# Sistema Digital de Música

Gerado por Doxygen 1.9.8

**1 Índice dos namespaces** 1

1.1 Lista de namespaces . . . 1

**2 Índice dos componentes** 3

2.1 Lista de componentes . . . 3

**3 Índice dos ficheiros** 5

3.1 Lista de ficheiros . . . 5

**4 Documentação dos namespaces** 7

4.1 Referência ao namespace core . . . 7

**5 Documentação da classe** 9

5.1 Referência à classe Album . . . 9

5.1.1 Descrição detalhada . . . 10

5.1.2 Documentação dos Construtores & Destrutor . . . 10

5.1.2.1 Album() [1/2] . . . 10

5.1.2.2 Album() [2/2] . . . 11

5.1.2.3 ~Album() . . . 11

5.1.3 Documentação das funções . . . 11

5.1.3.1 addSong() . . . 11

5.1.3.2 calculateTotalDuration() . . . 12

5.1.3.3 findSongById() . . . 12

5.1.3.4 findSongByTitle() . . . 12

5.1.3.5 getArtist() . . . 12

5.1.3.6 getFormattedDuration() . . . 13

5.1.3.7 getGenre() . . . 13

5.1.3.8 getId() . . . 13

5.1.3.9 getName() . . . 13

5.1.3.10 getNextSong() . . . 13

5.1.3.11 getPreviousSong() . . . 14

5.1.3.12 getSongAt() . . . 14

5.1.3.13 getSongCount() . . . 14

5.1.3.14 getSongs() . . . 15

5.1.3.15 getYear() . . . 15

5.1.3.16 removeSong() . . . 15

5.1.3.17 removeSongAt() . . . 15

5.1.3.18 setArtist() . . . 16

5.1.3.19 setGenre() . . . 16

5.1.3.20 setYear() . . . 16

5.1.3.21 toString() . . . 16

5.1.4 Documentação dos dados membro . . . 17

5.1.4.1 _artist . . . 17

5.1.4.2 _genre . . . 17

5.1.4.3 _id . . . 17
5.1.4.4 _name . . . 17
5.1.4.5 _songs . . . 17
5.1.4.6 _year . . . 17
5.2 Referência à classe Artist . . . 17
5.2.1 Descrição detalhada . . . 19
5.2.2 Documentação dos Construtores & Destrutor . . . 19
5.2.2.1 Artist() . . . 19
5.2.2.2 ∼Artist() . . . 19
5.2.3 Documentação das funções . . . 19
5.2.3.1 findAlbumByName() . . . 19
5.2.3.2 findSongByName() . . . 20
5.2.3.3 getAlbums() . . . 20
5.2.3.4 getAlbumsCount() . . . 20
5.2.3.5 getFormattedDuration() . . . 21
5.2.3.6 getGenre() . . . 21
5.2.3.7 getId() . . . 21
5.2.3.8 getName() . . . 21
5.2.3.9 getSongs() . . . 21
5.2.3.10 getSongsCount() . . . 22
5.2.3.11 getTotalDuration() . . . 22
5.2.3.12 hasAlbum() . . . 22
5.2.3.13 hasSong() . . . 22
5.2.3.14 removeAlbum() . . . 22
5.2.3.15 removeSong() . . . 23
5.2.3.16 setAlbum() . . . 23
5.2.3.17 setGenre() . . . 23
5.2.3.18 setName() . . . 24
5.2.3.19 setSong() . . . 24
5.2.3.20 toString() . . . 24
5.2.4 Documentação dos dados membro . . . 24
5.2.4.1 _albums . . . 24
5.2.4.2 _genre . . . 24
5.2.4.3 _id . . . 24
5.2.4.4 _name . . . 25
5.2.4.5 _songs . . . 25
5.3 Referência à classe core::Entity . . . 25
5.3.1 Descrição detalhada . . . 25
5.3.2 Documentação dos Construtores & Destrutor . . . 25
5.3.2.1 ∼Entity() . . . 25
5.3.3 Documentação das funções . . . 26
5.3.3.1 getId() . . . 26

5.3.3.2 operator==() . . . 26
5.3.3.3 setId() . . . 26
5.3.4 Documentação dos dados membro . . . 26
5.3.4.1 id . . . 26
5.4 Referência à classe Player . . . 27
5.4.1 Descrição detalhada . . . 28
5.4.2 Documentação dos Construtores & Destrutor . . . 28
5.4.2.1 Player() . . . 28
5.4.3 Documentação das funções . . . 29
5.4.3.1 advanceToNext() . . . 29
5.4.3.2 clearPlaylist() . . . 29
5.4.3.3 getCurrentSong() . . . 29
5.4.3.4 getPlaylist() . . . 29
5.4.3.5 getPlaylistSize() . . . 29
5.4.3.6 getProgress() . . . 30
5.4.3.7 getVolume() . . . 30
5.4.3.8 goToPrevious() . . . 30
5.4.3.9 hasNext() . . . 30
5.4.3.10 hasPrevious() . . . 30
5.4.3.11 isPaused() . . . 31
5.4.3.12 isPlaying() . . . 31
5.4.3.13 loadAudio() . . . 31
5.4.3.14 mute() . . . 31
5.4.3.15 next() . . . 31
5.4.3.16 pause() . . . 32
5.4.3.17 play() [1/2] . . . 32
5.4.3.18 play() [2/2] . . . 32
5.4.3.19 previous() . . . 32
5.4.3.20 resume() . . . 32
5.4.3.21 setVolume() . . . 32
5.4.3.22 unmute() . . . 33
5.4.4 Documentação dos dados membro . . . 33
5.4.4.1 _currentIndex . . . 33
5.4.4.2 _currentSong . . . 33
5.4.4.3 _isPlaying . . . 33
5.4.4.4 _isStopped . . . 33
5.4.4.5 _playlist . . . 33
5.4.4.6 _volume . . . 33
5.5 Referência à classe Song . . . 34
5.5.1 Descrição detalhada . . . 35
5.5.2 Documentação dos Construtores & Destrutor . . . 35
5.5.2.1 Song() . . . 35

5.5.3 Documentação das funções . . . 35
5.5.3.1 getAlbum() . . . 35
5.5.3.2 getArtist() . . . 36
5.5.3.3 getDuration() . . . 36
5.5.3.4 getFilePath() . . . 36
5.5.3.5 getFormattedDuration() . . . 36
5.5.3.6 getGenre() . . . 36
5.5.3.7 getId() . . . 37
5.5.3.8 getTitle() . . . 37
5.5.3.9 getYear() . . . 37
5.5.3.10 loadMetadata() . . . 37
5.5.3.11 setAlbum() . . . 37
5.5.3.12 setArtist() . . . 38
5.5.3.13 setDuration() . . . 38
5.5.3.14 setGenre() . . . 38
5.5.3.15 setTitle() . . . 38
5.5.3.16 setYear() . . . 39
5.5.3.17 toString() . . . 39
5.5.4 Documentação dos dados membro . . . 39
5.5.4.1 _album . . . 39
5.5.4.2 _artist . . . 39
5.5.4.3 _duration . . . 39
5.5.4.4 _file_path . . . 39
5.5.4.5 _genre . . . 40
5.5.4.6 _id . . . 40
5.5.4.7 _metadata_loaded . . . 40
5.5.4.8 _title . . . 40
5.5.4.9 _year . . . 40

**6 Documentação do ficheiro** **41**

6.1 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ Album.hpp . . . 41
6.1.1 Descrição detalhada . . . 42
6.2 Album.hpp . . . 42
6.3 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ Artist.hpp . . . 43
6.3.1 Descrição detalhada . . . 44
6.4 Artist.hpp . . . 45
6.5 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ EntitiesFWD.hpp . . . 45
6.6 EntitiesFWD.hpp . . . 46
6.7 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ Entity.hpp . . . 46

