

Sistema Digital de Música

Gerado por Doxygen 1.9.1

1 Namespaces	1
1.1 Lista de Namespaces	1
2 Índice Hierárquico	3
2.1 Hierarquia de Classes	3
3 Índice dos Componentes	5
3.1 Lista de Classes	5
4 Índice dos Arquivos	7
4.1 Lista de Arquivos	7
5 Namespace	9
5.1 Referência do Namespace core	9
5.1.1 Definições dos tipos	9
5.1.1.1 userid	9
6 Classes	11
6.1 Referência da Classe core::Album	11
6.1.1 Descrição detalhada	13
6.1.2 Construtores e Destrutores	14
6.1.2.1 Album() [1/4]	14
6.1.2.2 Album() [2/4]	14
6.1.2.3 Album() [3/4]	14
6.1.2.4 Album() [4/4]	14
6.1.2.5 ~Album()	15
6.1.3 Funções membros	15
6.1.3.1 addSong()	15
6.1.3.2 calculateTotalDuration()	15
6.1.3.3 findSongById()	16
6.1.3.4 findSongByTitle()	16
6.1.3.5 getArtist()	16
6.1.3.6 getAudioFilePath()	17
6.1.3.7 getFeaturingArtists()	17
6.1.3.8 getFormattedDuration()	17
6.1.3.9 getGenre()	17
6.1.3.10 getName()	18
6.1.3.11 getNextSong()	18
6.1.3.12 getPlayableObjects()	19
6.1.3.13 getPreviousSong()	19
6.1.3.14 getSongAt()	19
6.1.3.15 getSongCount()	20
6.1.3.16 getSongs()	20
6.1.3.17 getUser()	20

6.1.3.18	getYear()	20
6.1.3.19	operator"!=()	20
6.1.3.20	operator==()	21
6.1.3.21	removeSong()	21
6.1.3.22	setArtist()	22
6.1.3.23	setArtistLoader()	22
6.1.3.24	setFeaturingArtists()	22
6.1.3.25	setFeaturingArtistsLoader()	22
6.1.3.26	setFilePath()	23
6.1.3.27	setGenre()	23
6.1.3.28	setSongsLoader() [1/2]	23
6.1.3.29	setSongsLoader() [2/2]	24
6.1.3.30	setUser() [1/2]	24
6.1.3.31	setUser() [2/2]	24
6.1.3.32	setYear()	24
6.1.3.33	switchSong()	24
6.1.3.34	toString()	25
6.1.4	Atributos	25
6.1.4.1	_artist	25
6.1.4.2	_artist_id	25
6.1.4.3	_featuring_artists_ids	25
6.1.4.4	_file_path	26
6.1.4.5	_genre	26
6.1.4.6	_name	26
6.1.4.7	_songs	26
6.1.4.8	_user	26
6.1.4.9	_user_id	26
6.1.4.10	_year	27
6.1.4.11	artistLoader	27
6.1.4.12	featuringArtistsLoader	27
6.1.4.13	songsLoader	27
6.2	Referência da Classe core::Artist	28
6.2.1	Descrição detalhada	30
6.2.2	Construtores e Destrutores	30
6.2.2.1	Artist() [1/3]	31
6.2.2.2	Artist() [2/3]	31
6.2.2.3	Artist() [3/3]	31
6.2.2.4	~Artist()	31
6.2.3	Funções membros	31
6.2.3.1	addAlbum()	31
6.2.3.2	addSong() [1/2]	32
6.2.3.3	addSong() [2/2]	32

6.2.3.4 addSongAlbum()	32
6.2.3.5 calculateTotalDuration()	33
6.2.3.6 findAlbumByName()	33
6.2.3.7 findSongById()	33
6.2.3.8 findSongByTitle()	34
6.2.3.9 getAlbums()	34
6.2.3.10 getAlbumsCount()	34
6.2.3.11 getFormattedDuration()	35
6.2.3.12 getGenre()	35
6.2.3.13 getName()	35
6.2.3.14 getNextSong()	35
6.2.3.15 getNextSongAlbum()	36
6.2.3.16 getPlayableObjects()	36
6.2.3.17 getPreviousSong()	36
6.2.3.18 getPreviousSongAlbum()	37
6.2.3.19 getSongAt()	37
6.2.3.20 getSongAtAlbum()	37
6.2.3.21 getSongs() [1/2]	38
6.2.3.22 getSongs() [2/2]	38
6.2.3.23 getSongsAlbum()	38
6.2.3.24 getSongsCount()	39
6.2.3.25 getTotalDuration()	39
6.2.3.26 getUser()	39
6.2.3.27 hasAlbum()	39
6.2.3.28 hasSong()	40
6.2.3.29 operator!=(())	40
6.2.3.30 operator==(())	40
6.2.3.31 removeAlbum()	41
6.2.3.32 removeSong()	41
6.2.3.33 removeSongAlbum()	41
6.2.3.34 setAlbumsLoader()	42
6.2.3.35 setGenre()	42
6.2.3.36 setName()	42
6.2.3.37 setSongsLoader()	42
6.2.3.38 setUser()	43
6.2.3.39 switchSong()	43
6.2.3.40 toString()	43
6.2.4 Atributos	43
6.2.4.1 _albums	44
6.2.4.2 _genre	44
6.2.4.3 _name	44
6.2.4.4 _songs	44

6.2.4.5 _user	44
6.2.4.6 _user_id	44
6.2.4.7 albumsLoader	45
6.2.4.8 songsLoader	45
6.3 Referência da Classe core::Entity	45
6.3.1 Descrição detalhada	46
6.3.2 Construtores e Destrutores	46
6.3.2.1 Entity() [1/2]	46
6.3.2.2 Entity() [2/2]	46
6.3.2.3 ~Entity()	46
6.3.3 Funções membros	46
6.3.3.1 getDataCriacao()	47
6.3.3.2 getId()	47
6.3.3.3 operator!=(())	47
6.3.3.4 operator<()	47
6.3.3.5 operator==(())	48
6.3.3.6 setDataCriacao()	48
6.3.3.7 setId()	48
6.3.4 Atributos	49
6.3.4.1 _dataCriacao	49
6.3.4.2 _id	49
6.4 Referência da Classe core::HistoryPlayback	49
6.4.1 Descrição detalhada	51
6.4.2 Construtores e Destrutores	51
6.4.2.1 HistoryPlayback() [1/5]	51
6.4.2.2 HistoryPlayback() [2/5]	51
6.4.2.3 HistoryPlayback() [3/5]	51
6.4.2.4 HistoryPlayback() [4/5]	51
6.4.2.5 HistoryPlayback() [5/5]	52
6.4.2.6 ~HistoryPlayback()	52
6.4.3 Funções membros	52
6.4.3.1 getPlayedAt()	52
6.4.3.2 getSong()	52
6.4.3.3 getUser()	52
6.4.3.4 operator!=(())	52
6.4.3.5 operator==(())	53
6.4.3.6 setPlayedAt()	53
6.4.3.7 setSong()	53
6.4.3.8 setUser()	54
6.4.3.9 toString()	54
6.4.4 Atributos	54
6.4.4.1 _played_at	54

6.4.4.2 _song	55
6.4.4.3 _user	55
6.5 Referência da Classe core::Playlist	55
6.5.1 Descrição detalhada	57
6.5.2 Construtores e Destrutores	57
6.5.2.1 Playlist() [1/2]	57
6.5.2.2 Playlist() [2/2]	57
6.5.2.3 ~Playlist()	58
6.5.3 Funções membros	58
6.5.3.1 addSong()	58
6.5.3.2 calculateTotalDuration()	58
6.5.3.3 findSongById()	58
6.5.3.4 findSongByTitle()	59
6.5.3.5 getFormattedDuration()	59
6.5.3.6 getNextSong()	59
6.5.3.7 getPlayableObjects()	60
6.5.3.8 getPreviousSong()	60
6.5.3.9 getSongAt()	60
6.5.3.10 getSongs()	61
6.5.3.11 getTitle()	61
6.5.3.12 getTitulo()	61
6.5.3.13 getUser()	61
6.5.3.14 operator!=(())	61
6.5.3.15 operator==(())	62
6.5.3.16 removeSong()	62
6.5.3.17 setSongsLoader()	63
6.5.3.18 setTitulo()	63
6.5.3.19 setUser()	63
6.5.3.20 switchSong()	63
6.5.4 Atributos	64
6.5.4.1 _loader	64
6.5.4.2 _songs	64
6.5.4.3 _titulo	64
6.5.4.4 _user	64
6.5.4.5 _user_id	64
6.6 Referência da Classe core::Song	65
6.6.1 Descrição detalhada	67
6.6.2 Construtores e Destrutores	67
6.6.2.1 Song() [1/5]	67
6.6.2.2 Song() [2/5]	68
6.6.2.3 Song() [3/5]	68
6.6.2.4 Song() [4/5]	68

6.6.2.5 Song() [5/5]	69
6.6.2.6 ~Song()	69
6.6.3 Funções membros	69
6.6.3.1 getAlbum()	69
6.6.3.2 getArtist()	70
6.6.3.3 getAudioFilePath()	70
6.6.3.4 getDuration()	70
6.6.3.5 getFeaturingArtists()	70
6.6.3.6 getFeaturingArtistsId()	71
6.6.3.7 getFilePath()	71
6.6.3.8 getFormattedDuration()	71
6.6.3.9 getGenre()	71
6.6.3.10 getPlayableObjects()	72
6.6.3.11 getTitle()	72
6.6.3.12 getTrackNumber()	72
6.6.3.13 getUser()	72
6.6.3.14 getYear()	73
6.6.3.15 operator!=(())	73
6.6.3.16 operator==(())	74
6.6.3.17 setAlbum()	74
6.6.3.18 setAlbumLoader()	74
6.6.3.19 setArtist()	75
6.6.3.20 setArtistLoader()	75
6.6.3.21 setDuration()	75
6.6.3.22 setFeaturingArtists() [1/2]	75
6.6.3.23 setFeaturingArtists() [2/2]	76
6.6.3.24 setFeaturingArtistsLoader()	76
6.6.3.25 setGenre()	76
6.6.3.26 setTitle()	76
6.6.3.27 setTrackNumber()	77
6.6.3.28 setUser()	77
6.6.3.29 setYear()	77
6.6.3.30 toString()	78
6.6.4 Atributos	78
6.6.4.1 _album	78
6.6.4.2 _album_id	78
6.6.4.3 _artist	78
6.6.4.4 _artist_id	78
6.6.4.5 _duration	79
6.6.4.6 _featuring_artists_ids	79
6.6.4.7 _file_path	79
6.6.4.8 _genre	79

6.6.4.9 _metadata_loaded	79
6.6.4.10 _title	79
6.6.4.11 _track_number	80
6.6.4.12 _user	80
6.6.4.13 _year	80
6.6.4.14 albumLoader	80
6.6.4.15 artistLoader	80
6.6.4.16 decoderConfig	80
6.6.4.17 featuringArtistsLoader	81
6.6.4.18 user_id	81
6.7 Referência da Classe core::User	81
6.7.1 Descrição detalhada	82
6.7.2 Construtores e Destrutores	83
6.7.2.1 User() [1/4]	83
6.7.2.2 User() [2/4]	83
6.7.2.3 User() [3/4]	83
6.7.2.4 User() [4/4]	83
6.7.2.5 ~User()	83
6.7.3 Funções membros	84
6.7.3.1 getHomePath()	84
6.7.3.2 getInputPath()	84
6.7.3.3 getUID()	84
6.7.3.4 getUsername()	84
6.7.3.5 isCurrentUser()	85
6.7.3.6 operator!=(())	85
6.7.3.7 operator==(())	85
6.7.3.8 setHomePath()	86
6.7.3.9 setInputPath()	86
6.7.3.10 setIsCurrentUser()	86
6.7.3.11 setUID()	86
6.7.3.12 setUsername()	87
6.7.4 Atributos	87
6.7.4.1 _home_path	87
6.7.4.2 _input_path	87
6.7.4.3 _is_current_user	87
6.7.4.4 _uid	88
6.7.4.5 _username	88
7 Arquivos	89
7.1 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp	89
7.1.1 Descrição detalhada	90

7.2 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ Artist.hpp	91
7.2.1 Descrição detalhada	92
7.3 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ EntitiesFWD.hpp	92
7.4 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ Entity.hpp	92
7.4.1 Descrição detalhada	93
7.5 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ HistoryPlayback.hpp	94
7.5.1 Descrição detalhada	94
7.6 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ Playlist.hpp	95
7.6.1 Descrição detalhada	95
7.7 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/↔ Song.hpp	96
7.7.1 Descrição detalhada	97
7.8 Referência do Arquivo /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/User.hpp	97
7.8.1 Descrição detalhada	98
Índice Remissivo	99

Capítulo 1

Namespaces

1.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

core	9
----------------	---

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

core::Entity	45
core::Album	11
core::Artist	28
core::HistoryPlayback	49
core::Playlist	55
core::Song	65
core::User	81
core::ICollection	
core::Artist	28
core::Playlist	55
ICollection	
core::Album	11
core::IPlayable	
core::Artist	28
core::Playlist	55
core::Song	65
IPlayable	
core::Album	11
core::IPlayableObject	
core::Song	65
IPlayableObject	
core::Album	11

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

core::Album	Representa um álbum musical com suas músicas	11
core::Artist	Representa um artista musical com suas músicas e álbuns	28
core::Entity	Interface para entidades do sistema Interface que define características essenciais para entidades do sistema	45
core::HistoryPlayback	Entidade de histórico de reprodução	49
core::Playlist	Representa uma playlist de músicas	55
core::Song	Representa uma música com seus metadados	65
core::User	Classe que representa um usuário do sistema	81

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ Album.hpp	
Definição da classe Album para representar uma coleção de músicas de um artista	89
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ Artist.hpp	
Definição da classe Artist para representar um artista musical	91
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ EntitiesFWD.hpp	92
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ Entity.hpp	
Interface para entidades do sistema	92
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ HistoryPlayback.hpp	
Entidade de histórico de reprodução	94
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ Playlist.hpp	
Definição de uma coleção de músicas escolhidas pelo usuário	95
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ Song.hpp	
Definição da classe Song para representar uma música no sistema	96
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/ User.hpp	
Entidade para usuários do sistema	97

Capítulo 5

Namespace

5.1 Refência do Namespace core

Componentes

- class [Album](#)
Representa um álbum musical com suas músicas.
- class [Artist](#)
Representa um artista musical com suas músicas e álbuns.
- class [Entity](#)
Interface para entidades do sistema Interface que define características essenciais para entidades do sistema.
- class [HistoryPlayback](#)
Entidade de histórico de reprodução.
- class [Playlist](#)
Representa uma playlist de músicas.
- class [Song](#)
Representa uma música com seus metadados.
- class [User](#)
Classe que representa um usuário do sistema.

