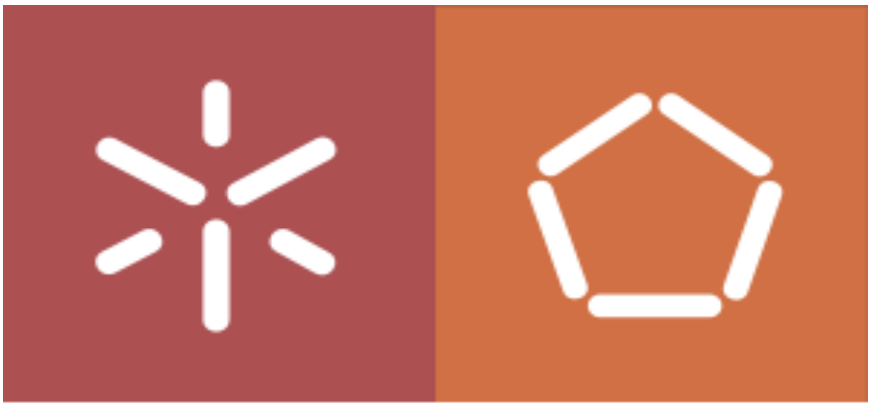
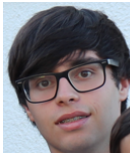


Universidade do Minho
Escola de Engenharia

Programação Orientada a Objetos
Trabalho Prático

Hugo Ferreira Soares (a107293)

Francisco Ribeiro Martins (a106902)



Jorge Gabriel Sá Barbosa(a106799)

17 Maio 2025

Grupo 74

Conteúdo

1.Introdução

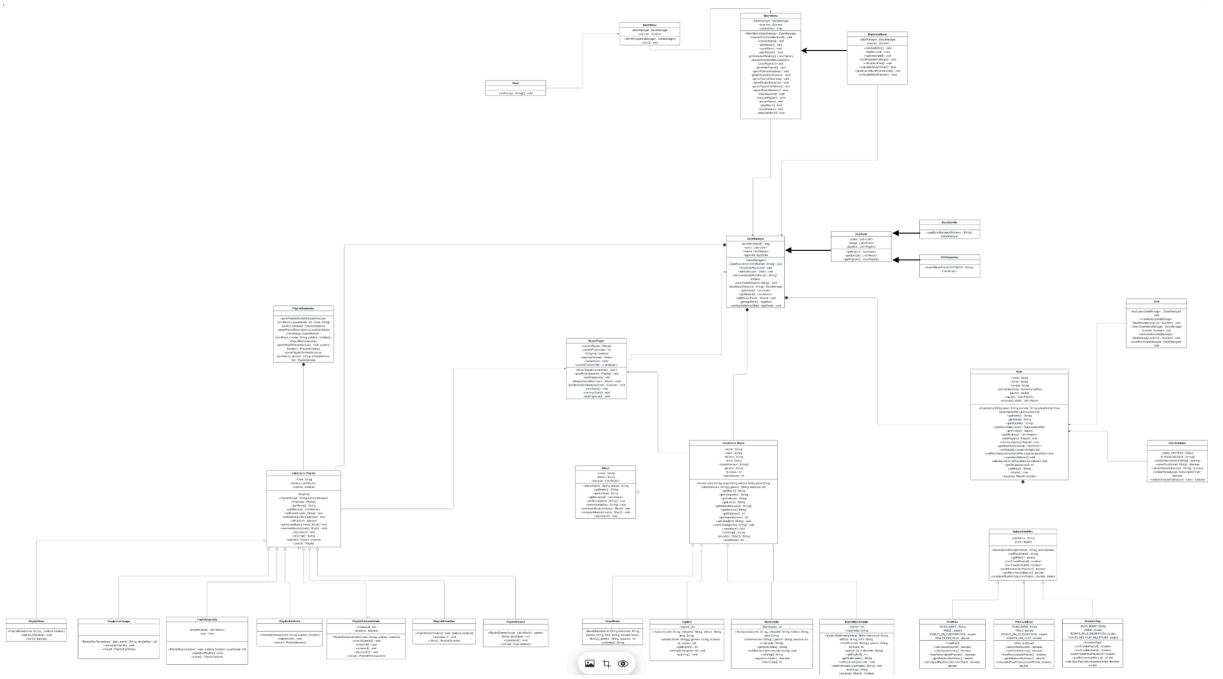
Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Programação Orientada a Objetos(POO), tendo como principal objetivo o desenvolvimento de uma aplicação capaz de gerenciar dados acerca de utilizadores e músicas, de forma a aplicar os conhecimentos adquiridos na UC, usando a linguagem java.