6.7.1 Descrição detalhada . . . . . 46
6.8 Entity.hpp . . . . . 47
6.9 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ Player.hpp . . . . . 47
6.9.1 Descrição detalhada . . . . . 48
6.10 Player.hpp . . . . . 48
6.11 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/↩ Song.hpp . . . . . 49
6.11.1 Descrição detalhada . . . . . 50
6.12 Song.hpp . . . . . 51

**Índice** **53**

# Capítulo 1

# Índice dos namespaces

## 1.1 Lista de namespaces

Lista dos namespaces com uma breve descrição:

core . . . . . 7

# Capítulo 2

# Índice dos componentes

## 2.1 Lista de componentes

Lista de classes, estruturas, uniões e interfaces com uma breve descrição:


Album
: Representa um álbum musical com suas músicas . . . . . 9

Artist
: Representa um artista musical com suas músicas e álbuns . . . . . 17

core::Entity
: Interface para entidades do sistema Interface que define caracteristicas essenciais para entidades do sistema . . . . . 25

Player
: Controlador de reprodução de áudio com funcionalidades básicas . . . . . 27

Song
: Representa uma música com seus metadados . . . . . 34

# Capítulo 3

# Índice dos ficheiros

## 3.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:


/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp
Definição da classe Album para representar uma coleção de músicas de um artista . . . . . 41
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Artist.hpp
Definição da classe Artist para representar um artista musical . . . . . 43
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/EntitiesFWD.hpp . . . . . 45
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Entity.hpp
Interface para entidades do sistema . . . . . 46
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Player.hpp
Classe responsável pelo controle básico de reprodução de músicas . . . . . 47
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp
Definição da classe Song para representar uma música no sistema . . . . . 49

# Capítulo 4

# Documentação dos namespaces

## 4.1 Referência ao namespace core

### Componentes

- class Entity

  *Interface para entidades do sistema Interface que define caracteristicas essenciais para entidades do sistema.*

# Capítulo 5

# Documentação da classe

## 5.1 Referência à classe Album

Representa um álbum musical com suas músicas.

`#include <Album.hpp>`

### Membros públicos

- Album (const std::string name, const std::string artist, const std::string genre)
  *Construtor da classe Album.*
- Album (unsigned id, const std::string name, const std::string artist, const std::string genre, int year)
  *Construtor completo da classe Album.*
- ~Album ()
  *Destrutor padrão da clase Album.*
- unsigned getId () const
  *Obtém o ID do álbum.*
- std::string getName () const
  *Obtém o nome do álbum.*
- std::string getArtist () const
  *Obtém o nome do artista.*
- std::vector< Song > getSongs () const
  *Obtém a lista de músicas do álbum.*
- std::string getGenre () const
  *Obtém o gênero do álbum.*
- int getYear () const
  *Obtém o ano de lançamento.*
- int getSongCount () const
  *Obtém a quantidade de músicas no álbum.*
- void setArtist (const std::string &artist)
  *Define o artista do álbum.*
- void setGenre (const std::string &genre)
  *Define o gênero do álbum.*
- void setYear (int year)
  *Define o ano de lançamento.*

- void addSong (const Song &song)

  *Adiciona uma música ao álbum.*
- bool removeSong (unsigned songId)

  *Remove uma música do álbum.*
- bool removeSongAt (int index)

  *Remove uma música do álbum por índice.*
- const Song * findSongById (unsigned songId) const

  *Busca uma música no álbum por ID.*
- const Song * findSongByTitle (const std::string &title) const

  *Busca uma música no álbum por título.*
- int calculateTotalDuration () const

  *Calcula a duração total do álbum.*
- std::string getFormattedDuration () const

  *Obtém a duração total formatada.*
- std::string toString () const

  *Obtém informações do álbum em formato de string.*
- Song getNextSong (const Song &current) const

  *Obtém a próxima música na sequência do álbum.*
- Song getPreviousSong (const Song &current) const

  *Obtém a música anterior na sequência do álbum.*
- Song getSongAt (int index) const

  *Obtém uma música por sua posição no álbum.*

**Atributos Privados**

- unsigned _id
- std::string _name
- std::string _artist
- std::vector< Song > * _songs
- std::string _genre
- int _year

### 5.1.1 Descrição detalhada

Representa um álbum musical com suas músicas.

A classe Album armazena informações como nome, artista, gênero e a lista de músicas contidas no álbum. Mantém a ordem original das faixas para reprodução sequencial.

### 5.1.2 Documentação dos Construtores & Destrutor

#### 5.1.2.1 Album() [1/2]

```
Album::Album (
            const std::string name,
            const std::string artist,
            const std::string genre )
```

Construtor da classe Album.

**Parâmetros**

| | |
|---|---|
| *name* | Nome do álbum |
| *artist* | Nome do artista do álbum |
| *genre* | Gênero do álbum |

#### 5.1.2.2 Album() [2/2]

```
Album::Album (
            unsigned id,
            const std::string name,
            const std::string artist,
            const std::string genre,
            int year )
```

Construtor completo da classe Album.

**Parâmetros**

| | |
|---|---|
| *id* | Identificador único |
| *name* | Nome do álbum |
| *artist* | Nome do artista |
| *genre* | Gênero musical |
| *year* | Ano de lançamento |

#### 5.1.2.3 ~Album()

```
Album::~Album ( )
```

Destrutor padrão da clase Album.

**Nota**

Responsável por liberar a memória dos vetores alocados

### 5.1.3 Documentação das funções

#### 5.1.3.1 addSong()

```
void Album::addSong (
            const Song & song )
```

Adiciona uma música ao álbum.

**Parâmetros**

| | |
|---|---|
| *song* | Música a ser adicionada |

#### 5.1.3.2 calculateTotalDuration()

```
int Album::calculateTotalDuration ( ) const
```

Calcula a duração total do álbum.

**Retorna**

Duração total em segundos

#### 5.1.3.3 findSongById()

```
const Song * Album::findSongById (
            unsigned songId ) const
```

Busca uma música no álbum por ID.

**Parâmetros**

| | |
|---|---|
| *songId* | ID da música a buscar |

**Retorna**

Ponteiro para a música encontrada ou nullptr se não encontrada

#### 5.1.3.4 findSongByTitle()

```
const Song * Album::findSongByTitle (
            const std::string & title ) const
```

Busca uma música no álbum por título.

**Parâmetros**

| | |
|---|---|
| *title* | Título da música a buscar |

**Retorna**

Ponteiro para a música encontrada ou nullptr se não encontrada

#### 5.1.3.5 getArtist()

```
std::string Album::getArtist ( ) const
```

Obtém o nome do artista.

**Retorna**

Nome do artista/banda

#### 5.1.3.6 getFormattedDuration()

```
std::string Album::getFormattedDuration ( ) const
```

Obtém a duração total formatada.

**Retorna**

String com duração no formato "HH:MM:SS" ou "MM:SS"

#### 5.1.3.7 getGenre()

```
std::string Album::getGenre ( ) const
```

Obtém o gênero do álbum.

**Retorna**

Gênero musical principal

#### 5.1.3.8 getId()

```
unsigned Album::getId ( ) const
```

Obtém o ID do álbum.

**Retorna**

Identificador único do álbum

#### 5.1.3.9 getName()

```
std::string Album::getName ( ) const
```

Obtém o nome do álbum.

**Retorna**

Nome do álbum

#### 5.1.3.10 getNextSong()

```
Song Album::getNextSong (
            const Song & current ) const
```

Obtém a próxima música na sequência do álbum.

**Parâmetros**

| | |
|---|---|
| *current* | Música atual |

**Retorna**

Próxima música ou a primeira se for a última

#### 5.1.3.11 getPreviousSong()

```
Song Album::getPreviousSong (
            const Song & current ) const
```

Obtém a música anterior na sequência do álbum.