Definições de Tipos

- using [userid](#) = uid_t

5.1.1 Definições dos tipos

5.1.1.1 userid

```
using core::userid = typedef uid_t
```

Tipo para representar o ID do usuário no OS

Definição na linha 24 do arquivo User.hpp.

Capítulo 6

Classes

6.1 Referência da Classe core::Album

Representa um álbum musical com suas músicas.

```
#include <Album.hpp>
```

Diagrama de hierarquia para core::Album:

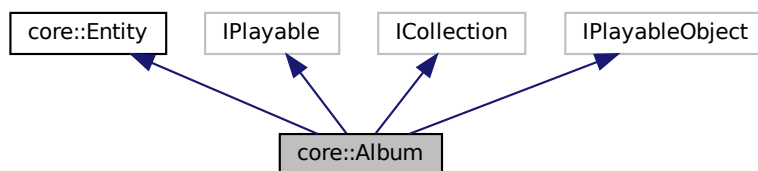
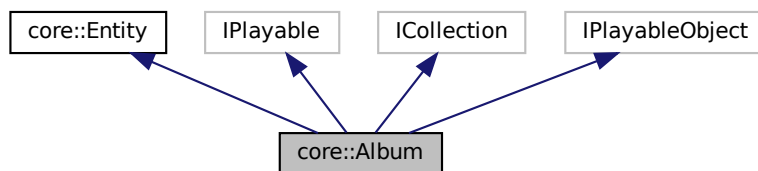


Diagrama de colaboração para core::Album:



Membros Públicos

- [Album](#) ()
Construtor vazio.
- [Album](#) (unsigned id, std::string title, int year, std::string genre, unsigned user_id)
- [Album](#) (const std::string name, std::shared_ptr< [Artist](#) > artist, const std::string genre)
Construtor da classe [Album](#).
- [Album](#) (unsigned id, const std::string name, std::shared_ptr< [Artist](#) > artist, const std::string genre, int year)
Construtor completo da classe [Album](#).
- [~Album](#) ()
Destrutor padrão da classe [Album](#).
- std::string [getName](#) () const
Obtém o nome do álbum.
- std::shared_ptr< [User](#) > [getUser](#) () const
- std::shared_ptr< [Artist](#) > [getArtist](#) () const
Obtém o artista do álbum.
- std::vector< std::shared_ptr< const [Artist](#) > > [getFeaturingArtists](#) () const
Obtém os artistas colaboradores (featuring)
- 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 [setFilePath](#) (std::string &path)
Define o caminho do álbum (se precisar depende da implementação)
- void [setArtist](#) (const std::shared_ptr< [Artist](#) > &artist)
Define o artista do álbum.
- void [setFeaturingArtists](#) (const std::vector< [Artist](#) > &artists)
Define os artistas colaboradores (featuring)
- void [setGenre](#) (const std::string &genre)
Define o gênero do álbum.
- void [setYear](#) (int year)
Define o ano de lançamento.
- void [setUser](#) (const [User](#) &user)
Define o usuário associado ao álbum.
- void [setArtistLoader](#) (const std::function< std::shared_ptr< [Artist](#) >()> &loader)
Define a função para carregar o artista do álbum.
- void [setFeaturingArtistsLoader](#) (const std::function< std::vector< std::shared_ptr< [Artist](#) >>()> &loader)
Define a função para carregar os artistas colaboradores do álbum.
- void [setUser](#) (std::shared_ptr< [User](#) > user)
- std::string [toString](#) () const
Obtém informações do álbum em formato de string.
- bool [operator==](#) (const [Entity](#) &other) const override
Compara dois Albums.
- bool [operator!=](#) (const [Entity](#) &other) const override
Compara dois Albums para desigualdade.
- std::vector< std::shared_ptr< IPlayableObject > > [getPlayableObjects](#) () const override
Obtém os objetos reproduzíveis.
- std::string [getAudioFilePath](#) () const override
Obtém o caminho do arquivo de áudio.

- void `setSongsLoader` (const std::function< std::vector< std::shared_ptr< IPlayable >>()> &loader)
Define a função de carregamento para as músicas do álbum.
- void `addSong` (`Song` &song) override
Adiciona uma música ao álbum.
- bool `switchSong` (unsigned id, unsigned index) override
Move uma música para uma nova posição no álbum.
- bool `removeSong` (unsigned id) override
Remove uma música do álbum pelo ID.
- std::shared_ptr< `Song` > `findSongById` (unsigned songId) override
Busca uma música pelo ID.
- std::shared_ptr< `Song` > `findSongByTitle` (const std::string &title) override
Busca uma música pelo título.
- int `calculateTotalDuration` () override
Calcula a duração total do álbum.
- std::string `getFormattedDuration` () override
Obtém a duração formatada do álbum.
- std::shared_ptr< `Song` > `getNextSong` (`Song` ¤t) override
Obtém a próxima música em relação à música atual.
- std::shared_ptr< `Song` > `getPreviousSong` (`Song` ¤t) override
Obtém a música anterior em relação à música atual.
- std::shared_ptr< `Song` > `getSongAt` (int index) override
Obtém a música em uma posição específica do álbum.
- std::vector< std::shared_ptr< `Song` > > `getSongs` () const override
- void `setSongsLoader` (const std::function< std::vector< std::shared_ptr< `Song` >>()> &loader) override

Atributos Privados

- std::string `_file_path`
- std::string `_name`
- unsigned `_artist_id`
- std::shared_ptr< `User` > `_user`
- unsigned `_user_id`
- std::shared_ptr< `Artist` > `_artist`
- std::vector< unsigned > `_featuring_artists_ids`
- std::vector< std::shared_ptr< `Song` > > `_songs`
- std::string `_genre`
- int `_year`
- std::function< std::vector< std::shared_ptr< `Song` > >> > `songsLoader`
- std::function< std::shared_ptr< `Artist` >> > `artistLoader`
- std::function< std::vector< std::shared_ptr< `Artist` > >> > `featuringArtistsLoader`

Outros membros herdados

6.1.1 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.

Definição na linha 40 do arquivo `Album.hpp`.

6.1.2 Construtores e Destrutores

6.1.2.1 Album() [1/4]

```
core::Album::Album ( )
```

Construtor vazio.

6.1.2.2 Album() [2/4]

```
core::Album::Album (
    unsigned id,
    std::string title,
    int year,
    std::string genre,
    unsigned user_id )
```

6.1.2.3 Album() [3/4]

```
core::Album::Album (
    const std::string name,
    std::shared_ptr< Artist > artist,
    const std::string genre )
```

Construtor da classe [Album](#).

Parâmetros

<i>name</i>	Nome do álbum
<i>artist</i>	Nome do artista do álbum
<i>genre</i>	Gênero do álbum

6.1.2.4 Album() [4/4]

```
core::Album::Album (
    unsigned id,
    const std::string name,
    std::shared_ptr< Artist > artist,
    const std::string genre,
    int year )
```

Construtor completo da classe [Album](#).

Parâmetros

<i>id</i>	Identificador único
<i>name</i>	Nome do álbum
<i>artist</i>	Nome do artista
<i>genre</i>	Gênero musical
<i>year</i>	Ano de lançamento

6.1.2.5 `~Album()`

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

Destrutor padrão da classe `Album`.

Observação

Responsável por liberar a memória dos vetores alocados

6.1.3 Funções membros

6.1.3.1 `addSong()`

```
void core::Album::addSong (
    Song & song ) [override]
```

Adiciona uma música ao álbum.

Parâmetros

<i>song</i>	Ponteiro compartilhado para <code>IPlayable</code> representando a música a ser adicionada
-------------	--

6.1.3.2 `calculateTotalDuration()`

```
int core::Album::calculateTotalDuration ( ) [override]
```

Calcula a duração total do álbum.

Retorna

Duração total em segundos

6.1.3.3 findSongById()

```
std::shared_ptr<Song> core::Album::findSongById (
    unsigned songId ) [override]
```

Busca uma música pelo ID.

Parâmetros

<i>songId</i>	ID da música a ser buscada
---------------	----------------------------

Retorna

Ponteiro compartilhado para IPlayable da música encontrada, ou nullptr se não encontrada

6.1.3.4 findSongByTitle()

```
std::shared_ptr<Song> core::Album::findSongByTitle (
    const std::string & title ) [override]
```

Busca uma música pelo título.

Parâmetros

<i>title</i>	Título da música a ser buscada
--------------	--------------------------------

Retorna

Ponteiro compartilhado para IPlayable da música encontrada, ou nullptr se não encontrada

6.1.3.5 getArtist()

```
std::shared_ptr<Artist> core::Album::getArtist ( ) const
```

Obtém o artista do álbum.

Retorna

Nome do artista/banda

6.1.3.6 `getAudioFilePath()`

```
std::string core::Album::getAudioFilePath ( ) const [override]
```

Obtém o caminho do arquivo de áudio.

Retorna

Caminho do arquivo de áudio

6.1.3.7 `getFeaturingArtists()`

```
std::vector<std::shared_ptr<const Artist> > core::Album::getFeaturingArtists ( ) const
```

Obtém os artistas colaboradores (featuring)

Retorna

Vetor de ponteiros compartilhados para os artistas colaboradores

6.1.3.8 `getFormattedDuration()`

```
std::string core::Album::getFormattedDuration ( ) [override]
```

Obtém a duração formatada do álbum.

Retorna

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

6.1.3.9 `getGenre()`

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

Obtém o gênero do álbum.

Retorna

Gênero musical principal

6.1.3.10 getName()

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

Obtém o nome do álbum.

Retorna

Nome do álbum

6.1.3.11 getNextSong()

```
std::shared_ptr<Song> core::Album::getNextSong (
    Song & current ) [override]
```

Obtém a próxima música em relação à música atual.

Parâmetros

<code>current</code>	Música atual como referência
----------------------	------------------------------

Retorna

Ponteiro compartilhado para `IPlayable` da próxima música, ou `nullptr` se não houver próxima

6.1.3.12 `getPlayableObjects()`

```
std::vector<std::shared_ptr<IPlayableObject> > core::Album::getPlayableObjects ( ) const  
[override]
```

Obtém os objetos reproduzíveis.

Retorna

Vetor contendo esta música como `IPlayableObject`

6.1.3.13 `getPreviousSong()`

```
std::shared_ptr<Song> core::Album::getPreviousSong (   
    Song & current ) [override]
```

Obtém a música anterior em relação à música atual.

Parâmetros

<code>current</code>	Música atual como referência
----------------------	------------------------------

Retorna

Ponteiro compartilhado para `IPlayable` da música anterior, ou `nullptr` se não houver anterior

6.1.3.14 `getSongAt()`

```
std::shared_ptr<Song> core::Album::getSongAt (   
    int index ) [override]
```

Obtém a música em uma posição específica do álbum.

Parâmetros

<i>index</i>	Índice da música no álbum
--------------	---------------------------

Retorna

Ponteiro compartilhado para IPlayable da música na posição especificada, ou nullptr se índice inválido

6.1.3.15 getSongCount()

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

Obtém a quantidade de músicas no álbum.

Retorna

Número total de músicas

6.1.3.16 getSongs()

```
std::vector<std::shared_ptr<Song> > core::Album::getSongs ( ) const [override]
```

6.1.3.17 getUser()

```
std::shared_ptr<User> core::Album::getUser ( ) const
```

6.1.3.18 getYear()

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

Obtém o ano de lançamento.

Retorna

Ano de lançamento do álbum

6.1.3.19 operator"!="()

```
bool core::Album::operator!= (
    const Entity & other ) const [override], [virtual]
```

Compara dois Albums para desigualdade.

Parâmetros

<i>other</i>	<code>Album</code> a ser comparada
--------------	------------------------------------

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementa `core::Entity`.

6.1.3.20 operator==()

```
bool core::Album::operator== (
    const Entity & other ) const [override], [virtual]
```

Compara dois Albums.

Parâmetros

<i>other</i>	<code>Album</code> a ser comparada
--------------	------------------------------------

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementa `core::Entity`.

6.1.3.21 removeSong()

```
bool core::Album::removeSong (
    unsigned id ) [override]
```

Remove uma música do álbum pelo ID.

Parâmetros

<i>id</i>	ID da música a ser removida
-----------	-----------------------------

Retorna

true se a música foi encontrada e removida, false caso contrário

6.1.3.22 setArtist()

```
void core::Album::setArtist (
    const std::shared_ptr< Artist > & artist )
```

Define o artista do álbum.

Parâmetros

<i>artist</i>	Novo nome do artista
---------------	----------------------

6.1.3.23 setArtistLoader()

```
void core::Album::setArtistLoader (
    const std::function< std::shared_ptr< Artist >()> & loader )
```

Define a função para carregar o artista do álbum.

Parâmetros

<i>loader</i>	Função que retorna um ponteiro compartilhado para o artista
---------------	---

6.1.3.24 setFeaturingArtists()

```
void core::Album::setFeaturingArtists (
    const std::vector< Artist > & artists )
```

Define os artistas colaboradores (featuring)

Parâmetros

<i>artists</i>	Vetor de artistas colaboradores
----------------	---------------------------------

6.1.3.25 setFeaturingArtistsLoader()

```
void core::Album::setFeaturingArtistsLoader (
    const std::function< std::vector< std::shared_ptr< Artist >>()> & loader )
```

Define a função para carregar os artistas colaboradores do álbum.