1.1 Descrição do Problema

A informação a ser processada pelo programa é inserida via utilizador. Esta deve ser capaz de solucionar um conjunto variado de questões que sejam mais básicas como a reprodução de uma música, até questões mais complexas como por exemplo a criação de playlists de caráter explícitas.
Neste relatório esperamos explicar as etapas de resolução do projeto apresentado, assim como as dificuldades e respectivas solução para os problemas evidenciados.

2.Sistema

2.1 Diagrama de Classes



Começando pela parte de Interação com o utilizador, esta é composto por três módulos(StartMenu, MainMenu e Statistic) que visam responder e implementar diversos métodos enunciados no trabalho prático.

2.2 Modularização

O trabalho é composto por diversas classes que estão organizadas em módulos de forma a facilitar a procura de informação relativa a um determinado tema.

A. Main
    a. main
B. Modulos de entidades
    a. MusicTypes
        i. BaseMusic
        ii. Explicit
        iii. ExplicitMultimedia
        iv. Multimedia
        v. Music
    b. PlaylistTypes
        i. Playlist
        ii. PlaylistAleatoria
        iii. PlaylistBase
        iv. PlaylisExplicita
        v. PlaylistFavoritos
        vi. PlaylistFavTempo
        vii. PlaylistGenero
        viii. PlaylistPersonalizada
    c. SubscriptionPlans
        i. FreePlan
        ii. PremiumBase
        iii. PremiumTop
        iv. SubscriptionPlan
    d. Album
    e. AppState
    f. User
C. modulos de Serviços
    a. DataManager
    b. MusicPlayer
    c. PlaylistGenerator

D. Módulos de IO
    a. MainMenu
    b. StartMenu
    c. StatisticsMenu
E. Modulos de utilidade
    a. CSVImporter
    b. ReadSerFile
    c. Utils
F. Modulos de Validação
    a. UserValidator

2.3 Caracterização das Entidades

As entidades apresentadas guardam diversos campos de informação de forma a facilitar a resposta ao utilizador de vários métodos

```java
public class User implements Serializable {
    private String nome;
    private String email;
    private String morada;
    private SubscriptionPlan planoSubscricao;
    private double pontos;
    private List<Playlist> biblioteca_pessoal_p = new ArrayList<>();
    private List<Music> musicasOuvidas = new ArrayList<>();
    private List<Album> biblioteca_pessoal_a = new ArrayList<>();
```

```java
public abstract class SubscriptionPlan implements Serializable {
    private final String planName;
    private final double price;
```

A classe User utiliza a classe SubscriptionPlan de forma a dividir os utilizadores por categorias, esta classe é representada como abstrata, tendo métodos representados nas classes: FreePlan, PremiumBasa e PremiumTop, estas classes utilizam um método de herança.

```java
public abstract class Music implements Serializable {
    private String nome;
    private String autor;
    private String editora;
    private String letra;
    private String[] notasMusicais;
    private String genero;
    private int duracao; // segundos
    private int reproducoes;
```

A entidade Music guarda as

informações relativas a uma música, as musicas são divididas nas seguintes classes: BaseMusic, Explicit, ExplicitMultimedia e Multimedia, onde a herança também é utilizada.

```
public abstract class Playlist implements Serializable {
    private String nome;
    private List<Music> musics;
    private boolean publica;
```