**Parâmetros**

| | |
|---|---|
| *current* | Música atual |

**Retorna**

Música anterior ou a última se for a primeira

#### 5.1.3.12 getSongAt()

```
Song Album::getSongAt (
            int index ) const
```

Obtém uma música por sua posição no álbum.

**Parâmetros**

| | |
|---|---|
| *index* | Índice da música (0-based) |

**Retorna**

Música na posição especificada

**Excepções**

| | |
|---|---|
| *std::out_of_range* | se índice inválido |

#### 5.1.3.13 getSongCount()

```
int Album::getSongCount ( ) const
```

Obtém a quantidade de músicas no álbum.

**Retorna**

Número total de músicas

#### 5.1.3.14 getSongs()

```
std::vector< Song > Album::getSongs ( ) const
```

Obtém a lista de músicas do álbum.

**Retorna**

Vector com as músicas do álbum em ordem

#### 5.1.3.15 getYear()

```
int Album::getYear ( ) const
```

Obtém o ano de lançamento.

**Retorna**

Ano de lançamento do álbum

#### 5.1.3.16 removeSong()

```
bool Album::removeSong (
            unsigned songId )
```

Remove uma música do álbum.

**Parâmetros**

| | |
|---|---|
| *songId* | ID da música a ser removida |

**Retorna**

true se a música foi removida, false se não encontrada

#### 5.1.3.17 removeSongAt()

```
bool Album::removeSongAt (
            int index )
```

Remove uma música do álbum por índice.

**Parâmetros**

| | |
|---|---|
| *index* | Índice da música a ser removida |

**Retorna**

true se a música foi removida, false se índice inválido

#### 5.1.3.18 setArtist()

```
void Album::setArtist (
            const std::string & artist )
```

Define o artista do álbum.

**Parâmetros**

| | |
|---|---|
| *artist* | Novo nome do artista |

#### 5.1.3.19 setGenre()

```
void Album::setGenre (
            const std::string & genre )
```

Define o gênero do álbum.

**Parâmetros**

| | |
|---|---|
| *genre* | Novo gênero musical |

#### 5.1.3.20 setYear()

```
void Album::setYear (
            int year )
```

Define o ano de lançamento.

**Parâmetros**

| | |
|---|---|
| *year* | Novo ano de lançamento |

#### 5.1.3.21 toString()

```
std::string Album::toString ( ) const
```

Obtém informações do álbum em formato de string.

**Retorna**

String com nome, artista, ano e quantidade de músicas

### 5.1.4 Documentação dos dados membro

#### 5.1.4.1 _artist

```
std::string Album::_artist  [private]
```

#### 5.1.4.2 _genre

```
std::string Album::_genre  [private]
```

#### 5.1.4.3 _id

```
unsigned Album::_id  [private]
```

#### 5.1.4.4 _name

```
std::string Album::_name  [private]
```

#### 5.1.4.5 _songs

```
std::vector<Song>* Album::_songs  [private]
```

#### 5.1.4.6 _year

```
int Album::_year  [private]
```

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp

## 5.2 Referência à classe Artist

Representa um artista musical com suas músicas e álbuns.

```
#include <Artist.hpp>
```

### Membros públicos

- Artist (const std::string &name, std::string &genre)

  *Construtor da classe Artist.*
- ~Artist ()

  *Destrutor da classe Artist.*
- unsigned getId () const

  *Obtém o ID do artista.*
- std::string getName () const

  *Obtém o nome do artista.*
- std::string getGenre () const

  *Obtém o gênero musical.*
- std::vector< Song > getSongs () const

  *Obtém todas as músicas do artista.*
- std::vector< Album > getAlbums () const

  *Obtém todos os álbuns do artista.*
- int getSongsCount () const

  *Obtém a quantidade de músicas do artista.*
- int getAlbumsCount () const

  *Obtém a quantidade de álbuns do artista.*
- void setName (const std::string &name)

  *Define o nome do artista.*
- void setGenre (const std::string &genre)

  *Define o gênero musical.*
- void setSong (const Song &song)

  *Adiciona uma música ao artista.*
- void setAlbum (const Album &album)

  *Adiciona um álbum ao artista.*
- bool removeSong (unsigned songId)

  *Remove uma música do artista.*
- bool removeAlbum (unsigned albumId)

  *Remove um álbum do artista.*
- const Song * findSongByName (const std::string &songName)

  *Busca uma música pelo nome.*
- const Album * findAlbumByName (const std::string &albumName)

  *Busca um álbum pelo nome.*
- int getTotalDuration () const

  *Calcula a duração total de todas as músicas.*
- std::string getFormattedDuration () const

  *Obtém a duração total formatada.*
- std::string toString () const

  *Converte o artista para string formatada.*
- bool hasSong () const

  *Verifica se o artista tem músicas.*
- bool hasAlbum () const

  *Verifica se o artista tem álbuns.*

**Atributos Privados**

- unsigned _id
- std::string _name
- std::string _genre
- std::vector< Song > * _songs
- std::vector< Album > * _albums

### 5.2.1 Descrição detalhada

Representa um artista musical com suas músicas e álbuns.

A classe Artist gerencia as informações básicas de um artista e mantém referências para todas as suas músicas e álbuns associados. Permite operações de busca, adição e remoção de itens do catálogo.

### 5.2.2 Documentação dos Construtores & Destrutor

#### 5.2.2.1 Artist()

```
Artist::Artist (
            const std::string & name,
            std::string & genre )
```

Construtor da classe Artist.

**Parâmetros**

| | |
|---|---|
| *name* | Nome do artista |
| *genre* | Gênero musical do artista |

#### 5.2.2.2 ~Artist()

```
Artist::~Artist ( )
```

Destrutor da classe Artist.

**Nota**

Responsável por liberar a memória dos vetores alocados

### 5.2.3 Documentação das funções

#### 5.2.3.1 findAlbumByName()

```
const Album * Artist::findAlbumByName (
            const std::string & albumName )
```

Busca um álbum pelo nome.

**Parâmetros**

| | |
|---|---|
| *albumName* | Nome do álbum a buscar |

**Retorna**

Ponteiro para o álbum encontrado ou nullptr se não encontrado

#### 5.2.3.2 findSongByName()

```
const Song * Artist::findSongByName (
            const std::string & songName )
```

Busca uma música pelo nome.

**Parâmetros**

| | |
|---|---|
| *songName* | Nome da música a buscar |

**Retorna**

Ponteiro para a música encontrada ou nullptr se não encontrada

#### 5.2.3.3 getAlbums()

```
std::vector< Album > Artist::getAlbums ( ) const
```

Obtém todos os álbuns do artista.

**Retorna**

Vector com todos os álbuns do artista

#### 5.2.3.4 getAlbumsCount()

```
int Artist::getAlbumsCount ( ) const
```

Obtém a quantidade de álbuns do artista.

**Retorna**

Número total de álbuns

#### 5.2.3.5 getFormattedDuration()

```
std::string Artist::getFormattedDuration ( ) const
```

Obtém a duração total formatada.

**Retorna**

String com duração no formato "HH:MM:SS" ou "MM:SS"

#### 5.2.3.6 getGenre()

```
std::string Artist::getGenre ( ) const
```

Obtém o gênero musical.

**Retorna**

Gênero musical principal do artista

#### 5.2.3.7 getId()

```
unsigned Artist::getId ( ) const
```

Obtém o ID do artista.

**Retorna**

Identificador único do artista

#### 5.2.3.8 getName()

```
std::string Artist::getName ( ) const
```

Obtém o nome do artista.

**Retorna**

Nome do artista/banda

#### 5.2.3.9 getSongs()

```
std::vector< Song > Artist::getSongs ( ) const
```

Obtém todas as músicas do artista.