Parâmetros

<code>loader</code>	Função que retorna um vetor de ponteiros compartilhados para os artistas colaboradores
---------------------	--

6.1.3.26 `setFilePath()`

```
void core::Album::setFilePath (
    std::string & path )
```

Define o caminho do álbum (se precisar depende da implementação)

Parâmetros

<code>path</code>	
-------------------	--

6.1.3.27 `setGenre()`

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

Define o gênero do álbum.

Parâmetros

<code>genre</code>	Novo gênero musical
--------------------	---------------------

6.1.3.28 `setSongsLoader()` [1/2]

```
void core::Album::setSongsLoader (
    const std::function< std::vector< std::shared_ptr< IPlayable >>()> & loader )
```

Define a função de carregamento para as músicas do álbum.

Parâmetros

<code>loader</code>	Função que retorna um vetor de ponteiros compartilhados para <code>IPlayable</code>
---------------------	---

6.1.3.29 setSongsLoader() [2/2]

```
void core::Album::setSongsLoader (
    const std::function< std::vector< std::shared_ptr< Song >>()> & loader ) [override]
```

6.1.3.30 setUser() [1/2]

```
void core::Album::setUser (
    const User & user )
```

Define o usuário associado ao álbum.

Parâmetros

<i>user</i>	Ponteiro compartilhado para o usuário
-------------	---------------------------------------

6.1.3.31 setUser() [2/2]

```
void core::Album::setUser (
    std::shared_ptr< User > user )
```

6.1.3.32 setYear()

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

Define o ano de lançamento.

Parâmetros

<i>year</i>	Novo ano de lançamento
-------------	------------------------

6.1.3.33 switchSong()

```
bool core::Album::switchSong (
    unsigned id,
    unsigned index ) [override]
```

Move uma música para uma nova posição no álbum.

Parâmetros

<i>id</i>	ID da música a ser movida
<i>index</i>	Nova posição (índice) da música no álbum

Retorna

true se a operação foi bem-sucedida, false caso contrário

6.1.3.34 toString()

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

Obtém informações do álbum em formato de string.

Retorna

String com nome, artista, ano e quantidade de músicas

6.1.4 Atributos**6.1.4.1 _artist**

```
std::shared_ptr<Artist> core::Album::_artist [mutable], [private]
```

Definição na linha 50 do arquivo Album.hpp.

6.1.4.2 _artist_id

```
unsigned core::Album::_artist_id [private]
```

Definição na linha 47 do arquivo Album.hpp.

6.1.4.3 _featuring_artists_ids

```
std::vector<unsigned> core::Album::_featuring_artists_ids [mutable], [private]
```

Definição na linha 51 do arquivo Album.hpp.

6.1.4.4 `_file_path`

```
std::string core::Album::_file_path [private]
```

Definição na linha 45 do arquivo Album.hpp.

6.1.4.5 `_genre`

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

Definição na linha 53 do arquivo Album.hpp.

6.1.4.6 `_name`

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

Definição na linha 46 do arquivo Album.hpp.

6.1.4.7 `_songs`

```
std::vector<std::shared_ptr<Song> > core::Album::_songs [mutable], [private]
```

Definição na linha 52 do arquivo Album.hpp.

6.1.4.8 `_user`

```
std::shared_ptr<User> core::Album::_user [private]
```

Definição na linha 48 do arquivo Album.hpp.

6.1.4.9 `_user_id`

```
unsigned core::Album::_user_id [private]
```

Definição na linha 49 do arquivo Album.hpp.

6.1.4.10 `_year`

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

Definição na linha 54 do arquivo Album.hpp.

6.1.4.11 `artistLoader`

```
std::function<std::shared_ptr<Artist>>> core::Album::artistLoader [private]
```

Definição na linha 57 do arquivo Album.hpp.

6.1.4.12 `featuringArtistsLoader`

```
std::function<std::vector<std::shared_ptr<Artist> >>> core::Album::featuringArtistsLoader  
[private]
```

Definição na linha 59 do arquivo Album.hpp.

6.1.4.13 `songsLoader`

```
std::function<std::vector<std::shared_ptr<Song> >>> core::Album::songsLoader [private]
```

Definição na linha 56 do arquivo Album.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- </home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp>

6.2 Referência da Classe core::Artist

Representa um artista musical com suas músicas e álbuns.

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

Diagrama de hierarquia para core::Artist:

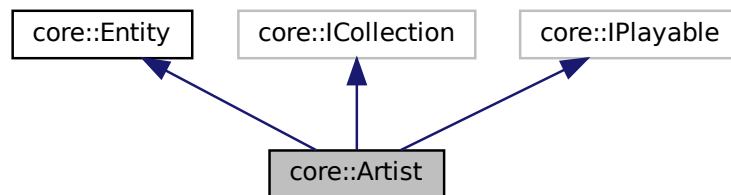
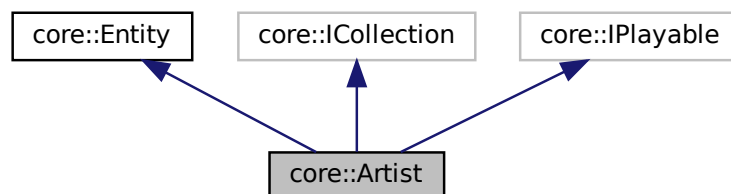


Diagrama de colaboração para core::Artist:



Membros Públicos

- [Artist \(\)](#)
Construtor vazio.
- [Artist \(unsigned id, std::string name, unsigned user_id\)](#)
- [Artist \(const std::string &name, const std::string &genre\)](#)
Construtor da classe [Artist](#).
- [~Artist \(\)](#)
Destrutor da classe [Artist](#).
- [std::string getName \(\) const](#)
Obtém o nome do artista.
- [std::string getGenre \(\) const](#)
Obtém o gênero musical.
- [std::vector< std::shared_ptr< \[Song > > getSongs \\(\\) const\]\(#\)](#)
Obtém todas as músicas do artista.

- `std::vector< std::shared_ptr< 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 setAlbumsLoader (const std::function< std::vector< std::shared_ptr< Album >>()> &loader)`
Define a função para carregar os 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 setUser (const User &user)`
Define o usuário associado ao artista.
- `void addSong (const Song &song)`
Adiciona uma música ao artista.
- `void addAlbum (const Album &album)`
Adiciona um álbum ao artista.
- `bool switchSong (unsigned id, unsigned index) override`
Move uma música para uma nova posição no Artist.
- `std::shared_ptr< Song > findSongByTitle (const std::string &title) override`
Busca uma musica pelo título.
- `std::shared_ptr< Song > findSongById (unsigned songId) override`
Busca uma música de um artista pelo ID.
- `int calculateTotalDuration () override`
Calcula a duração total de um artista.
- `bool removeAlbum (unsigned albumId)`
Remove um álbum do artista.
- `std::shared_ptr< Album > findAlbumByName (const std::string &albumName) const`
Busca uma música pelo nome.
- `int getTotalDuration () const`
Calcula a duração total de todas as músicas.
- `std::string getFormattedDuration () override`
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.
- `std::vector< std::shared_ptr< IPlayableObject > > getPlayableObjects () const override`
Obtém os objetos reproduzíveis.
- `bool operator== (const Entity &other) const override`
Compara dois Artistas.
- `bool operator!= (const Entity &other) const override`
Compara dois Artistas para desigualdade.
- `std::vector< std::shared_ptr< IPlayable > > getSongs ()`
Obtém um vetor com todas as músicas de um Artista.
- `std::vector< std::shared_ptr< IPlayable > > getSongsAlbum (unsigned idAlbum)`
Obtém um vetor com todas as músicas do álbum de um Artista.
- `void addSong (Song &song) override`

- *Adiciona uma música ao artista.*
• void `addSongAlbum` (`Song` &song, unsigned idAlbum)
- *Adiciona uma música a algum album.*
• bool `removeSong` (unsigned id) override
- *Remove uma música do artista pelo ID.*
• bool `removeSongAlbum` (unsigned id, unsigned idAlbum)
- *Remove uma música do álbum pelo ID.*
• std::shared_ptr< `Song` > `getNextSong` (`Song` ¤t) override
- *Obtém a próxima música em relação à música atual.*
• std::shared_ptr< `Song` > `getPreviousSong` (`Song` ¤t) override
- *Obtém a música anterior em relação à música atual.*
• std::shared_ptr< `Song` > `getNextSongAlbum` (`Song` ¤t, unsigned idAlbum)
- *Obtém a próxima música em relação à música atual do album.*
• std::shared_ptr< `Song` > `getPreviousSongAlbum` (`Song` ¤t, unsigned idAlbum)
- *Obtém a música anterior em relação à música atual do album.*
• std::shared_ptr< `Song` > `getSongAt` (int index) override
- *Obtém a música em uma posição específica de songs.*
• std::shared_ptr< IPlayable > `getSongAtAlbum` (int index, unsigned idAlbum)
- *Obtém a música em uma posição específica de Album.*
• std::shared_ptr< `User` > `getUser` () const
- void `setSongsLoader` (const std::function< std::vector< std::shared_ptr< `Song` >>()> &loader) override

Atributos Privados

- std::string `_name`
- std::string `_genre`
- std::vector< std::shared_ptr< `Song` > > `_songs`
- std::vector< std::shared_ptr< `Album` > > `_albums`
- std::shared_ptr< `User` > `_user`
- unsigned `_user_id`
- std::function< std::vector< std::shared_ptr< `Song` > >> `songsLoader`
- std::function< std::vector< std::shared_ptr< `Album` > >> `albumsLoader`

Outros membros herdados

6.2.1 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.

Definição na linha 37 do arquivo Artist.hpp.

6.2.2 Construtores e Destrutores

6.2.2.1 `Artist()` [1/3]

```
core::Artist::Artist ( )
```

Construtor vazio.

6.2.2.2 `Artist()` [2/3]

```
core::Artist::Artist (
    unsigned id,
    std::string name,
    unsigned user_id )
```

6.2.2.3 `Artist()` [3/3]

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

Construtor da classe [Artist](#).

Parâmetros

<i>name</i>	Nome do artista
<i>genre</i>	Gênero musical do artista

6.2.2.4 `~Artist()`

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

Destrutor da classe [Artist](#).

Observação

Responsável por liberar a memória dos vetores alocados

6.2.3 Funções membros

6.2.3.1 `addAlbum()`

```
void core::Artist::addAlbum (
    const Album & album )
```

Adiciona um álbum ao artista.

Parâmetros

<i>album</i>	Álbum a ser adicionado
--------------	------------------------

6.2.3.2 addSong() [1/2]

```
void core::Artist::addSong (
    const Song & song )
```

Adiciona uma música ao artista.

Parâmetros

<i>song</i>	Música a ser adicionada
-------------	-------------------------

6.2.3.3 addSong() [2/2]

```
void core::Artist::addSong (
    Song & song ) [override]
```

Adiciona uma música ao artista.

Parâmetros

<i>song</i>	Ponteiro compartilhado para IPlayable representando a música a ser adicionada
-------------	---

6.2.3.4 addSongAlbum()

```
void core::Artist::addSongAlbum (
    Song & song,
    unsigned idAlbum )
```

Adiciona uma música a algum album.

Parâmetros

<i>song</i>	Ponteiro compartilhado para IPlayable representando a
<i>idAlbum</i>	unsigned com id do album música a ser adicionada

6.2.3.5 `calculateTotalDuration()`

```
int core::Artist::calculateTotalDuration ( ) [override]
```

Calcula a duração total de um artista.

Retorna

Duração total em segundos

6.2.3.6 `findAlbumByName()`

```
std::shared_ptr<Album> core::Artist::findAlbumByName (
    const std::string & albumName ) const
```

Busca uma música pelo nome.

Parâmetros

<i>songName</i>	Nome da música a buscar
-----------------	-------------------------

Retorna

Ponteiro para a música encontrada ou nullptr se não encontrada

Busca um álbum pelo nome

Parâmetros

<i>albumName</i>	Nome do álbum a buscar
------------------	------------------------

Retorna

Ponteiro para o álbum encontrado ou nullptr se não encontrado

6.2.3.7 `findSongById()`

```
std::shared_ptr<Song> core::Artist::findSongById (
    unsigned songId ) [override]
```

Busca uma música de um artista pelo ID.

Parâmetros

<i>song</i> ↔ <i>Id</i>	ID da música a ser buscada
----------------------------	----------------------------

Retorna

Ponteiro compartilhado para IPlayable da música encontrada, ou nullptr se não encontrada

6.2.3.8 findSongByTitle()

```
std::shared_ptr<Song> core::Artist::findSongByTitle (
    const std::string & title ) [override]
```

Busca uma musica pelo título.

Parâmetros

<i>title</i>	Título da musica de um artista a ser buscado
--------------	--

Retorna

Ponteiro compartilhado para IPlayable da música encontrada, ou nullptr se não encontrada

6.2.3.9 getAlbums()

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

Obtém todos os álbuns do artista.

Retorna

Vector com todos os álbuns do artista

6.2.3.10 getAlbumsCount()

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

Obtém a quantidade de álbuns do artista.

Retorna

Número total de álbuns

6.2.3.11 `getFormattedDuration()`

```
std::string core::Artist::getFormattedDuration ( ) [override]
```

Obtém a duração total formatada.

Retorna

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

6.2.3.12 `getGenre()`

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

Obtém o gênero musical.

Retorna

Gênero musical principal do artista

6.2.3.13 `getName()`

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

Obtém o nome do artista.

Retorna

Nome do artista/banda

6.2.3.14 `getNextSong()`

```
std::shared_ptr<Song> core::Artist::getNextSong (
    Song & current ) [override]
```

Obtém a próxima música em relação à música atual.

Parâmetros

<i>current</i>	Música atual como referência
----------------	------------------------------

Retorna

Ponteiro compartilhado para IPlayable da próxima música, ou nullptr se não houver próxima

6.2.3.15 getNextSongAlbum()

```
std::shared_ptr<Song> core::Artist::getNextSongAlbum (
    Song & current,
    unsigned idAlbum )
```

Obtém a próxima música em relação à música atual do album.