Esta entidade refere-se a uma organização de uma certa quantidade de músicas de forma a os utilizadores escutarem as músicas de forma mais clara. As Playlist são divididas em : PlaylistAleatoria, PlaylistBase, PlaylistFavoritos, PlaylistExplicita (só para musicas favoritas), PlaylistFavTempo, PlaylistGenero e PlaylistPersonalizada, estas classes utilizam um método de herança de forma a se comunicar com classes de níveis superiores.

```
public class Album implements Serializable {
    private String nome;
    private String artista;
    private String editora;
    private String anoLancamento;
    private String genero;
    private boolean publica;
    private List<Music> musicas;
```

A entidade Álbum representa uma coleção de músicas associadas a um determinado artista esta classe foi introduzida para que os utilizadores, apesar de não conseguirem alterar o conteúdo dos álbuns, poderem ouvir músicas dos seus artistas preferidos.

2.4 Descrição de alguns módulos

- Módulos de IO
    - Dividimos os módulos de IO em três classes

    - A **StartMenu** implementa a primeira iteração com o utilizador onde é realizado o registo ou login de um utilizador e onde também é possível encerrar o SpotifUM
    - A **MainMenu** é a classe que implementa o maior número de funcionalidades, presente na aplicação, é o local de interação onde métodos relacionados com as músicas são aplicados, por exemplo: reprodução de músicas, criação de playlist, reprodução de playlist e álbuns bem como a remoção dos mesmos. Na MainMenu também é possível alterar o plano de Subscrição do utilizador, remover o utilizador em questão, ver as informações do utilizador tais como o email, o plano de subscrição e a biblioteca pessoal tanto de álbuns como de playlists, existe também uma funcionalidade que permite aos utilizadores adicionarem playlists(públicas) e álbuns a sua biblioteca pessoal.
    - A **Statistic** é a classe onde os métodos relacionados com as estatísticas do programa, encontram-se implementadas 7 estatísticas.
- Módulos de Serviços
    - Neste módulo são executadas as principais tarefas de interação do utilizador com o SpotifUM.
    - Neste módulo são geradas as playlists, músicas são reproduzidas e também são guardadas informações acerca dos utilizadores, músicas e álbuns.
- Módulos utils e validation
    - Nestes módulos são executados métodos de carácter geral, pois são utilizados em diferentes classes e de forma sistemática
    - As classes **ReadSerFile** e **CSVImporter** permitem a leitura e escrita de informação de locais externos além dos dados introduzidos pelos utilizadores

2.5. Funcionalidades

Neste tópico vou apresentar algumas das operações que a aplicação é capaz de realizar, tal como foi descrito anteriormente. Estas funcionalidades demonstram na prática a forma como o sistema

interage com o utilizador e como foram aplicados os conceitos de programação orientada a objetos no desenvolvimento do projeto.

Menu inicial:

```
=== SpotifUM ===

Lista de utilizadores:
- admin
- pobre
- Hugo

1. Sign Up (Criar conta)
2. Sign In (Entrar)
X. Sair
Escolha uma opção:
```

Menu principal:

```
=== SpotifUM ===
1. Ver Informações do Utilizador
2. Tocar Música
3. Tocar Playlist
4. Gerar Playlist
5. Ver Playlists Públicas
6. Remover Playlist
7. Ver Estatísticas
R. Remover Utilizador
8. Alterar Plano de Subscrição
9. Tocar Álbum
10. Ver Álbuns
11. Remover Álbum
X. Menu Inicial
Escolha:
```

Menu das estatísticas:

```
Statistics
1. Musica mais reproduzida
2. Interprete mais escutado
3. Utilizador mais ativo
4. Utilizador com mais pontos
5. Tipo de musica mais reproduzida
6. Quantas playlists publicas existem
7. Utilizador que possui mais playlists
X. Menu SpotifUM
Enter choice:
```

Informações do Utilizador:

```
=== Informações do Utilizador ===
Nome: admin
Email: a@gmail.com
Plano: Premium Top
Pontos: 128,01
Total de playlists: 2
Biblioteca Pessoal de Playlists:
- SICKO MODE (Privada)
- Favs (Privada)
Total de álbuns: 0
→ Nenhum álbum adicionado ainda.
```