**Retorna**

Vector com todas as músicas do artista

#### 5.2.3.10 getSongsCount()

```
int Artist::getSongsCount ( ) const
```

Obtém a quantidade de músicas do artista.

**Retorna**

Número total de músicas

#### 5.2.3.11 getTotalDuration()

```
int Artist::getTotalDuration ( ) const
```

Calcula a duração total de todas as músicas.

**Retorna**

Duração total em segundos

#### 5.2.3.12 hasAlbum()

```
bool Artist::hasAlbum ( ) const
```

Verifica se o artista tem álbuns.

**Retorna**

true se possui pelo menos um álbum, false caso contrário

#### 5.2.3.13 hasSong()

```
bool Artist::hasSong ( ) const
```

Verifica se o artista tem músicas.

**Retorna**

true se possui pelo menos uma música, false caso contrário

#### 5.2.3.14 removeAlbum()

```
bool Artist::removeAlbum (
            unsigned albumId )
```

Remove um álbum do artista.

**Parâmetros**

| | |
|---|---|
| *albumId* | ID do álbum a ser removido |

**Retorna**

true se o álbum foi removido com sucesso, false caso contrário

#### 5.2.3.15 removeSong()

```
bool Artist::removeSong (
            unsigned songId )
```

Remove uma música do artista.

**Parâmetros**

| | |
|---|---|
| *songId* | ID da música a ser removida |

**Retorna**

true se a música foi removida com sucesso, false caso contrário

#### 5.2.3.16 setAlbum()

```
void Artist::setAlbum (
            const Album & album )
```

Adiciona um álbum ao artista.

**Parâmetros**

| | |
|---|---|
| *album* | Álbum a ser adicionado |

#### 5.2.3.17 setGenre()

```
void Artist::setGenre (
            const std::string & genre )
```

Define o gênero musical.

**Parâmetros**

| | |
|---|---|
| *genre* | Novo gênero musical |

#### 5.2.3.18 setName()

```
void Artist::setName (
            const std::string & name )
```

Define o nome do artista.

**Parâmetros**

| | |
|---|---|
| *name* | Novo nome do artista |

#### 5.2.3.19 setSong()

```
void Artist::setSong (
            const Song & song )
```

Adiciona uma música ao artista.

**Parâmetros**

| | |
|---|---|
| *song* | Música a ser adicionada |

#### 5.2.3.20 toString()

```
std::string Artist::toString ( ) const
```

Converte o artista para string formatada.

**Retorna**

String com nome, gênero e estatísticas do artista

### 5.2.4 Documentação dos dados membro

#### 5.2.4.1 _albums

```
std::vector<Album>* Artist::_albums  [private]
```

#### 5.2.4.2 _genre

```
std::string Artist::_genre  [private]
```

#### 5.2.4.3 _id

```
unsigned Artist::_id  [private]
```

#### 5.2.4.4 _name

```
std::string Artist::_name  [private]
```

#### 5.2.4.5 _songs

```
std::vector<Song>* Artist::_songs  [private]
```

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Artist.hpp

## 5.3 Referência à classe core::Entity

Interface para entidades do sistema Interface que define caracteristicas essenciais para entidades do sistema.

```
#include <Entity.hpp>
```

### Membros públicos

- virtual ~Entity ()=default
- unsigned getId () const

  *Obtém o ID da entidade.*
- void setId (unsigned id)

  *Define o ID da entidade.*
- virtual bool operator== (const Entity &other) const =0

  *Compara duas entidades.*

### Atributos Protegidos

- unsigned id = 0

### 5.3.1 Descrição detalhada

Interface para entidades do sistema Interface que define caracteristicas essenciais para entidades do sistema.

### 5.3.2 Documentação dos Construtores & Destrutor

#### 5.3.2.1 ~Entity()

```
virtual core::Entity::~Entity ( )  [virtual], [default]
```

### 5.3.3 Documentação das funções

#### 5.3.3.1 getId()

```
unsigned core::Entity::getId ( ) const
```

Obtém o ID da entidade.

**Retorna**

ID da entidade

#### 5.3.3.2 operator==()

```
virtual bool core::Entity::operator== (
            const Entity & other ) const  [pure virtual]
```

Compara duas entidades.

**Parâmetros**

| | |
|---|---|
| *other* | Entidade a ser comparada |

**Retorna**

true se as entidades forem iguais, false caso contrário

#### 5.3.3.3 setId()

```
void core::Entity::setId (
            unsigned id )
```

Define o ID da entidade.

**Parâmetros**

| | |
|---|---|
| *id* | Novo ID da entidade |

### 5.3.4 Documentação dos dados membro

#### 5.3.4.1 id

```
unsigned core::Entity::id = 0  [protected]
```

ID da entidade

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

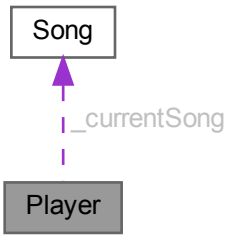
- /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Entity.hpp

## 5.4 Referência à classe Player

Controlador de reprodução de áudio com funcionalidades básicas.

`#include <Player.hpp>`

Diagrama de colaboração para Player:




### Membros públicos

- Player ()
  *Construtor da classe Player.*
- void play (const Song &song)
  *Reproduz uma música específica.*
- void play (const std::vector< Song > &songs)
  *Reproduz uma lista de músicas.*
- void pause ()
  *Pausa a reprodução atual.*
- void resume ()
  *Retoma a reprodução pausada.*
- void next ()
  *Avança para a próxima música na playlist.*
- void previous ()
  *Volta para a música anterior na playlist.*
- void setVolume (float &volume)
  *Define o nível de volume do player.*
- float getVolume () const
  *Obtém o nível de volume atual.*
- void mute ()
  *Silencia o player.*
- void unmute ()
  *Restaura o volume anterior.*
- Song getCurrentSong () const
  *Obtém a música atualmente em reprodução.*
- bool isPlaying () const
  *Verifica se o player está reproduzindo.*
- bool isPaused () const

*Verifica se o player está pausado.*

- float getProgress () const

  *Obtém o progresso atual da reprodução.*

- int getPlaylistSize () const

  *Obtém o tamanho da playlist atual.*

- void clearPlaylist ()

  *Limpa toda a playlist.*

- std::vector< Song > getPlaylist () const

  *Obtém uma cópia da playlist atual.*

- void advanceToNext ()

  *Avança para a próxima música na sequência.*

- void goToPrevious ()

  *Retrocede para a música anterior na sequência.*

- bool hasNext () const

  *Verifica se existe próxima música na playlist.*

- bool hasPrevious () const

  *Verifica se existe música anterior na playlist.*

- void loadAudio (const Song &song)

  *Carrega o áudio de uma música para reprodução.*

**Atributos Privados**

- std::vector< Song > _playlist
- Song * _currentSong
- int _currentIndex
- bool _isPlaying
- bool _isStopped
- float _volume

### 5.4.1 Descrição detalhada

Controlador de reprodução de áudio com funcionalidades básicas.

Gerencia uma playlist de músicas e fornece controles essenciais para reprodução sequencial. Esta é uma versão inicial que será expandida posteriormente para suportar a interface IPlayable.

### 5.4.2 Documentação dos Construtores & Destrutor

#### 5.4.2.1 Player()

```
Player::Player ( )
```

Construtor da classe Player.

Inicializa o player com estado parado e volume máximo.

### 5.4.3 Documentação das funções

#### 5.4.3.1 advanceToNext()

```
void Player::advanceToNext ( )
```

Avança para a próxima música na sequência.

#### 5.4.3.2 clearPlaylist()

```
void Player::clearPlaylist ( )
```

Limpa toda a playlist.

Remove todas as músicas da playlist e para a reprodução atual.

#### 5.4.3.3 getCurrentSong()

```
Song Player::getCurrentSong ( ) const
```

Obtém a música atualmente em reprodução.