Parâmetros

<i>current</i>	Música atual como referência
<i>idAlbum</i>	unsigned id do album

Retorna

Ponteiro compartilhado para IPlayable da próxima música, ou nullptr se não houver próxima

6.2.3.16 getPlayableObjects()

```
std::vector<std::shared_ptr<IPlayableObject> > core::Artist::getPlayableObjects ( ) const
[override]
```

Obtém os objetos reproduzíveis.

Retorna

Vetor contendo esta música como IPlayableObject

6.2.3.17 getPreviousSong()

```
std::shared_ptr<Song> core::Artist::getPreviousSong (
    Song & current ) [override]
```

Obtém a música anterior em relação à música atual.

Parâmetros

<i>current</i>	Música atual como referência
----------------	------------------------------

Retorna

Ponteiro compartilhado para `IPlayable` da próxima música, ou `nullptr` se não houver próxima

6.2.3.18 `getPreviousSongAlbum()`

```
std::shared_ptr<Song> core::Artist::getPreviousSongAlbum (
    Song & current,
    unsigned idAlbum )
```

Obtém a música anterior em relação à música atual do album.

Parâmetros

<i>current</i>	Música atual como referência
<i>idAlbum</i>	unsigned id do album

Retorna

Ponteiro compartilhado para `IPlayable` da próxima música, ou `nullptr` se não houver próxima

6.2.3.19 `getSongAt()`

```
std::shared_ptr<Song> core::Artist::getSongAt (
    int index ) [override]
```

Obtém a música em uma posição específica de songs.

Parâmetros

<i>index</i>	Índice da música em songs
--------------	---------------------------

Retorna

Ponteiro compartilhado para `IPlayable` da música na posição especificada, ou `nullptr` se índice inválido

6.2.3.20 `getSongAtAlbum()`

```
std::shared_ptr<IPlayable> core::Artist::getSongAtAlbum (
    int index,
    unsigned idAlbum )
```

Obtém a música em uma posição específica de [Album](#).

Parâmetros

<i>index</i>	Índice da música em songs
<i>idAlbum</i>	Id do album

Retorna

Ponteiro compartilhado para IPlayable da música na posição especificada, ou nullptr se índice inválido

6.2.3.21 `getSongs()` [1/2]

```
std::vector<std::shared_ptr<IPlayable> > core::Artist::getSongs ( )
```

Obtém um vetor com todas as músicas de um Artista.

Retorna

Vetor de ponteiros compartilhados para IPlayable contendo todas as músicas

6.2.3.22 `getSongs()` [2/2]

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

Obtém todas as músicas do artista.

Retorna

Vector com todas as músicas do artista

6.2.3.23 `getSongsAlbum()`

```
std::vector<std::shared_ptr<IPlayable> > core::Artist::getSongsAlbum (
    unsigned idAlbum )
```

Obtém um vetor com todas as músicas do álbum de um Artista.

Parâmetros

<i>id</i>	Album
-----------	-----------------------

Retorna

Vetor de ponteiros compartilhados para `IPlayable` contendo todas as músicas

6.2.3.24 `getSongsCount()`

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

Obtém a quantidade de músicas do artista.

Retorna

Número total de músicas

6.2.3.25 `getTotalDuration()`

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

Calcula a duração total de todas as músicas.

Retorna

Duração total em segundos

6.2.3.26 `getUser()`

```
std::shared_ptr<User> core::Artist::getUser ( ) const
```

6.2.3.27 `hasAlbum()`

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

Verifica se o artista tem álbuns.

Retorna

true se possui pelo menos um álbum, false caso contrário

6.2.3.28 hasSong()

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

Verifica se o artista tem músicas.

Retorna

true se possui pelo menos uma música, false caso contrário

6.2.3.29 operator!=()

```
bool core::Artist::operator!= (
    const Entity & other ) const [override], [virtual]
```

Compara dois Artistas para desigualdade.

Parâmetros

<i>other</i>	Artista a ser comparada
--------------	-------------------------

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementa [core::Entity](#).

6.2.3.30 operator==()

```
bool core::Artist::operator== (
    const Entity & other ) const [override], [virtual]
```

Compara dois Artistas.

Parâmetros

<i>other</i>	Artista a ser comparada
--------------	-------------------------

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementa [core::Entity](#).

6.2.3.31 `removeAlbum()`

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

Remove um álbum do artista.

Parâmetros

<i>albumId</i>	ID do álbum a ser removido
----------------	----------------------------

Retorna

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

6.2.3.32 `removeSong()`

```
bool core::Artist::removeSong (
    unsigned id ) [override]
```

Remove uma música do artista pelo ID.

Parâmetros

<i>id</i>	ID da música a ser removida
-----------	-----------------------------

Retorna

true se a música foi encontrada e removida, false caso contrário

6.2.3.33 `removeSongAlbum()`

```
bool core::Artist::removeSongAlbum (
    unsigned id,
    unsigned idAlbum )
```

Remove uma música do álbum pelo ID.

Parâmetros

<i>id</i>	ID da música a ser removida
<i>id</i>	ID do álbum alvo

Retorna

true se a música foi encontrada e removida, false caso contrário

6.2.3.34 setAlbumsLoader()

```
void core::Artist::setAlbumsLoader (
    const std::function< std::vector< std::shared_ptr< Album >>()> & loader )
```

Define a função para carregar os álbuns do artista.

Parâmetros

<i>loader</i>	Função que retorna um vetor de álbuns
---------------	---------------------------------------

6.2.3.35 setGenre()

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

Define o gênero musical.

Parâmetros

<i>genre</i>	Novo gênero musical
--------------	---------------------

6.2.3.36 setName()

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

Define o nome do artista.

Parâmetros

<i>name</i>	Novo nome do artista
-------------	----------------------

6.2.3.37 setSongsLoader()

```
void core::Artist::setSongsLoader (
```

```
const std::function< std::vector< std::shared_ptr< Song >>()> & loader ) [override]
```

6.2.3.38 setUser()

```
void core::Artist::setUser (
    const User & user )
```

Define o usuário associado ao artista.

Parâmetros

<i>user</i>	Ponteiro compartilhado para o usuário
-------------	---------------------------------------

6.2.3.39 switchSong()

```
bool core::Artist::switchSong (
    unsigned id,
    unsigned index ) [override]
```

Move uma música para uma nova posição no [Artist](#).

Parâmetros

<i>id</i>	ID da música a ser movida
<i>index</i>	Nova posição (índice) da música no álbum

Retorna

true se a operação foi bem-sucedida, false caso contrário

6.2.3.40 toString()

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

Converte o artista para string formatada.

Retorna

String com nome, gênero e estatísticas do artista

6.2.4 Atributos

6.2.4.1 `_albums`

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

Definição na linha 45 do arquivo Artist.hpp.

6.2.4.2 `_genre`

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

Definição na linha 43 do arquivo Artist.hpp.

6.2.4.3 `_name`

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

Definição na linha 42 do arquivo Artist.hpp.

6.2.4.4 `_songs`

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

Definição na linha 44 do arquivo Artist.hpp.

6.2.4.5 `_user`

```
std::shared_ptr<User> core::Artist::_user [private]
```

Definição na linha 46 do arquivo Artist.hpp.

6.2.4.6 `_user_id`

```
unsigned core::Artist::_user_id [private]
```

Definição na linha 47 do arquivo Artist.hpp.

6.2.4.7 albumsLoader

```
std::function<std::vector<std::shared_ptr<Album> >>> core::Artist::albumsLoader [private]
```

Definição na linha 50 do arquivo Artist.hpp.

6.2.4.8 songsLoader

```
std::function<std::vector<std::shared_ptr<Song> >>> core::Artist::songsLoader [private]
```

Definição na linha 49 do arquivo Artist.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

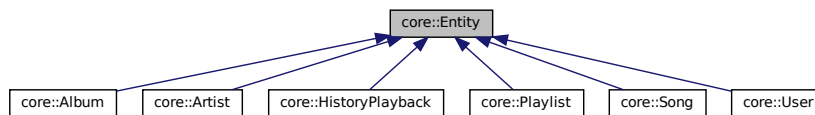
- /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/[Artist.hpp](#)

6.3 Referência da Classe core::Entity

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

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

Diagrama de hierarquia para core::Entity:



Membros Públicos

- [Entity](#) ()
- [Entity](#) (unsigned id)
- virtual [~Entity](#) ()=default
- unsigned [getId](#) () const
Obtém o ID da entidade.
- void [setId](#) (unsigned id)
Define o ID da entidade.
- const Datetime [getDataCriacao](#) () const
Obtém a data de criação.
- void [setDataCriacao](#) (Datetime date)
Define a data de criação da entidade.
- virtual bool [operator==](#) (const [Entity](#) &other) const
Compara duas entidades.
- virtual bool [operator!=](#) (const [Entity](#) &other) const
Compara duas entidades para desigualdade.
- virtual bool [operator<](#) (const [Entity](#) &other) const
Compara qual entidade é menor.

Atributos Protegidos

- unsigned `_id` = 0
- Datetime `_dataCriacao`

6.3.1 Descrição detalhada

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

Definição na linha 21 do arquivo Entity.hpp.

6.3.2 Construtores e Destrutores

6.3.2.1 Entity() [1/2]

```
core::Entity::Entity ( ) [inline]
```

Data de criação

Definição na linha 27 do arquivo Entity.hpp.

```
27 : _id(0), _dataCriacao(Datetime()) {};
```

6.3.2.2 Entity() [2/2]

```
core::Entity::Entity (
    unsigned id ) [inline]
```

Definição na linha 28 do arquivo Entity.hpp.

```
28 : _id(id), _dataCriacao(Datetime()) {}
```

6.3.2.3 ~Entity()

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

6.3.3 Funções membros

6.3.3.1 `getDataCriacao()`

```
const Datetime core::Entity::getDataCriacao ( ) const
```

Obtém a data de criação.

Retorna

a data de criação da entidade

6.3.3.2 `getId()`

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

Obtém o ID da entidade.

Retorna

ID da entidade

6.3.3.3 `operator!=(())`

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

Compara duas entidades para desigualdade.

Parâmetros

<i>other</i>	Entidade a ser comparada
--------------	--------------------------

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementado por `core::User`, `core::Song`, `core::Playlist`, `core::HistoryPlayback`, `core::Artist` e `core::Album`.

6.3.3.4 `operator<()`

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

Compara qual entidade é menor.

Parâmetros

<i>other</i>	Entidade a ser comparada
--------------	--------------------------

Retorna

true se a entidade atual for menor que a outra, false caso contrário

6.3.3.5 operator==()

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

Compara duas entidades.

Parâmetros

<i>other</i>	Entidade a ser comparada
--------------	--------------------------

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementado por [core::User](#), [core::Song](#), [core::Playlist](#), [core::HistoryPlayback](#), [core::Artist](#) e [core::Album](#).

6.3.3.6 setDataCriacao()

```
void core::Entity::setDataCriacao (
    Datetime date )
```

Define a data de criação da entidade.

Parâmetros

<i>date</i>	Data de criação
-------------	-----------------

6.3.3.7 setId()

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

Define o ID da entidade.

Parâmetros

<i>id</i>	Novo ID da entidade
-----------	---------------------

6.3.4 Atributos

6.3.4.1 _dataCriacao

```
Datetime core::Entity::_dataCriacao [protected]
```

Definição na linha 24 do arquivo Entity.hpp.

6.3.4.2 _id

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

ID da entidade

Definição na linha 23 do arquivo Entity.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/[Entity.hpp](#)

6.4 Referência da Classe core::HistoryPlayback

Entidade de histórico de reprodução.

```
#include <HistoryPlayback.hpp>
```

Diagrama de hierarquia para core::HistoryPlayback:

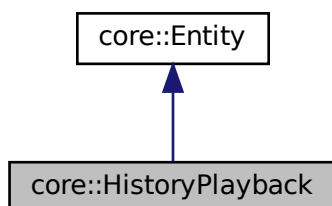
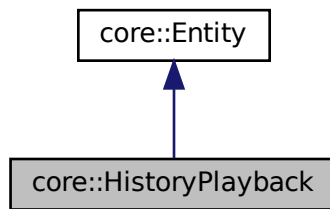


Diagrama de colaboração para core::HistoryPlayback:



Membros Públicos

- `HistoryPlayback ()`
- `HistoryPlayback (User &user, Song &song)`
- `HistoryPlayback (unsigned id, User &user, Song &song)`
- `HistoryPlayback (unsigned id, User &user, Song &song, std::time_t played_at)`
- `HistoryPlayback (User &user, Song &song, std::time_t played_at)`
- `~HistoryPlayback ()` override=default
- `std::shared_ptr< const User > getUser ()` const
Obtém o usuário.
- `void setUser (const User &user)`
Define o usuário.
- `std::shared_ptr< const Song > getSong ()` const
Obtém a música associada ao histórico de reprodução.
- `void setSong (const Song &song)`
Define a música associada ao histórico de reprodução.
- `std::time_t getPlayedAt ()` const
Obtém o timestamp da reprodução.
- `void setPlayedAt (std::time_t played_at)`
Define o timestamp da reprodução.
- `std::string toString ()` const
Obtém uma representação em string do histórico de reprodução.
- `bool operator== (const Entity &other)` const override
Compara duas HistoryPlayback.
- `bool operator!= (const Entity &other)` const override
Compara duas HistoryPlayback para desigualdade.

Atributos Privados

- `std::shared_ptr< User > _user`
- `std::shared_ptr< Song > _song`
- `std::time_t _played_at`

Outros membros herdados

6.4.1 Descrição detalhada

Entidade de histórico de reprodução.

Representa um registro de reprodução de uma música, incluindo informações como o usuário, a música e timestamp da reprodução.

Definição na linha 30 do arquivo `HistoryPlayback.hpp`.