Criação de Playlists

```
Como deseja criar a sua playlist:
1 - Aleatória
2 - Por Seleção
3 - Favoritos
4 - Favoritos Explicita
5 - Favorita Por Tempo
6 - Por Tempo e Gênero
1
Qual o nome da playlist
nova
Deseja tornar a playlist pública? (s/n): s
Quantas músicas deseja na playlist
4
Playlist adicionada ao utilizador admin
```

Menu para reproduzir playlist e como elas são reproduzidas

```
=== Playlists Disponíveis ===
--- Biblioteca Pessoal ---
1. SICKO MODE (Privada)
2. Favs (Privada)

--- Playlists Públicas ---
3. pla (Pública)
2

Iniciando reprodução da playlist: Favs
A tocar: Rap God
Artista: Eminem

Letra:
Letra...

Controlos: [>] Próxima | [<] Anterior | [R] Shuffle | [S] Sair
```

3. Discussão

Um dos principais desafios para a realização do trabalho foi a divisão/ separação de músicas com diferentes atributos, para isso utilizamos o mecanismo de herança, que resultou na reutilização de código das superclasses que evitou a duplicação de código, além de herdar os métodos das superclasses, tornou-se apenas necessário adicionar os métodos necessários a cada subclasses ou ainda reescrever o código presente na superclasses.

Outro desafio encontrado foi perceber qual era a classe mais adequada a realizar determinadas funcionalidades. Este desafio não possui uma resolução concreta, pois dependendo do objetivo do método, este pode e deve ser inserido em uma classe diferente, portanto resolvemos este problema da forma que não quebra-se a integridade dos dados e que não obriga-se a uma duplicação significativa de código.

4. Possíveis Melhorias Futuras

Apesar de o projeto cumprir os requisitos essenciais propostos no enunciado, existem várias funcionalidades e melhorias que poderiam ser implementadas no futuro para enriquecer a experiência do utilizador e tornar o sistema mais completo e flexível.

Uma das funcionalidades que consideramos implementar, mas que não conseguimos concluir, foi a importação de músicas a partir de um ficheiro CSV. Esta funcionalidade permitiria carregar um conjunto alargado de músicas de forma automática, evitando a inserção manual e aumentando significativamente a diversidade de opções disponíveis para o utilizador. A integração deste mecanismo seria uma mais-valia clara para futuras versões da aplicação.

Outra melhoria possível seria a criação de novos tipos de playlists, como por exemplo playlists baseadas em estados de espírito ou atividades (ex: “Para Estudar”, “Para Treinar”), ou ainda playlists colaborativas entre utilizadores.

No que diz respeito aos planos de subscrição, poderiam ser adicionados mais níveis, com funcionalidades diferenciadas, como a remoção total de limitações no número de playlists criadas ou o acesso a estatísticas mais detalhadas.

Também seria interessante desenvolver um modelo gráfico da aplicação (GUI), substituindo o atual modelo baseado em consola. Isso tornaria a aplicação mais atrativa e acessível, principalmente para utilizadores menos técnicos. Com esta interface gráfica, poder-se-ia ainda integrar uma funcionalidade para visualização de vídeo nas músicas classificadas como "Multimedia", oferecendo assim uma experiência mais rica e interativa.
Por fim, outro ponto a considerar seria o reforço do módulo de estatísticas, com gráficos visuais, rankings em tempo real ou dados personalizados por período de tempo, promovendo uma maior análise e personalização da experiência musical do utilizador.
Estas ideias representam caminhos interessantes para evoluir o projeto, tornando-o mais robusto, moderno e adaptado a diferentes tipos de utilizadores.

4.Conclusão

O desenvolvimento deste trabalho prático permitiu adquirir alguns conceitos relacionados com a unidade Curricular Programação Orientada a Objetos, nomeadamente o encapsulamento, a herança e a abstração de dados, que achamos que serão importantes no nosso futuro quer seja a nível académico quer a nível profissional.
Na realização deste trabalho também foram necessários conceitos adquiridos em UC 's anteriores nomeadamente o conceito de encapsulamento que foi aprofundado em POO assim como o conceito de modularidade.