**Retorna**

Música atual ou música inválida se não houver reprodução

#### 5.4.3.4 getPlaylist()

```
std::vector< Song > Player::getPlaylist ( ) const
```

Obtém uma cópia da playlist atual.

**Retorna**

Vector com todas as músicas da playlist

#### 5.4.3.5 getPlaylistSize()

```
int Player::getPlaylistSize ( ) const
```

Obtém o tamanho da playlist atual.

**Retorna**

Número de músicas na playlist

#### 5.4.3.6 getProgress()

```
float Player::getProgress ( ) const
```

Obtém o progresso atual da reprodução.

**Retorna**

Progresso entre 0.0 (início) e 1.0 (fim) da música atual

#### 5.4.3.7 getVolume()

```
float Player::getVolume ( ) const
```

Obtém o nível de volume atual.

**Retorna**

Volume atual entre 0.0 e 1.0

#### 5.4.3.8 goToPrevious()

```
void Player::goToPrevious ( )
```

Retrocede para a música anterior na sequência.

#### 5.4.3.9 hasNext()

```
bool Player::hasNext ( ) const
```

Verifica se existe próxima música na playlist.

**Retorna**

true se há próxima música, false caso contrário

#### 5.4.3.10 hasPrevious()

```
bool Player::hasPrevious ( ) const
```

Verifica se existe música anterior na playlist.

**Retorna**

true se há música anterior, false caso contrário

#### 5.4.3.11 isPaused()

```
bool Player::isPaused ( ) const
```

Verifica se o player está pausado.

**Retorna**

true se está pausado, false se reproduzindo ou parado

#### 5.4.3.12 isPlaying()

```
bool Player::isPlaying ( ) const
```

Verifica se o player está reproduzindo.

**Retorna**

true se está reproduzindo, false se pausado ou parado

#### 5.4.3.13 loadAudio()

```
void Player::loadAudio (
            const Song & song )
```

Carrega o áudio de uma música para reprodução.

**Parâmetros**

| | |
|---|---|
| *song* | Música a ser carregada |

#### 5.4.3.14 mute()

```
void Player::mute ( )
```

Silencia o player.

Define o volume para 0.0 sem perder a configuração anterior.

#### 5.4.3.15 next()

```
void Player::next ( )
```

Avança para a próxima música na playlist.

Se estiver na última música, pode parar ou voltar para o início dependendo da implementação.

#### 5.4.3.16 pause()

```
void Player::pause ( )
```

Pausa a reprodução atual.

Mantém a música atual carregada mas para a reprodução.

#### 5.4.3.17 play() [1/2]

```
void Player::play (
            const Song & song )
```

Reproduz uma música específica.

**Parâmetros**

| | |
|---|---|
| *song* | Música a ser reproduzida |

#### 5.4.3.18 play() [2/2]

```
void Player::play (
            const std::vector< Song > & songs )
```

Reproduz uma lista de músicas.

**Parâmetros**

| | |
|---|---|
| *songs* | Vector de músicas a serem reproduzidas em sequência |

#### 5.4.3.19 previous()

```
void Player::previous ( )
```

Volta para a música anterior na playlist.

Se estiver na primeira música, pode parar ou ir para a última dependendo da implementação.

#### 5.4.3.20 resume()

```
void Player::resume ( )
```

Retoma a reprodução pausada.

Continua a reprodução da música atual de onde parou.

#### 5.4.3.21 setVolume()

```
void Player::setVolume (
            float & volume )
```

Define o nível de volume do player.

**Parâmetros**

| | |
|---|---|
| *volume* | Novo nível de volume entre 0.0 (mudo) e 1.0 (máximo) |

#### 5.4.3.22 unmute()

```
void Player::unmute ( )
```

Restaura o volume anterior.

Retorna o volume para o nível definido antes do mute.

### 5.4.4 Documentação dos dados membro

#### 5.4.4.1 _currentIndex

```
int Player::_currentIndex  [private]
```

#### 5.4.4.2 _currentSong

```
Song* Player::_currentSong  [private]
```

#### 5.4.4.3 _isPlaying

```
bool Player::_isPlaying  [private]
```

#### 5.4.4.4 _isStopped

```
bool Player::_isStopped  [private]
```

#### 5.4.4.5 _playlist

```
std::vector<Song> Player::_playlist  [private]
```

#### 5.4.4.6 _volume

```
float Player::_volume  [private]
```

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Player.hpp

## 5.5 Referência à classe Song

Representa uma música com seus metadados.

`#include <Song.hpp>`

### Membros públicos

- Song (int id, const std::string &file_path, const std::string &title, const std::string &artist)
  *Construtor da classe Song.*
- int getId () const
  *Obtém o ID da música.*
- std::string getFilePath () const
  *Obtém o caminho do arquivo.*
- std::string getTitle () const
  *Obtém o título da música.*
- std::string getArtist () const
  *Obtém o artista.*
- std::string getAlbum () const
  *Obtém o álbum.*
- int getDuration () const
  *Obtém a duração.*
- std::string getGenre () const
  *Obtém o gênero.*
- int getYear () const
  *Obtém o ano de lançamento.*
- void setTitle (const std::string &title)
  *Define o título da música.*
- void setArtist (const std::string &artist)
  *Define o artista.*
- void setAlbum (const std::string &album)
  *Define o álbum.*
- void setDuration (int duration)
  *Define a duração.*
- void setGenre (const std::string &genre)
  *Define o gênero.*
- void setYear (int year)
  *Define o ano de lançamento.*
- bool loadMetadata ()
  *Carrega metadados do arquivo de áudio.*
- std::string getFormattedDuration () const
  *Obtém a duração formatada.*
- std::string toString () const
  *Converte a música para string.*

**Atributos Privados**

- unsigned _id
- std::string _file_path
- std::string _title
- std::string _artist
- std::string _album
- int _duration
- std::string _genre
- int _year
- bool _metadata_loaded

### 5.5.1 Descrição detalhada

Representa uma música com seus metadados.

A classe Song armazena informações como título, artista, álbum, duração, gênero e ano de lançamento. Pode ser expandida para usar a biblioteca TagLib para extração automática de metadados de arquivos de áudio. Ademais, deve ser implementado a interface IPlayable

### 5.5.2 Documentação dos Construtores & Destrutor

#### 5.5.2.1 Song()

```
Song::Song (
            int id,
            const std::string & file_path,
            const std::string & title,
            const std::string & artist )
```

Construtor da classe Song.

**Parâmetros**

| | |
|---|---|
| *id* | Identificador único da música |
| *file_path* | Caminho do arquivo de áudio |
| *title* | Título da música |
| *artist* | Artista/banda |

### 5.5.3 Documentação das funções

#### 5.5.3.1 getAlbum()

```
std::string Song::getAlbum ( ) const
```

Obtém o álbum.

**Retorna**

Nome do álbum

#### 5.5.3.2 getArtist()

```
std::string Song::getArtist ( ) const
```

Obtém o artista.

**Retorna**

Nome do artista/banda

#### 5.5.3.3 getDuration()

```
int Song::getDuration ( ) const
```

Obtém a duração.

**Retorna**

Duração em segundos

#### 5.5.3.4 getFilePath()

```
std::string Song::getFilePath ( ) const
```

Obtém o caminho do arquivo.

**Retorna**

Caminho completo do arquivo de áudio

#### 5.5.3.5 getFormattedDuration()

```
std::string Song::getFormattedDuration ( ) const
```

Obtém a duração formatada.

**Retorna**

String com a duração no formato "MM:SS"

#### 5.5.3.6 getGenre()

```
std::string Song::getGenre ( ) const
```

Obtém o gênero.

**Retorna**

Gênero musical

#### 5.5.3.7 getId()

```
int Song::getId ( ) const
```

Obtém o ID da música.