6.4.2 Construtores e Destrutores

6.4.2.1 `HistoryPlayback()` [1/5]

```
core::HistoryPlayback::HistoryPlayback ( )
```

6.4.2.2 `HistoryPlayback()` [2/5]

```
core::HistoryPlayback::HistoryPlayback (
    User & user,
    Song & song )
```

6.4.2.3 `HistoryPlayback()` [3/5]

```
core::HistoryPlayback::HistoryPlayback (
    unsigned id,
    User & user,
    Song & song )
```

6.4.2.4 `HistoryPlayback()` [4/5]

```
core::HistoryPlayback::HistoryPlayback (
    unsigned id,
    User & user,
    Song & song,
    std::time_t played_at )
```

6.4.2.5 HistoryPlayback() [5/5]

```
core::HistoryPlayback::HistoryPlayback (
    User & user,
    Song & song,
    std::time_t played_at )
```

6.4.2.6 ~HistoryPlayback()

```
core::HistoryPlayback::~~HistoryPlayback ( ) [override], [default]
```

6.4.3 Funções membros

6.4.3.1 getPlayedAt()

```
std::time_t core::HistoryPlayback::getPlayedAt ( ) const
```

Obtém o timestamp da reprodução.

Retorna

Timestamp da reprodução

6.4.3.2 getSong()

```
std::shared_ptr<const Song> core::HistoryPlayback::getSong ( ) const
```

Obtém a música associada ao histórico de reprodução.

Retorna

Ponteiro compartilhado para a música

6.4.3.3 getUser()

```
std::shared_ptr<const User> core::HistoryPlayback::getUser ( ) const
```

Obtém o usuário.

Retorna

Ponteiro compartilhado para o usuário

6.4.3.4 operator!=()

```
bool core::HistoryPlayback::operator!= (
    const Entity & other ) const [override], [virtual]
```

Compara duas [HistoryPlayback](#) para desigualdade.

Parâmetros

<i>other</i>	HistoryPlayback a ser comparada
--------------	---

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementa [core::Entity](#).

6.4.3.5 operator==()

```
bool core::HistoryPlayback::operator== (
    const Entity & other ) const [override], [virtual]
```

Compara duas [HistoryPlayback](#).

Parâmetros

<i>other</i>	HistoryPlayback a ser comparada
--------------	---

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementa [core::Entity](#).

6.4.3.6 setPlayedAt()

```
void core::HistoryPlayback::setPlayedAt (
    std::time_t played_at )
```

Define o timestamp da reprodução.

Parâmetros

<i>played_at</i>	Timestamp da reprodução
------------------	-------------------------

6.4.3.7 setSong()

```
void core::HistoryPlayback::setSong (
```

```
const Song & song )
```

Define a música associada ao histórico de reprodução.

Parâmetros

<i>song</i>	Referência para a música
-------------	--------------------------

6.4.3.8 setUser()

```
void core::HistoryPlayback::setUser (
    const User & user )
```

Define o usuário.

Parâmetros

<i>user</i>	Referência para o usuário
-------------	---------------------------

6.4.3.9 toString()

```
std::string core::HistoryPlayback::toString ( ) const
```

Obtém uma representação em string do histórico de reprodução.

Retorna

String representando o histórico de reprodução

6.4.4 Atributos

6.4.4.1 _played_at

```
std::time_t core::HistoryPlayback::_played_at [private]
```

Definição na linha 34 do arquivo HistoryPlayback.hpp.

6.4.4.2 _song

```
std::shared_ptr<Song> core::HistoryPlayback::_song [private]
```

Definição na linha 33 do arquivo HistoryPlayback.hpp.

6.4.4.3 _user

```
std::shared_ptr<User> core::HistoryPlayback::_user [private]
```

Definição na linha 32 do arquivo HistoryPlayback.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/[HistoryPlayback.hpp](#)

6.5 Referência da Classe core::Playlist

Representa uma playlist de músicas.

```
#include <Playlist.hpp>
```

Diagrama de hierarquia para core::Playlist:

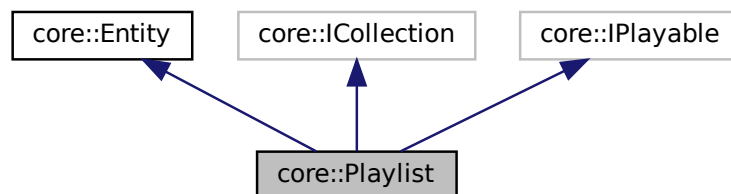
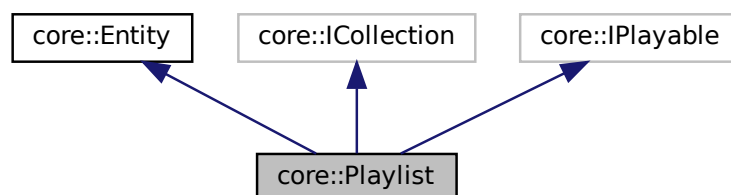


Diagrama de colaboração para core::Playlist:



Membros Públicos

- [Playlist](#) ()
Construtor padrão da classe [Playlist](#).
- [Playlist](#) (const unsigned id, const std::string titulo)
Construtor de classe [Playlist](#).
- [~Playlist](#) ()
Destrutor padrão da classe [Playlist](#).
- std::string [getTitulo](#) ()
Getter de título.
- std::string [getTitle](#) () const
- void [setTitulo](#) (const std::string nome)
Setter de título.
- virtual bool [operator==](#) (const [Entity](#) &other) const override
Compara duas [Playlist](#).
- virtual bool [operator!=](#) (const [Entity](#) &other) const override
Compara duas [Playlists](#) para desigualdade.
- virtual std::vector< std::shared_ptr< [Song](#) > > [getSongs](#) () const override
Getter do vector de músicas de [ICollection](#).
- virtual void [setSongsLoader](#) (const std::function< std::vector< std::shared_ptr< [Song](#) >>()> &loader) override
Setter da função de loader.
- virtual void [addSong](#) ([Song](#) &song) override
Adiciona música à coleção.
- virtual bool [switchSong](#) (unsigned id, unsigned index) override
Troca uma música de posição na playlist.
- virtual bool [removeSong](#) (unsigned id) override
- virtual std::shared_ptr< [Song](#) > [findSongById](#) (unsigned songId) override
Acha uma música pelo ID da música.
- virtual std::shared_ptr< [Song](#) > [findSongByTitle](#) (const std::string &title) override
Acha uma música pelo título da música.
- virtual int [calculateTotalDuration](#) () override
Calcula duração total da música.
- virtual std::string [getFormattedDuration](#) () override
Calcula a duração e formata a string.
- virtual std::shared_ptr< [Song](#) > [getNextSong](#) ([Song](#) ¤t) override
Pega a próxima música na ordem da playlist.
- virtual std::shared_ptr< [Song](#) > [getPreviousSong](#) ([Song](#) ¤t) override
Pega a música anterior.
- virtual std::shared_ptr< [Song](#) > [getSongAt](#) (int index) override
Pega a música na posição.
- virtual std::vector< std::shared_ptr< [IPlayableObject](#) > > [getPlayableObjects](#) () const override
Obtém os objetos reproduzíveis.
- std::shared_ptr< [User](#) > [getUser](#) () const
Getter de user da interface [IPlayable](#).
- void [setUser](#) (const [User](#) &user)
Setter para o user da interface [IPlayable](#).

Atributos Privados

- `std::string _titulo`
- `unsigned _user_id`
- `std::shared_ptr< User > _user`
- `std::vector< std::shared_ptr< Song > > _songs`
- `std::function< std::vector< std::shared_ptr< Song > >> _loader`

Outros membros herdados

6.5.1 Descrição detalhada

Representa uma playlist de músicas.

Representa a entidade `Playlist`.

Definição na linha 37 do arquivo `Playlist.hpp`.

6.5.2 Construtores e Destrutores

6.5.2.1 `Playlist()` [1/2]

```
core::Playlist::Playlist ( )
```

Construtor padrão da classe `Playlist`.

6.5.2.2 `Playlist()` [2/2]

```
core::Playlist::Playlist (
    const unsigned id,
    const std::string titulo )
```

Construtor de classe `Playlist`.

Parâmetros

<i>id</i>	Identificador único
<i>titulo</i>	Título da <code>Playlist</code>

6.5.2.3 ~Playlist()

```
core::Playlist::~~Playlist ( )
```

Destrutor padrão da classe [Playlist](#).

Observação

Libera a memória dos vetores alocados

6.5.3 Funções membros

6.5.3.1 addSong()

```
virtual void core::Playlist::addSong (
    Song & song ) [override], [virtual]
```

Adiciona música à coleção.

Parâmetros

<i>song</i>	música a ser adicionada
-------------	-------------------------

6.5.3.2 calculateTotalDuration()

```
virtual int core::Playlist::calculateTotalDuration ( ) [override], [virtual]
```

Calcula duração total da música.

Retorna

duração da música em segundos

6.5.3.3 findSongById()

```
virtual std::shared_ptr<Song> core::Playlist::findSongById (
    unsigned songId ) [override], [virtual]
```

Acha uma música pelo ID da música.

Parâmetros

<i>song</i> ↔ <i>Id</i>	id da música
----------------------------	--------------

Retorna

`std::shared_ptr<Song>` para a instância da música
`nullptr` caso a música não exista

6.5.3.4 findSongByTitle()

```
virtual std::shared_ptr<Song> core::Playlist::findSongByTitle (
    const std::string & title ) [override], [virtual]
```

Acha uma música pelo título da música.

Parâmetros

<i>title</i>	título da música
--------------	------------------

Retorna

`std::shared_ptr<Song>` para a instância da música
`nullptr` caso a música não exista

6.5.3.5 getFormattedDuration()

```
virtual std::string core::Playlist::getFormattedDuration ( ) [override], [virtual]
```

Calcula a duração e formata a string.

Retorna

`std::string` no formato hh:mm

6.5.3.6 getNextSong()

```
virtual std::shared_ptr<Song> core::Playlist::getNextSong (
    Song & current ) [override], [virtual]
```

Pega a próxima música na ordem da playlist.

Parâmetros

<i>current</i>	música atual
----------------	--------------

Retorna

`std::shared_ptr<Song>` para a próxima

6.5.3.7 getPlayableObjects()

```
virtual std::vector<std::shared_ptr<IPlayableObject> > core::Playlist::getPlayableObjects ( )  
const [override], [virtual]
```

Obtém os objetos reproduzíveis.

Retorna

Vetor de ponteiros compartilhados para objetos reproduzíveis

6.5.3.8 getPreviousSong()

```
virtual std::shared_ptr<Song> core::Playlist::getPreviousSong (  
    Song & current ) [override], [virtual]
```

Pega a música anterior.

Parâmetros

<i>current</i>	música atual
----------------	--------------

Retorna

`std::shared_ptr<Song>` para a anterior

6.5.3.9 getSongAt()

```
virtual std::shared_ptr<Song> core::Playlist::getSongAt (  
    int index ) [override], [virtual]
```

Pega a música na posição.

Parâmetros

<i>index</i>	posição da música
--------------	-------------------

Retorna

std::shared_ptr<Song> para a música no index indicado
nullptr caso o index esteja fora do escopo

6.5.3.10 getSongs()

```
virtual std::vector<std::shared_ptr<Song> > core::Playlist::getSongs ( ) const [override],  
[virtual]
```

Getter do vector de músicas de ICollection.

6.5.3.11 getTitle()

```
std::string core::Playlist::getTitle ( ) const
```

6.5.3.12 getTitulo()

```
std::string core::Playlist::getTitulo ( )
```

Getter de título.

6.5.3.13 getUser()

```
std::shared_ptr<User> core::Playlist::getUser ( ) const
```

Getter de user da interface IPlayable.

Retorna

std::shared_ptr<User> para o usuário

6.5.3.14 operator!=(())

```
virtual bool core::Playlist::operator!= (   
    const Entity & other ) const [override], [virtual]
```

Compara duas Playlists para desigualdade.

Parâmetros

<i>other</i>	Playlist a ser comparada
--------------	--

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementa [core::Entity](#).

6.5.3.15 operator==()

```
virtual bool core::Playlist::operator== (  
    const Entity & other ) const [override], [virtual]
```

Compara duas [Playlist](#).

Parâmetros

<i>other</i>	Playlist a ser comparada
--------------	--

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementa [core::Entity](#).

6.5.3.16 removeSong()

```
virtual bool core::Playlist::removeSong (  
    unsigned id ) [override], [virtual]
```

Parâmetros

<i>id</i>	
-----------	--

Retorna

true

false

6.5.3.17 setSongsLoader()

```
virtual void core::Playlist::setSongsLoader (
    const std::function< std::vector< std::shared_ptr< Song >>()> & loader ) [override],
[virtual]
```

Setter da função de loader.

Parâmetros

<i>loader</i>	função para fazer load de música
---------------	----------------------------------

6.5.3.18 setTitulo()

```
void core::Playlist::setTitulo (
    const std::string nome )
```

Setter de título.

6.5.3.19 setUser()

```
void core::Playlist::setUser (
    const User & user )
```

Setter para o user da interface IPlayable.

Parâmetros

<i>user</i>	usuário
-------------	---------

6.5.3.20 switchSong()

```
virtual bool core::Playlist::switchSong (
    unsigned id,
    unsigned index ) [override], [virtual]
```

Troca uma música de posição na playlist.

Parâmetros

<i>id</i>	da música a ser trocada de posição
<i>index</i>	para o qual a música será realocada

Retorna

true caso a troca seja bem sucedida
false caso a troca não seja bem sucedida

6.5.4 Atributos

6.5.4.1 _loader

```
std::function<std::vector<std::shared_ptr<Song> >>> core::Playlist::_loader [private]
```

Definição na linha 43 do arquivo Playlist.hpp.

6.5.4.2 _songs

```
std::vector<std::shared_ptr<Song> > core::Playlist::_songs [mutable], [private]
```

Definição na linha 42 do arquivo Playlist.hpp.

6.5.4.3 _titulo

```
std::string core::Playlist::_titulo [private]
```

Definição na linha 39 do arquivo Playlist.hpp.

6.5.4.4 _user

```
std::shared_ptr<User> core::Playlist::_user [private]
```

Definição na linha 41 do arquivo Playlist.hpp.

6.5.4.5 _user_id

```
unsigned core::Playlist::_user_id [private]
```

Definição na linha 40 do arquivo Playlist.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/[Playlist.hpp](#)

6.6 Referência da Classe core::Song

Representa uma música com seus metadados.

