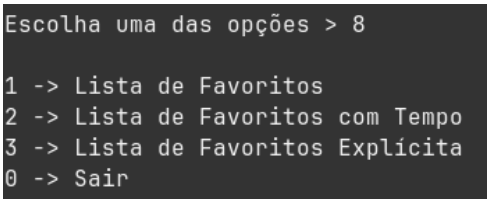


Figura 18: Menu para a criação de listas de favoritos

Por fim, um utilizador pode também alterar o seu plano de subscrição a qualquer momento.

Tal como é pedido no enunciado do trabalho prático o grupo desenvolveu o conceito de música explícita e música multimédia. Como já foi referido na descrição da arquitetura, as suas reproduções são ligeiramente diferentes.

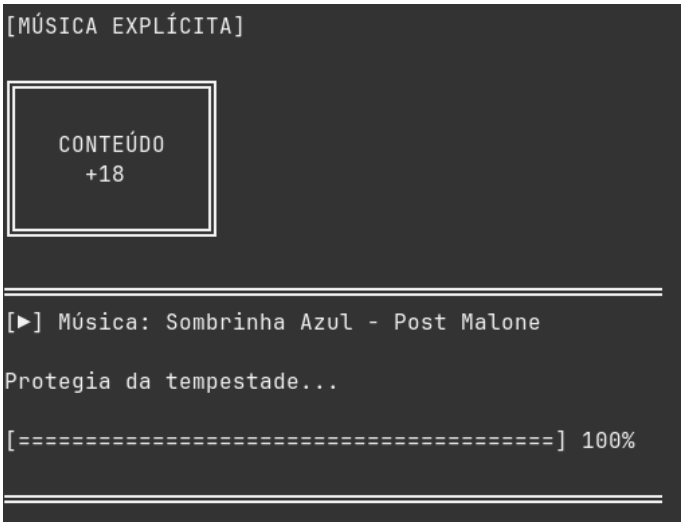


Figura 19: Reprodução de uma música explícita

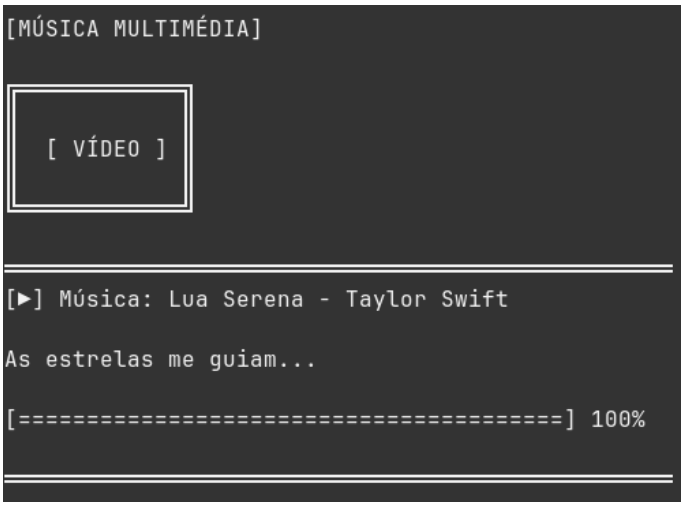


Figura 20: Reprodução de uma música multimédia

4. Conclusão

Em suma, o nosso grupo foi capaz de desenvolver a aplicação proposta, conseguindo cumprir todos os requisitos estipulados no enunciado do trabalho prático.

Sentimos alguma dificuldade no início do trabalho, já que não estávamos completamente acostumados com o paradigma de programação por objetos, mas com o tempo ganhamos confiança e conseguimos dominar, dentro do possível, a linguagem Java de modo a resolver o problema que tínhamos pela frente.

De um modo geral, sentimos-nos satisfeitos e orgulhosos com a aplicação desenvolvida, temos um código modular, respeitamos o encapsulamento e a arquitetura desenvolvida foi pensada numa expansibilidade futura.

5. Anexos