**Retorna**

Identificador único da música

#### 5.5.3.8 getTitle()

```
std::string Song::getTitle ( ) const
```

Obtém o título da música.

**Retorna**

Título da música

#### 5.5.3.9 getYear()

```
int Song::getYear ( ) const
```

Obtém o ano de lançamento.

**Retorna**

Ano de lançamento

#### 5.5.3.10 loadMetadata()

```
bool Song::loadMetadata ( )
```

Carrega metadados do arquivo de áudio.

**Retorna**

true se os metadados foram carregados com sucesso, false caso contrário

**Nota**

Futuramente será implementado com a biblioteca TagLib

#### 5.5.3.11 setAlbum()

```
void Song::setAlbum (
            const std::string & album )
```

Define o álbum.

**Parâmetros**

| | |
|---|---|
| *album* | Novo álbum |

#### 5.5.3.12 setArtist()

```
void Song::setArtist (
            const std::string & artist )
```

Define o artista.

**Parâmetros**

| | |
|---|---|
| *artist* | Novo artista |

#### 5.5.3.13 setDuration()

```
void Song::setDuration (
            int duration )
```

Define a duração.

**Parâmetros**

| | |
|---|---|
| *duration* | Nova duração em segundos |

#### 5.5.3.14 setGenre()

```
void Song::setGenre (
            const std::string & genre )
```

Define o gênero.

**Parâmetros**

| | |
|---|---|
| *genre* | Novo gênero |

#### 5.5.3.15 setTitle()

```
void Song::setTitle (
            const std::string & title )
```

Define o título da música.

**Parâmetros**

| | |
|---|---|
| *title* | Novo título |

#### 5.5.3.16 setYear()

```
void Song::setYear (
            int year )
```

Define o ano de lançamento.

**Parâmetros**

| | |
|---|---|
| *year* | Novo ano |

#### 5.5.3.17 toString()

```
std::string Song::toString ( ) const
```

Converte a música para string.

**Retorna**

String com todas as informações da música formatadas

### 5.5.4 Documentação dos dados membro

#### 5.5.4.1 _album

```
std::string Song::_album  [private]
```

#### 5.5.4.2 _artist

```
std::string Song::_artist  [private]
```

#### 5.5.4.3 _duration

```
int Song::_duration  [private]
```

#### 5.5.4.4 _file_path

```
std::string Song::_file_path  [private]
```

#### 5.5.4.5 _genre

```
std::string Song::_genre  [private]
```

#### 5.5.4.6 _id

```
unsigned Song::_id  [private]
```

#### 5.5.4.7 _metadata_loaded

```
bool Song::_metadata_loaded  [private]
```

#### 5.5.4.8 _title

```
std::string Song::_title  [private]
```

#### 5.5.4.9 _year

```
int Song::_year  [private]
```

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

- /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp

# Capítulo 6

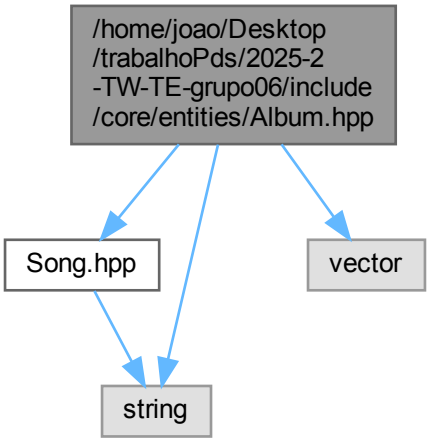
# Documentação do ficheiro

## 6.1 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-↩ TE-grupo06/include/core/entities/Album.hpp

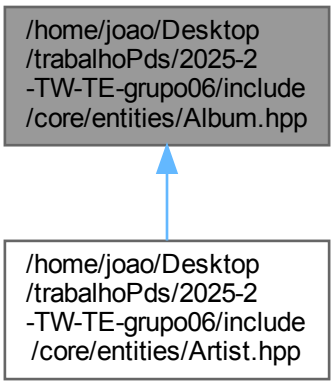
Definição da classe Album para representar uma coleção de músicas de um artista.

```
#include "Song.hpp"
#include <string>
#include <vector>
```

Diagrama de dependências de inclusão para Album.hpp:



Este grafo mostra quais são os ficheiros que incluem directamente ou indirectamente este ficheiro:




**Componentes**

- class Album

  *Representa um álbum musical com suas músicas.*

### 6.1.1 Descrição detalhada

Definição da classe Album para representar uma coleção de músicas de um artista.

Esta classe armazena um álbum de um artista, e fornece métodos para manipulação e exibição das músicas incluídas no álbum. Ademais, deve ser implementado a interface IPlayable

**Autor**

Joao Tavares

**Data**

2025-10-09

## 6.2 Album.hpp

Ir para a documentação deste ficheiro.

```
00001
00014 #ifndef ALBUM_HPP
00015 #define ALBUM_HPP
00016
00017 #include "Song.hpp"
00018 #include <string>
00019 #include <vector>
00028 class Album {
00029 private:
00030   unsigned _id;
```

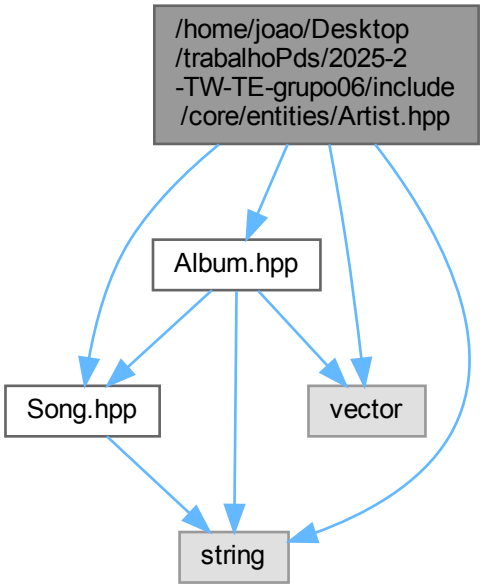
```
00031   std::string _name;
00032   std::string _artist;
00033   std::vector<Song> *_songs;
00034   std::string _genre;
00035   int _year;
00036
00037 public:
00044   Album(const std::string name, const std::string artist,
00045         const std::string genre);
00046
00055   Album(unsigned id, const std::string name, const std::string artist,
00056         const std::string genre, int year);
00061   ~Album();
00062
00063   // Getters
00064
00069   unsigned getId() const;
00070
00075   std::string getName() const;
00076
00081   std::string getArtist() const;
00082
00087   std::vector<Song> getSongs() const;
00088
00093   std::string getGenre() const;
00094
00099   int getYear() const;
00100
00105   int getSongCount() const;
00106
00107   // Setters
00108
00113   void setArtist(const std::string &artist);
00114
00119   void setGenre(const std::string &genre);
00120
00125   void setYear(int year);
00126
00131   void addSong(const Song &song);
00132
00138   bool removeSong(unsigned songId);
00139
00145   bool removeSongAt(int index);
00146
00152   const Song *findSongById(unsigned songId) const;
00153
00159   const Song *findSongByTitle(const std::string &title) const;
00160
00161   // Metodos
00162
00167   int calculateTotalDuration() const;
00168
00173   std::string getFormattedDuration() const;
00174
00179   std::string toString() const;
00180
00186   Song getNextSong(const Song &current) const;
00187
00193   Song getPreviousSong(const Song &current) const;
00194
00201   Song getSongAt(int index) const;
00202 };
00203
00204 #endif // ALBUM_HPP
```

## 6.3 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Artist.hpp

Definição da classe Artist para representar um artista musical.

```
#include "Album.hpp"
#include "Song.hpp"
#include <string>
```

`#include <vector>`
Diagrama de dependências de inclusão para Artist.hpp:




**Componentes**

- class Artist

  *Representa um artista musical com suas músicas e álbuns.*

### 6.3.1 Descrição detalhada

Definição da classe Artist para representar um artista musical.