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

Diagrama de hierarquia para core::Song:

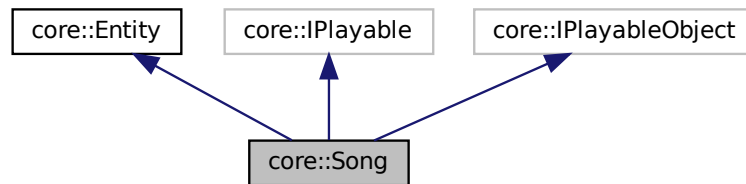
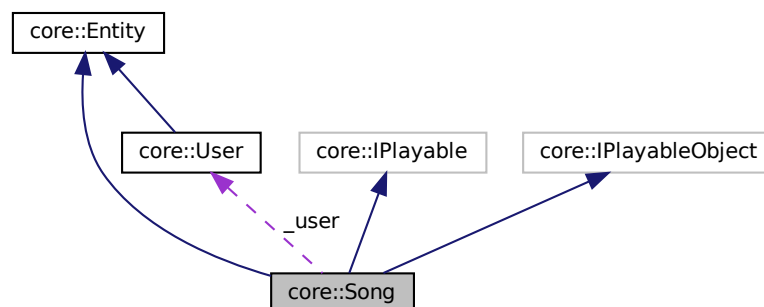


Diagrama de colaboração para core::Song:



Membros Públicos

- `Song()`
Construtor padrão - inicializa uma música com valores padrão.
- `Song(const std::string &title, std::shared_ptr< Artist > &artist, std::shared_ptr< Album > &album)`
Construtor com título, artista e álbum.
- `Song(unsigned id, std::string file_path, std::string title, unsigned artist_id)`
Construtor básico para criação de música com informações essenciais.
- `Song(unsigned id, const std::string &title, unsigned &artist, unsigned &user_id)`
Construtor para música com ID, título, artista e usuário.
- `Song(const std::string &title, Artist &artist, Album &album, User &user)`
Construtor completo com objetos relacionados.
- `~Song()`

- Destrutor da classe [Song](#).*
- `std::string getFilePath () const`
Obtém o caminho do arquivo.
 - `std::string getTitle () const`
Obtém o título da música.
 - `std::shared_ptr< const Artist > getArtist () const`
Obtém o artista.
 - `std::vector< unsigned > getFeaturingArtistsId () const`
 - `std::vector< std::shared_ptr< const Artist > > getFeaturingArtists ()`
Obtém os artistas colaboradores (featuring)
 - `std::shared_ptr< const Album > getAlbum () const`
Obtém o álbum.
 - `int getDuration () const`
Obtém a duração.
 - `unsigned getTrackNumber () const`
Obtém o número da faixa no album.
 - `std::string getGenre () const`
Obtém o gênero.
 - `int getYear () const`
Obtém o ano de lançamento.
 - `std::shared_ptr< User > getUser () const`
Obtém o usuário dono da música.
 - `void setUser (const User &user)`
Define o usuário dono da música.
 - `void setTitle (const std::string &title)`
Define o título da música.
 - `void setArtist (std::shared_ptr< Artist > &artist)`
Define o artista principal.
 - `void setTrackNumber (unsigned track_number)`
Define o número da faixa no álbum.
 - `void setFeaturingArtists (std::shared_ptr< Artist > &artist)`
 - `void setArtistLoader (const std::function< std::shared_ptr< Artist >()> &loader)`
Define a função para carregar o artista.
 - `void setFeaturingArtistsLoader (const std::function< std::vector< std::shared_ptr< Artist >>()> &loader)`
Define a função para carregar os artistas colaboradores.
 - `void setFeaturingArtists (const std::vector< Artist > &artists)`
Define os artistas colaboradores (featuring)
 - `void setAlbumLoader (const std::function< std::shared_ptr< Album >()> &loader)`
Define função para carregar o álbum.
 - `void setAlbum (const Album &album)`
Define o álbum.
 - `void setGenre (const std::string &genre)`
Define o gênero.
 - `void setYear (int year)`
Define o ano de lançamento.
 - `void setDuration (int sec)`
 - `std::string getFormattedDuration () const`
Carrega metadados do arquivo de áudio.
 - `std::string toString () const`
Converte a música para string.
 - `std::vector< std::shared_ptr< IPlayableObject > > getPlayableObjects () const override`

- *Obtém os objetos reproduzíveis (a própria música)*
• `bool operator== (const Entity &other) const` override
Compara dois `Song`.
- `bool operator!= (const Entity &other) const` override
Compara duas `Song` para desigualdade.
- `std::string getAudioFilePath ()` const override
Obtém o caminho do arquivo de áudio.

Atributos Privados

- `std::string _file_path`
- `std::string _title`
- `unsigned _artist_id`
- `unsigned user_id`
- `User _user`
- `std::shared_ptr< Artist > _artist`
- `std::vector< unsigned > _featuring_artists_ids`
- `unsigned _album_id`
- `std::shared_ptr< Album > _album`
- `int _duration`
- `std::string _genre`
- `int _year`
- `unsigned _track_number`
- `bool _metadata_loaded`
- `ma_decoder_config decoderConfig`
- `std::function< std::shared_ptr< Artist > >> artistLoader`
- `std::function< std::vector< std::shared_ptr< Artist > >> featuringArtistsLoader`
- `std::function< std::shared_ptr< Album > >> albumLoader`

Outros membros herdados

6.6.1 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`

Definição na linha 39 do arquivo `Song.hpp`.

6.6.2 Construtores e Destrutores

6.6.2.1 `Song()` [1/5]

```
core::Song::Song ( )
```

Construtor padrão - inicializa uma música com valores padrão.

6.6.2.2 Song() [2/5]

```
core::Song::Song (
    const std::string & title,
    std::shared_ptr< Artist > & artist,
    std::shared_ptr< Album > & album )
```

Construtor com título, artista e álbum.

Parâmetros

<i>title</i>	Título da música
<i>artist</i>	Ponteiro compartilhado para o artista principal
<i>album</i>	Ponteiro compartilhado para o álbum

6.6.2.3 Song() [3/5]

```
core::Song::Song (
    unsigned id,
    std::string file_path,
    std::string title,
    unsigned artist_id )
```

Construtor básico para criação de música com informações essenciais.

Parâmetros

<i>id</i>	Identificador único da música
<i>file_path</i>	Caminho do arquivo de áudio
<i>title</i>	Título da música
<i>artist_id</i>	Identificador do artista principal

6.6.2.4 Song() [4/5]

```
core::Song::Song (
    unsigned id,
    const std::string & title,
    unsigned & artist,
    unsigned & user_id )
```

Construtor para música com ID, título, artista e usuário.

Parâmetros

<i>id</i>	Identificador único da música
<i>title</i>	Título da música
<i>artist</i>	Referência para o ID do artista
<i>user_id</i>	Referência para o ID do usuário proprietário

6.6.2.5 `Song()` [5/5]

```
core::Song::Song (
    const std::string & title,
    Artist & artist,
    Album & album,
    User & user )
```

Construtor completo com objetos relacionados.

Parâmetros

<i>title</i>	Título da música
<i>artist</i>	Referência para o objeto Artist
<i>album</i>	Referência para o objeto Album
<i>user</i>	Referência para o objeto User proprietário

6.6.2.6 `~Song()`

```
core::Song::~~Song ( )
```

Destrutor da classe [Song](#).

6.6.3 Funções membros

6.6.3.1 `getAlbum()`

```
std::shared_ptr<const Album> core::Song::getAlbum ( ) const
```

Obtém o álbum.

Retorna

Nome do álbum

6.6.3.2 getArtist()

```
std::shared_ptr<const Artist> core::Song::getArtist ( ) const
```

Obtém o artista.

Retorna

Nome do artista/banda

6.6.3.3 getAudioFilePath()

```
std::string core::Song::getAudioFilePath ( ) const [override]
```

Obtém o caminho do arquivo de áudio.

Retorna

Caminho do arquivo de áudio

6.6.3.4 getDuration()

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

Obtém a duração.

Retorna

Duração em segundos

6.6.3.5 getFeaturingArtists()

```
std::vector<std::shared_ptr<const Artist> > core::Song::getFeaturingArtists ( )
```

Obtém os artistas colaboradores (featuring)

Retorna

Vetor de ponteiros compartilhados para os artistas colaboradores

6.6.3.6 getFeaturingArtistsId()

```
std::vector<unsigned> core::Song::getFeaturingArtistsId ( ) const
```

6.6.3.7 getFilePath()

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

Obtém o caminho do arquivo.

Retorna

Caminho completo do arquivo de áudio

6.6.3.8 getFormattedDuration()

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

Carrega metadados do arquivo de áudio.

Retorna

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

Observação

Futuramente será implementado com a biblioteca TagLib

Obtém a duração formatada

Retorna

String com a duração no formato "MM:SS"

6.6.3.9 getGenre()

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

Obtém o gênero.

Retorna

Gênero musical

6.6.3.10 getPlayableObjects()

```
std::vector<std::shared_ptr<IPlayableObject> > core::Song::getPlayableObjects ( ) const [override]
```

Obtém os objetos reproduzíveis (a própria música)

Retorna

Vetor contendo esta música como IPlayableObject

6.6.3.11 getTitle()

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

Obtém o título da música.

Retorna

Título da música

6.6.3.12 getTrackNumber()

```
unsigned core::Song::getTrackNumber ( ) const
```

Obtém o número da faixa no album.

Retorna

Número da faixa

6.6.3.13 getUser()

```
std::shared_ptr<User> core::Song::getUser ( ) const
```

Obtém o usuário dono da música.

Retorna

Usuário dono da música

6.6.3.14 `getYear()`

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

Obtém o ano de lançamento.

Retorna

Ano de lançamento

6.6.3.15 `operator!=()`

```
bool core::Song::operator!= (
    const Entity & other ) const [override], [virtual]
```

Compara duas `Song` para desigualdade.

Parâmetros

<i>other</i>	Song a ser comparada
--------------	--------------------------------------

Retorna

true se as entidades forem diferentes, false caso contrário

Reimplementa [core::Entity](#).

6.6.3.16 operator==()

```
bool core::Song::operator== (
    const Entity & other ) const [override], [virtual]
```

Compara dois [Song](#).

Parâmetros

<i>other</i>	Song a ser comparada
--------------	--------------------------------------

Retorna

true se as entidades forem iguais, false caso contrário

Reimplementa [core::Entity](#).

6.6.3.17 setAlbum()

```
void core::Song::setAlbum (
    const Album & album )
```

Define o álbum.

Parâmetros

<i>album</i>	Novo álbum
--------------	------------

6.6.3.18 setAlbumLoader()

```
void core::Song::setAlbumLoader (
    const std::function< std::shared_ptr< Album >()> & loader )
```

Define função para carregar o álbum.

Parâmetros

<i>loader</i>	Função que retorna um ponteiro compartilhado para o álbum
---------------	---

6.6.3.19 setArtist()

```
void core::Song::setArtist (
    std::shared_ptr< Artist > & artist )
```

Define o artista principal.

Parâmetros

<i>artist</i>	Novo artista
---------------	--------------

6.6.3.20 setArtistLoader()

```
void core::Song::setArtistLoader (
    const std::function< std::shared_ptr< Artist >()> & loader )
```

Define a função para carregar o artista.

Parâmetros

<i>loader</i>	Função que retorna um ponteiro compartilhado para o artista
---------------	---

6.6.3.21 setDuration()

```
void core::Song::setDuration (
    int sec )
```

6.6.3.22 setFeaturingArtists() [1/2]

```
void core::Song::setFeaturingArtists (
    const std::vector< Artist > & artists )
```

Define os artistas colaboradores (featuring)

Parâmetros

<i>artists</i>	Vetor de artistas colaboradores
----------------	---------------------------------

6.6.3.23 setFeaturingArtists() [2/2]

```
void core::Song::setFeaturingArtists (
    std::shared_ptr< Artist > & artist )
```

6.6.3.24 setFeaturingArtistsLoader()

```
void core::Song::setFeaturingArtistsLoader (
    const std::function< std::vector< std::shared_ptr< Artist >>()> & loader )
```

Define a função para carregar os artistas colaboradores.

Parâmetros

<i>loader</i>	Função que retorna um vetor de ponteiros compartilhados para os artistas colaboradores
---------------	--

6.6.3.25 setGenre()

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

Define o gênero.

Parâmetros

<i>genre</i>	Novo gênero
--------------	-------------

6.6.3.26 setTitle()

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

Define o título da música.

Parâmetros

<i>title</i>	Novo título
--------------	-------------

6.6.3.27 `setTrackNumber()`

```
void core::Song::setTrackNumber (
    unsigned track_number )
```

Define o número da faixa no álbum.

Parâmetros

<i>track_number</i>	Novo número da faixa
---------------------	----------------------

6.6.3.28 `setUser()`

```
void core::Song::setUser (
    const User & user )
```

Define o usuário dono da música.

Parâmetros

<i>user</i>	Novo usuário
-------------	--------------

6.6.3.29 `setYear()`

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

Define o ano de lançamento.

Parâmetros

<i>year</i>	Novo ano
-------------	----------

6.6.3.30 toString()

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

Converte a música para string.

Retorna

String com todas as informações da música formatadas

6.6.4 Atributos

6.6.4.1 _album

```
std::shared_ptr<Album> core::Song::_album [mutable], [private]
```

Definição na linha 51 do arquivo Song.hpp.

6.6.4.2 _album_id

```
unsigned core::Song::_album_id [private]
```

Definição na linha 50 do arquivo Song.hpp.

6.6.4.3 _artist

```
std::shared_ptr<Artist> core::Song::_artist [private]
```

Definição na linha 48 do arquivo Song.hpp.

6.6.4.4 _artist_id

```
unsigned core::Song::_artist_id [private]
```

Definição na linha 45 do arquivo Song.hpp.

6.6.4.5 `_duration`

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

Definição na linha 52 do arquivo `Song.hpp`.

6.6.4.6 `_featuring_artists_ids`

```
std::vector<unsigned> core::Song::_featuring_artists_ids [private]
```

Definição na linha 49 do arquivo `Song.hpp`.

6.6.4.7 `_file_path`

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

Definição na linha 43 do arquivo `Song.hpp`.

6.6.4.8 `_genre`

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

Definição na linha 53 do arquivo `Song.hpp`.