Esta classe armazena informações de um artista musical, incluindo seu nome, gênero, e coleções de músicas e álbuns. Fornece métodos para gerenciamento e consulta do catálogo do artista.

**Autor**

Joao Tavares

**Data**

2025-11-09

## 6.4 Artist.hpp

Ir para a documentação deste ficheiro.

```
00001
00013 #ifndef ARTIST_HPP
00014 #define ARTIST_HPP
00015
00016 #include "Album.hpp"
00017 #include "Song.hpp"
00018 #include <string>
00019 #include <vector>
00020
00029 class Artist {
00030 private:
00031   unsigned _id;
00032   std::string _name;
00033   std::string _genre;
00034   std::vector<Song> *_songs;
00035   std::vector<Album> *_albums;
00036
00037 public:
00043   Artist(const std::string &name, std::string &genre);
00044
00049   ~Artist();
00050
00055   unsigned getId() const;
00056
00061   std::string getName() const;
00062
00067   std::string getGenre() const;
00068
00073   std::vector<Song> getSongs() const;
00074
00079   std::vector<Album> getAlbums() const;
00080
00085   int getSongsCount() const;
00086
00091   int getAlbumsCount() const;
00092
00097   void setName(const std::string &name);
00098
00103   void setGenre(const std::string &genre);
00104
00109   void setSong(const Song &song);
00110
00115   void setAlbum(const Album &album);
00116
00122   bool removeSong(unsigned songId);
00123
00129   bool removeAlbum(unsigned albumId);
00130
00136   const Song *findSongByName(const std::string &songName);
00137
00143   const Album *findAlbumByName(const std::string &albumName);
00144
00149   int getTotalDuration() const;
00150
00155   std::string getFormattedDuration() const;
00156
00161   std::string toString() const;
00162
00167   bool hasSong() const;
00168
00173   bool hasAlbum() const;
00174 };
00175
00176 #endif // ARTIST_HPP
```

## 6.5 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-↩ TE-grupo06/include/core/entities/EntitiesFWD.hpp

### Namespaces

- namespace core

## 6.6 EntitiesFWD.hpp

Ir para a documentação deste ficheiro.

```
00001 // Arquivo de foward declaration para evitar dependencias circulares
00002 // TODO remover depois
00003
00004 namespace core {
00005
00006     class User;
00007     class Song;
00008     class Album;
00009     class Artist;
00010     class Playlist;
00011     class HistoryPlayback;
00012
00013 }
```

## 6.7 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Entity.hpp

Interface para entidades do sistema.

### Componentes

- class core::Entity

  *Interface para entidades do sistema Interface que define caracteristicas essenciais para entidades do sistema.*

### Namespaces

- namespace core

### 6.7.1 Descrição detalhada

Interface para entidades do sistema.

Interface que define caracteristicas essenciais para entidades do sistema.

**Autor**

Eloy Maciel

**Data**

2025-10-09

## 6.8 Entity.hpp

Ir para a documentação deste ficheiro.

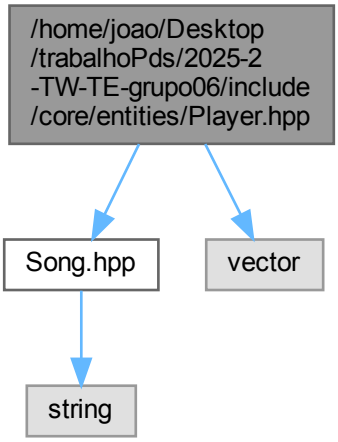
```
00001
00011 #ifndef ENTITY_HPP
00012 #define ENTITY_HPP
00013
00014 namespace core {
00015
00020 class Entity {
00021 protected:
00022   unsigned id = 0;
00024 public:
00025   virtual ~Entity() = default;
00030   unsigned getId() const;
00031
00036   void setId(unsigned id);
00037
00043   virtual bool operator==(const Entity &other) const = 0;
00044 };
00045 } // namespace core
00046
00047 #endif
```

## 6.9 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-↩ TE-grupo06/include/core/entities/Player.hpp

Classe responsável pelo controle básico de reprodução de músicas.

```
#include "Song.hpp"
#include <vector>
```

Diagrama de dependências de inclusão para Player.hpp:




### Componentes

- class Player

  *Controlador de reprodução de áudio com funcionalidades básicas.*

### 6.9.1 Descrição detalhada

Classe responsável pelo controle básico de reprodução de músicas.

Fornece funcionalidades essenciais para reprodução de músicas individuais e playlists, com controles de play, pause, next, previous e volume.

**Autor**

Joao Tavares

**Data**

2025-11-09

## 6.10 Player.hpp

Ir para a documentação deste ficheiro.

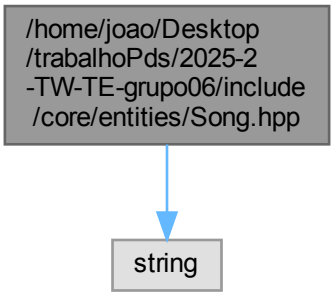
```
00001
00012 #ifndef PLAYER_HPP
00013 #define PLAYER_HPP
00014
00015 #include "Song.hpp"
00016 #include <vector>
00017
00026 class Player {
00027 private:
00028   std::vector<Song> _playlist;
00029   Song *_currentSong;
00030   int _currentIndex;
00031   bool _isPlaying;
00032   bool _isStopped;
00033   float _volume;
00034
00035 public:
00041   Player();
00042
00047   void play(const Song &song);
00048
00053   void play(const std::vector<Song> &songs);
00054
00060   void pause();
00061
00067   void resume();
00068
00075   void next();
00076
00083   void previous();
00084
00089   void setVolume(float &volume);
00090
00095   float getVolume() const;
00096
00102   void mute();
00103
00109   void unmute();
00110
00115   Song getCurrentSong() const;
00116
00121   bool isPlaying() const;
00122
00127   bool isPaused() const;
00128
00133   float getProgress() const;
00134
00139   int getPlaylistSize() const;
00140
00146   void clearPlaylist();
00147
00152   std::vector<Song> getPlaylist() const;
00153
00158   void advanceToNext();
00159
```

```
00164   void goToPrevious();
00165
00171   bool hasNext() const;
00172
00178   bool hasPrevious() const;
00179
00185   void loadAudio(const Song &song);
00186 };
00187
00188 #endif // PLAYER_HPP
```

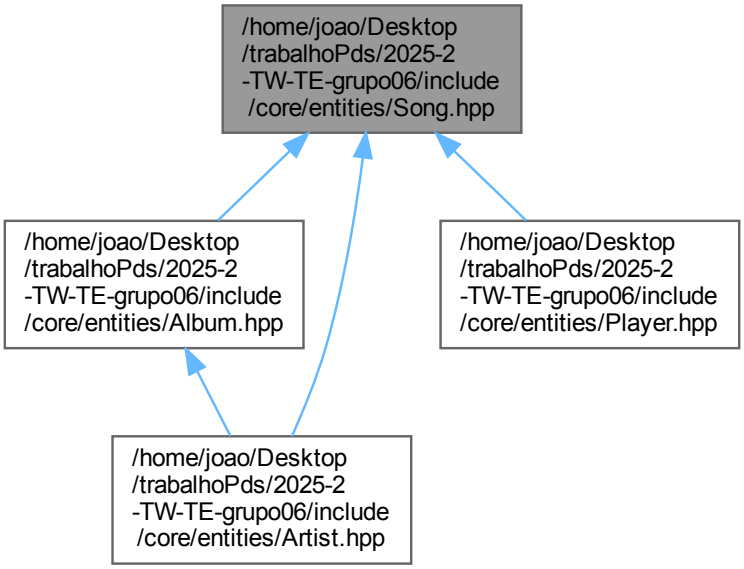
## 6.11 Referência ao ficheiro /home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp

Definição da classe Song para representar uma música no sistema.