6.6.4.9 `_metadata_loaded`

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

Definição na linha 56 do arquivo `Song.hpp`.

6.6.4.10 `_title`

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

Definição na linha 44 do arquivo `Song.hpp`.

6.6.4.11 `_track_number`

```
unsigned core::Song::_track_number [private]
```

Definição na linha 55 do arquivo Song.hpp.

6.6.4.12 `_user`

```
User core::Song::_user [private]
```

Definição na linha 47 do arquivo Song.hpp.

6.6.4.13 `_year`

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

Definição na linha 54 do arquivo Song.hpp.

6.6.4.14 `albumLoader`

```
std::function<std::shared_ptr<Album>>> core::Song::albumLoader [private]
```

Definição na linha 62 do arquivo Song.hpp.

6.6.4.15 `artistLoader`

```
std::function<std::shared_ptr<Artist>>> core::Song::artistLoader [private]
```

Definição na linha 59 do arquivo Song.hpp.

6.6.4.16 `decoderConfig`

```
ma_decoder_config core::Song::decoderConfig [private]
```

Definição na linha 57 do arquivo Song.hpp.

6.6.4.17 featuringArtistsLoader

```
std::function<std::vector<std::shared_ptr<Artist> >> > core::Song::featuringArtistsLoader  
[private]
```

Definição na linha 61 do arquivo Song.hpp.

6.6.4.18 user_id

```
unsigned core::Song::user_id [private]
```

Definição na linha 46 do arquivo Song.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- /home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp

6.7 Referência da Classe core::User

Classe que representa um usuário do sistema.

```
#include <User.hpp>
```

Diagrama de hierarquia para core::User:

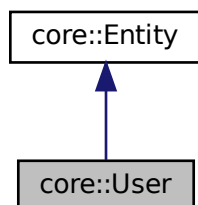
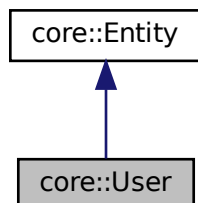


Diagrama de colaboração para core::User:



Membros Públicos

- `User ()`
- `User (const std::string &username)`
Constroi um novo usuário.
- `User (const std::string &username, const std::string &home_path, const std::string &input_path, const userid &uid)`
- `User (unsigned id, const std::string &username, const std::string &home_path, const std::string &input_path, const userid &uid)`
- `~User ()`
Destrutor de um usuário.
- `std::string getUsername () const`
Obtém o nome do usuário.
- `void setUsername (const std::string &username)`
Seta o nome do usuário.
- `std::string getHomePath () const`
Obtém o caminho do diretório home do usuário.
- `void setHomePath (const std::string &home_path)`
Seta o caminho do diretório home do usuário.
- `std::string getInputPath () const`
Obtém o caminho do diretório de entrada de músicas do usuário.
- `void setInputPath (const std::string &input_path)`
Seta o caminho do diretório de entrada de músicas do usuário.
- `userid getUID () const`
Obtém o ID do usuário no OS.
- `void setUID (const userid &uid)`
Seta o ID do usuário no OS.
- `bool isCurrentUser () const`
Verifica se é o usuário atual do sistema.
- `void setIsCurrentUser (bool is_current_user)`
- `bool operator== (const Entity &other) const override`
Compara dois usuários.
- `bool operator!= (const Entity &other) const override`
Compara dois usuários para desigualdade.

Atributos Privados

- `std::string _username`
- `std::string _home_path`
- `std::string _input_path`
- `userid _uid`
- `bool _is_current_user`

Outros membros herdados

6.7.1 Descrição detalhada

Classe que representa um usuário do sistema.

Definição na linha 31 do arquivo User.hpp.

6.7.2 Construtores e Destrutores

6.7.2.1 User() [1/4]

```
core::User::User ( )
```

6.7.2.2 User() [2/4]

```
core::User::User (
    const std::string & username )
```

Constroi um novo usuário.

Parâmetros

<code>_username</code>	Nome do usuário
------------------------	-----------------

6.7.2.3 User() [3/4]

```
core::User::User (
    const std::string & username,
    const std::string & home_path,
    const std::string & input_path,
    const userid & uid )
```

6.7.2.4 User() [4/4]

```
core::User::User (
    unsigned id,
    const std::string & username,
    const std::string & home_path,
    const std::string & input_path,
    const userid & uid )
```

6.7.2.5 ~User()

```
core::User::~~User ( )
```

Destrutor de um usuário.

6.7.3 Funções membros

6.7.3.1 getHomePath()

```
std::string core::User::getHomePath ( ) const
```

Obtém o caminho do diretório home do usuário.

Retorna

String com o caminho do diretório home

6.7.3.2 getInputPath()

```
std::string core::User::getInputPath ( ) const
```

Obtém o caminho do diretório de entrada de músicas do usuário.

Retorna

String com o caminho do diretório de entrada

6.7.3.3 getUID()

```
userid core::User::getUID ( ) const
```

Obtém o ID do usuário no OS.

Retorna

ID do usuário

6.7.3.4 getUsername()

```
std::string core::User::getUsername ( ) const
```

Obtém o nome do usuário.

Retorna

String com o nome do usuário

6.7.3.5 `isCurrentUser()`

```
bool core::User::isCurrentUser ( ) const
```

Verifica se é o usuário atual do sistema.

Retorna

true se for o usuário atual, false caso contrário

6.7.3.6 `operator!=(())`

```
bool core::User::operator!= (
    const Entity & other ) const [override], [virtual]
```

Compara dois usuários para desigualdade.

Parâmetros

<i>other</i>	Outro usuário para comparação
--------------	-------------------------------

Retorna

true se forem diferentes, false caso contrário

Reimplementa `core::Entity`.

6.7.3.7 `operator==(())`

```
bool core::User::operator== (
    const Entity & other ) const [override], [virtual]
```

Compara dois usuários.

Parâmetros

<i>other</i>	Outro usuário para comparação
--------------	-------------------------------

Retorna

true se forem iguais, false caso contrário

Reimplementa `core::Entity`.

6.7.3.8 setHomePath()

```
void core::User::setHomePath (
    const std::string & home_path )
```

Seta o caminho do diretório home do usuário.

Parâmetros

<i>home_path</i>	Novo caminho do diretório home
------------------	--------------------------------

6.7.3.9 setInputPath()

```
void core::User::setInputPath (
    const std::string & input_path )
```

Seta o caminho do diretório de entrada de músicas do usuário.

Parâmetros

<i>input_path</i>	Novo caminho do diretório de entrada
-------------------	--------------------------------------

6.7.3.10 setIsCurrentUser()

```
void core::User::setIsCurrentUser (
    bool is_current_user )
```

6.7.3.11 setUID()

```
void core::User::setUID (
    const userid & uid )
```

Seta o ID do usuário no OS.

Parâmetros

<i>uid</i>	Novo ID do usuário
------------	--------------------

6.7.3.12 setUsername()

```
void core::User::setUsername (
    const std::string & username )
```

Seta o nome do usuário.

Parâmetros

<code>username</code>	Novo nome do usuário
-----------------------	----------------------

6.7.4 Atributos

6.7.4.1 `_home_path`

```
std::string core::User::_home_path [private]
```

Caminho do diretório home do usuário

Definição na linha 34 do arquivo `User.hpp`.

6.7.4.2 `_input_path`

```
std::string core::User::_input_path [private]
```

Caminho do diretório de entrada de músicas do usuário

Definição na linha 35 do arquivo `User.hpp`.

6.7.4.3 `_is_current_user`

```
bool core::User::_is_current_user [private]
```

Indica se é o usuário atual do sistema

Definição na linha 38 do arquivo `User.hpp`.

6.7.4.4 `_uid`

```
userid core::User::_uid [private]
```

ID do usuário no OS

Definição na linha 37 do arquivo User.hpp.

6.7.4.5 `_username`

```
std::string core::User::_username [private]
```

Nome do usuário

Definição na linha 33 do arquivo User.hpp.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- </home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/User.hpp>

Capítulo 7

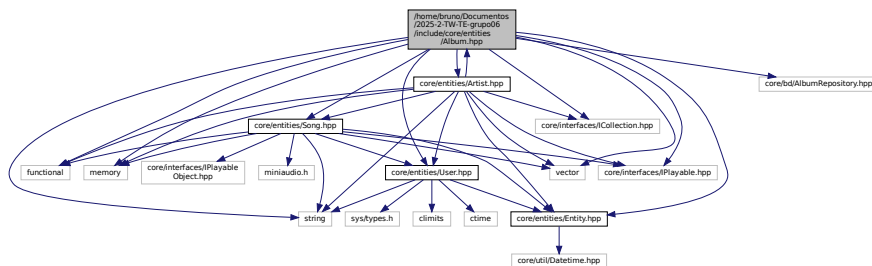
Arquivos

7.1 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp

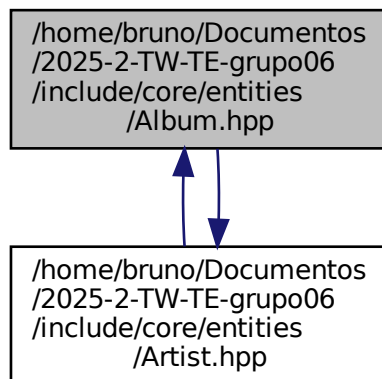
Definição da classe Album para representar uma coleção de músicas de um artista.

```
#include <functional>
#include <memory>
#include <string>
#include <vector>
#include "core/bd/AlbumRepository.hpp"
#include "core/entities/Artist.hpp"
#include "core/entities/Entity.hpp"
#include "core/entities/Song.hpp"
#include "core/entities/User.hpp"
#include "core/interfaces/ICollection.hpp"
#include "core/interfaces/IPlayable.hpp"
```

Gráfico de dependência de inclusões para Album.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class `core::Album`

Representa um álbum musical com suas músicas.

Namespaces

- `core`

7.1.1 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

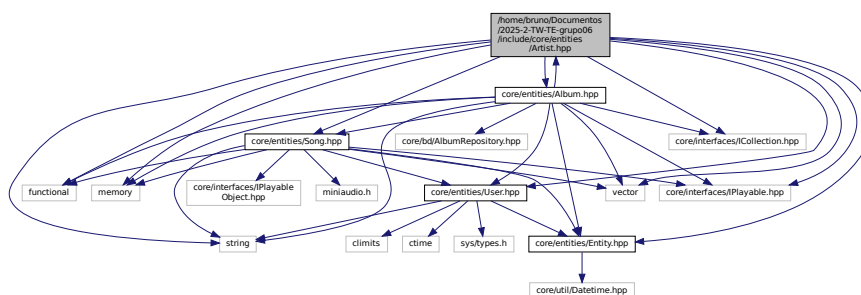
Joao Tavares

Data

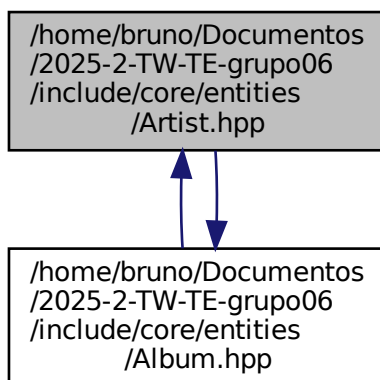
2025-10-09

```
#include "core/entities/Album.hpp"
#include "core/entities/Song.hpp"
#include "core/entities/User.hpp"
#include "core/entities/Entity.hpp"
#include "core/interfaces/ICollection.hpp"
#include "core/interfaces/IPlayable.hpp"
#include <functional>
#include <memory>
#include <string>
#include <vector>
```

Gráfico de dependência de inclusões para Artist.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



- class `core::Artist`

Representa um artista musical com suas músicas e álbuns.

Namespaces

- [core](#)

7.2.1 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

7.3 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/EntitiesFWD.hpp

Namespaces

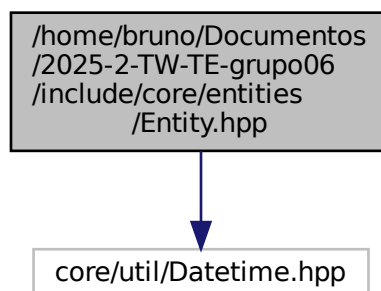
- [core](#)

7.4 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/Entity.hpp

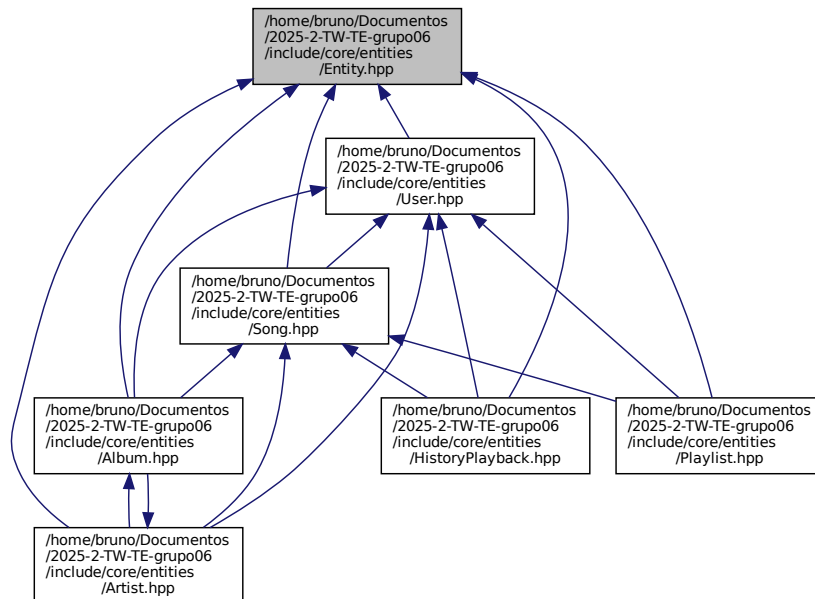
Interface para entidades do sistema.

```
#include "core/util/Datetime.hpp"
```

Gráfico de dependência de inclusões para Entity.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class `core::Entity`

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

Namespaces

- `core`

7.4.1 Descrição detalhada

Interface para entidades do sistema.

Interface que define características essenciais para entidades do sistema.