`#include <string>`
Diagrama de dependências de inclusão para Song.hpp:



Este grafo mostra quais são os ficheiros que incluem directamente ou indirectamente este ficheiro:




**Componentes**

- class Song

  *Representa uma música com seus metadados.*

### 6.11.1 Descrição detalhada

Definição da classe Song para representar uma música no sistema.

Esta classe armazena os metadados de uma música e fornece métodos para manipulação e exibição dessas informações. Futuramente será integrada com a biblioteca TagLib para extração automática de metadados.

**Autor**

Joao Tavares

**Data**

2025-10-09

# 6.12 Song.hpp

Ir para a documentação deste ficheiro.

```
00001
00013 #ifndef SONG_HPP
00014 #define SONG_HPP
00015
00016 #include <string>
00017
00027 class Song {
00028 private:
00029   unsigned _id;
00030   std::string _file_path;
00031   std::string _title;
00032   std::string _artist;
00033   std::string _album;
00034   int _duration;
00035   std::string _genre;
00036   int _year;
00037   bool _metadata_loaded;
00038
00039 public:
00047   Song(int id, const std::string &file_path, const std::string &title,
00048        const std::string &artist);
00049
00050   // Getters
00055   int getId() const;
00056
00061   std::string getFilePath() const;
00066   std::string getTitle() const;
00071   std::string getArtist() const;
00076   std::string getAlbum() const;
00081   int getDuration() const;
00086   std::string getGenre() const;
00091   int getYear() const;
00092   // Setters
00097   void setTitle(const std::string &title);
00102   void setArtist(const std::string &artist);
00107   void setAlbum(const std::string &album);
00112   void setDuration(int duration);
00117   void setGenre(const std::string &genre);
00122   void setYear(int year);
00123   // Métodos
00130   bool loadMetadata();
00131
00136   std::string getFormattedDuration() const;
00137
00142   std::string toString() const;
00143
00144   // Operações no banco de dados será responsabilidade da classe?
00145 };
00146
00147 #endif // SONG_HPP
```

# Índice


/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp, 41, 42
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Artist.hpp, 43, 45
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/EntitiesFWD.hpp, 45, 46
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Entity.hpp, 46, 47
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Player.hpp, 47, 48
/home/joao/Desktop/trabalhoPds/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp, 49, 51
_album
Song, 39
_albums
Artist, 24
_artist
Album, 17
Song, 39
_currentIndex
Player, 33
_currentSong
Player, 33
_duration
Song, 39
_file_path
Song, 39
_genre
Album, 17
Artist, 24
Song, 39
_id
Album, 17
Artist, 24
Song, 40
_isPlaying
Player, 33
_isStopped
Player, 33
_metadata_loaded
Song, 40
_name
Album, 17
Artist, 24
_playlist
Player, 33
_songs
Album, 17
Artist, 25
_title
Song, 40
_volume
Player, 33
_year
Album, 17
Song, 40
~Album
Album, 11
~Artist
Artist, 19
~Entity
core::Entity, 25

addSong
Album, 11
advanceToNext
Player, 29
Album, 9
_artist, 17
_genre, 17
_id, 17
_name, 17
_songs, 17
_year, 17
~Album, 11
addSong, 11
Album, 10, 11
calculateTotalDuration, 12
findSongById, 12
findSongByTitle, 12
getArtist, 12
getFormattedDuration, 12
getGenre, 13
getId, 13
getName, 13
getNextSong, 13
getPreviousSong, 14
getSongAt, 14
getSongCount, 14
getSongs, 15
getYear, 15
removeSong, 15

removeSongAt, 15
setArtist, 16
setGenre, 16
setYear, 16
toString, 16
Artist, 17
_albums, 24
_genre, 24
_id, 24
_name, 24
_songs, 25
~Artist, 19
Artist, 19
findAlbumByName, 19
findSongByName, 20
getAlbums, 20
getAlbumsCount, 20
getFormattedDuration, 20
getGenre, 21
getId, 21
getName, 21
getSongs, 21
getSongsCount, 21
getTotalDuration, 22
hasAlbum, 22
hasSong, 22
removeAlbum, 22
removeSong, 23
setAlbum, 23
setGenre, 23
setName, 24
setSong, 24
toString, 24

calculateTotalDuration
Album, 12
clearPlaylist
Player, 29
core, 7
core::Entity, 25
~Entity, 25
getId, 26
id, 26
operator==, 26
setId, 26

findAlbumByName
Artist, 19
findSongById
Album, 12
findSongByName
Artist, 20
findSongByTitle
Album, 12

getAlbum
Song, 35
getAlbums
Artist, 20
getAlbumsCount
Artist, 20
getArtist
Album, 12
Song, 35
getCurrentSong
Player, 29
getDuration
Song, 36
getFilePath
Song, 36
getFormattedDuration
Album, 12
Artist, 20
Song, 36
getGenre
Album, 13
Artist, 21
Song, 36
getId
Album, 13
Artist, 21
core::Entity, 26
Song, 36
getName
Album, 13
Artist, 21
getNextSong
Album, 13
getPlaylist
Player, 29
getPlaylistSize
Player, 29
getPreviousSong
Album, 14
getProgress
Player, 29
getSongAt
Album, 14
getSongCount
Album, 14
getSongs
Album, 15
Artist, 21
getSongsCount
Artist, 21
getTitle
Song, 37
getTotalDuration
Artist, 22
getVolume
Player, 30
getYear
Album, 15
Song, 37
goToPrevious
Player, 30

hasAlbum

Artist, 22
hasNext
Player, 30
hasPrevious
Player, 30
hasSong
Artist, 22

id
core::Entity, 26
isPaused
Player, 30
isPlaying
Player, 31

loadAudio
Player, 31
loadMetadata
Song, 37

mute
Player, 31

next
Player, 31

operator==
core::Entity, 26

pause
Player, 31
play
Player, 32
Player, 27
_currentIndex, 33
_currentSong, 33
_isPlaying, 33
_isStopped, 33
_playlist, 33
_volume, 33
advanceToNext, 29
clearPlaylist, 29
getCurrentSong, 29
getPlaylist, 29
getPlaylistSize, 29
getProgress, 29
getVolume, 30
goToPrevious, 30
hasNext, 30
hasPrevious, 30
isPaused, 30
isPlaying, 31
loadAudio, 31
mute, 31
next, 31
pause, 31
play, 32
Player, 28
previous, 32
resume, 32
setVolume, 32
unmute, 33
previous
Player, 32

removeAlbum
Artist, 22
removeSong
Album, 15
Artist, 23
removeSongAt
Album, 15
resume
Player, 32

setAlbum
Artist, 23
Song, 37
setArtist
Album, 16
Song, 38
setDuration
Song, 38
setGenre
Album, 16
Artist, 23
Song, 38
setId
core::Entity, 26
setName
Artist, 24
setSong
Artist, 24
setTitle
Song, 38
setVolume
Player, 32
setYear
Album, 16
Song, 39
Song, 34
_album, 39
_artist, 39
_duration, 39
_file_path, 39
_genre, 39
_id, 40
_metadata_loaded, 40
_title, 40
_year, 40
getAlbum, 35
getArtist, 35
getDuration, 36
getFilePath, 36
getFormattedDuration, 36
getGenre, 36
getId, 36
getTitle, 37
getYear, 37

loadMetadata, 37
setAlbum, 37
setArtist, 38
setDuration, 38
setGenre, 38
setTitle, 38
setYear, 39
Song, 35
toString, 39

toString
Album, 16
Artist, 24
Song, 39

unmute
Player, 33