Autor

Eloy Maciel

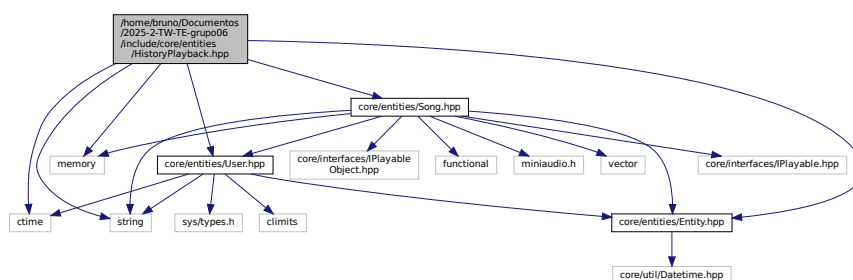
Data

2025-10-09

7.5 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/HistoryPlayback.hpp

Entidade de histórico de reprodução.

```
#include <ctime>
#include <memory>
#include <string>
#include "core/entities/Entity.hpp"
#include "core/entities/Song.hpp"
#include "core/entities/User.hpp"
Gráfico de dependência de inclusões para HistoryPlayback.hpp:
```



Componentes

- class `core::HistoryPlayback`
Entidade de histórico de reprodução.

Namespaces

- `core`

7.5.1 Descrição detalhada

Entidade de histórico de reprodução.

O Arquivo define a entidade `HistoryPlayback`, que representa um registro de reprodução de uma música,

Autor

Eloy Maciel

Data

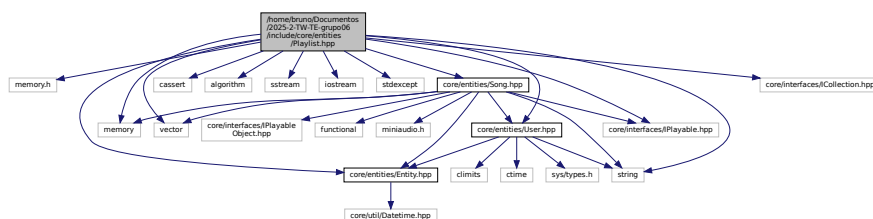
2025-10-12

7.6 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/Playlist.hpp

Definição de uma coleção de músicas escolhidas pelo usuário.

```
#include <memory.h>
#include <memory>
#include <string>
#include <vector>
#include <cassert>
#include <algorithm>
#include <sstream>
#include <iostream>
#include <stdexcept>
#include "core/entities/Entity.hpp"
#include "core/entities/Song.hpp"
#include "core/entities/User.hpp"
#include "core/interfaces/ICollection.hpp"
#include "core/interfaces/IPlayable.hpp"
```

Gráfico de dependência de inclusões para Playlist.hpp:



Componentes

- class `core::Playlist`
Representa uma playlist de músicas.

Namespaces

- core

7.6.1 Descrição detalhada

Definição de uma coleção de músicas escolhidas pelo usuário.

Essa classe armazena um conjunto de músicas à gosto do usuário.

Autor

Pedro Gabriel

Data

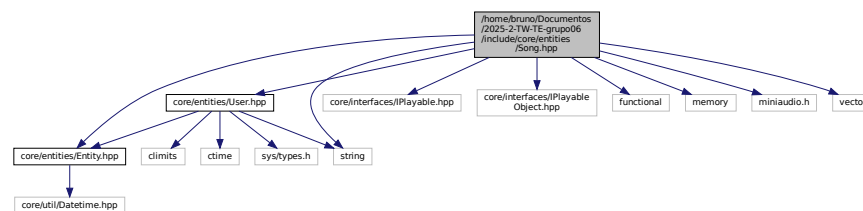
2025-10-13

7.7 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp

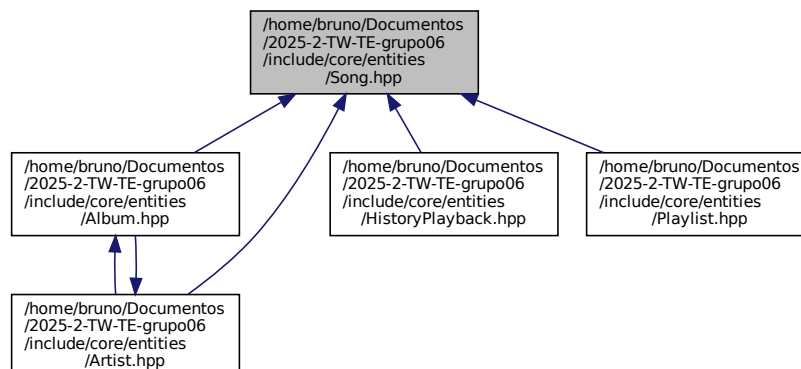
Definição da classe Song para representar uma música no sistema.

```
#include "core/entities/Entity.hpp"
#include "core/entities/User.hpp"
#include "core/interfaces/IPlayable.hpp"
#include "core/interfaces/IPlayableObject.hpp"
#include <functional>
#include <memory>
#include <miniaudio.h>
#include <string>
#include <vector>
```

Gráfico de dependência de inclusões para Song.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class `core::Song`
Representa uma música com seus metadados.

Namespaces

- `core`

7.7.1 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

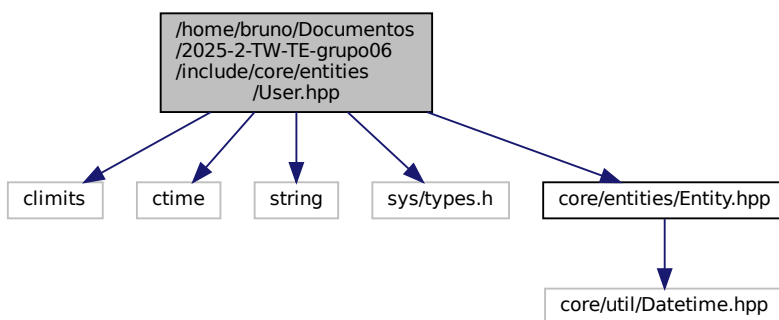
2025-10-09

7.8 Referência do Arquivo /home/bruno/Documents/2025-2-TW-TE-grupo06/include/core/entities/User.hpp

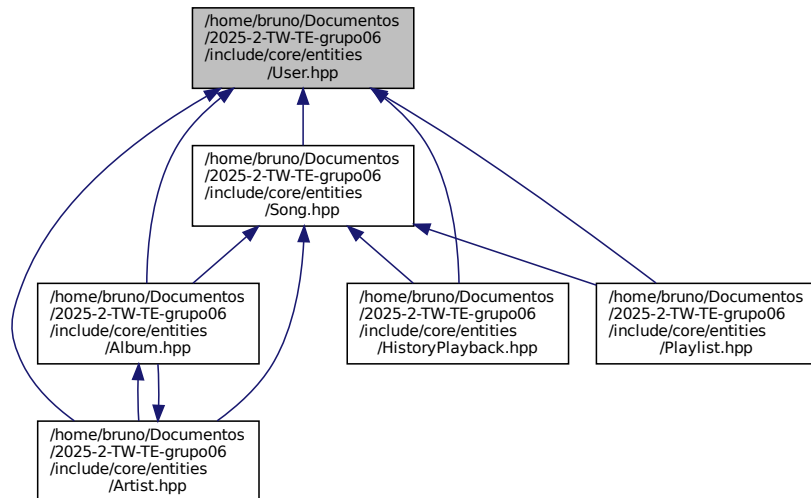
entidade para usuários do sistema

```
#include <climits>
#include <ctime>
#include <string>
#include <sys/types.h>
#include "core/entities/Entity.hpp"
```

Gráfico de dependência de inclusões para User.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class `core::User`

Classe que representa um usuário do sistema.

Namespaces

- `core`

Definições de Tipos

- using `core::userid` = `uid_t`

7.8.1 Descrição detalhada

entidade para usuários do sistema

Autor

Bruno Vieira

Versão

0.1

Data

2025-10-15

Índice Remissivo

/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Album.hpp, 89
_loader
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Playlist.hpp, 60
_metadata_loaded
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/SongsFWD.hpp, 70
_name
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/AlbumEntity.hpp, 26
_name
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/HistoryPlayback.hpp, 54
_name
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Playlist.hpp, 54
_name
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/Song.hpp, 26
_name
/home/bruno/Documentos/2025-2-TW-TE-grupo06/include/core/entities/User.hpp, 44
_name
_album
core::Song, 78
_album_id
core::Song, 78
_albums
core::Artist, 43
_artist
core::Album, 25
core::Song, 78
_artist_id
core::Album, 25
core::Song, 78
_dataCriacao
core::Entity, 49
_duration
core::Song, 78
_featuring_artists_ids
core::Album, 25
core::Song, 79
_file_path
core::Album, 25
core::Song, 79
_genre
core::Album, 26
core::Artist, 44
core::Song, 79
_home_path
core::User, 87
_id
core::Entity, 49
_input_path
core::User, 87
_is_current_user
_title
core::Song, 79
_titulo
core::Playlist, 64
_track_number
core::Song, 79
_uid
core::User, 87
_user
core::Album, 26
core::Artist, 44
core::HistoryPlayback, 55
core::Playlist, 64
core::Song, 80
_user_id
core::Album, 26
core::Artist, 44
core::Playlist, 64
_username
core::User, 88
_year
core::Album, 26
core::Song, 80
~Album
core::Album, 15
~Artist
core::Artist, 31
~Entity
core::Entity, 46
~HistoryPlayback
core::HistoryPlayback, 52
~Playlist
core::Playlist, 57

- ~Song
 - core::Song, 69
- ~User
 - core::User, 83
- addAlbum
 - core::Artist, 31
- addSong
 - core::Album, 15
 - core::Artist, 32
 - core::Playlist, 58
- addSongAlbum
 - core::Artist, 32
- Album
 - core::Album, 14
- albumLoader
 - core::Song, 80
- albumsLoader
 - core::Artist, 44
- Artist
 - core::Artist, 30, 31
- artistLoader
 - core::Album, 27
 - core::Song, 80
- calculateTotalDuration
 - core::Album, 15
 - core::Artist, 32
 - core::Playlist, 58
- core, 9
 - userid, 9
- core::Album, 11
 - _artist, 25
 - _artist_id, 25
 - _featuring_artists_ids, 25
 - _file_path, 25
 - _genre, 26
 - _name, 26
 - _songs, 26
 - _user, 26
 - _user_id, 26
 - _year, 26
 - ~Album, 15
 - addSong, 15
 - Album, 14
 - artistLoader, 27
 - calculateTotalDuration, 15
 - featuringArtistsLoader, 27
 - findSongById, 15
 - findSongByTitle, 16
 - getArtist, 16
 - getAudioFilePath, 16
 - getFeaturingArtists, 17
 - getFormattedDuration, 17
 - getGenre, 17
 - getName, 17
 - getNextSong, 18
 - getPlayableObjects, 19
 - getPreviousSong, 19
 - getSongAt, 19
 - getSongCount, 20
 - getSongs, 20
 - getUser, 20
 - getYear, 20
 - operator!=, 20
 - operator==, 21
 - removeSong, 21
 - setArtist, 21
 - setArtistLoader, 22
 - setFeaturingArtists, 22
 - setFeaturingArtistsLoader, 22
 - setFilePath, 23
 - setGenre, 23
 - setSongsLoader, 23
 - setUser, 24
 - setYear, 24
 - songsLoader, 27
 - switchSong, 24
 - toString, 25
- core::Artist, 28
 - _albums, 43
 - _genre, 44
 - _name, 44
 - _songs, 44
 - _user, 44
 - _user_id, 44
 - ~Artist, 31
 - addAlbum, 31
 - addSong, 32
 - addSongAlbum, 32
 - albumsLoader, 44
 - Artist, 30, 31
 - calculateTotalDuration, 32
 - findAlbumByName, 33
 - findSongById, 33
 - findSongByTitle, 34
 - getAlbums, 34
 - getAlbumsCount, 34
 - getFormattedDuration, 34
 - getGenre, 35
 - getName, 35
 - getNextSong, 35
 - getNextSongAlbum, 36
 - getPlayableObjects, 36
 - getPreviousSong, 36
 - getPreviousSongAlbum, 37
 - getSongAt, 37
 - getSongAtAlbum, 37
 - getSongs, 38
 - getSongsAlbum, 38
 - getSongsCount, 39
 - getTotalDuration, 39
 - getUser, 39
 - hasAlbum, 39
 - hasSong, 39
 - operator!=, 40
 - operator==, 40

- removeAlbum, 40
- removeSong, 41
- removeSongAlbum, 41
- setAlbumsLoader, 42
- setGenre, 42
- setName, 42
- setSongsLoader, 42
- setUser, 43
- songsLoader, 45
- switchSong, 43
- toString, 43
- core::Entity, 45
 - _dataCriacao, 49
 - _id, 49
 - ~Entity, 46
 - Entity, 46
 - getDataCriacao, 46
 - getId, 47
 - operator!=, 47
 - operator<, 47
 - operator==, 48
 - setDataCriacao, 48
 - setId, 48
- core::HistoryPlayback, 49
 - _played_at, 54
 - _song, 54
 - _user, 55
 - ~HistoryPlayback, 52
 - getPlayedAt, 52
 - getSong, 52
 - getUser, 52
 - HistoryPlayback, 51
 - operator!=, 52
 - operator==, 53
 - setPlayedAt, 53
 - setSong, 53
 - setUser, 54
 - toString, 54
- core::Playlist, 55
 - _loader, 64
 - _songs, 64
 - _titulo, 64
 - _user, 64
 - _user_id, 64
 - ~Playlist, 57
 - addSong, 58
 - calculateTotalDuration, 58
 - findSongById, 58
 - findSongByTitle, 59
 - getFormattedDuration, 59
 - getNextSong, 59
 - getPlayableObjects, 60
 - getPreviousSong, 60
 - getSongAt, 60
 - getSongs, 61
 - getTitle, 61
 - getTitulo, 61
 - getUser, 61
 - operator!=, 61
 - operator==, 62
 - Playlist, 57
 - removeSong, 62
 - setSongsLoader, 62
 - setTitulo, 63
 - setUser, 63
 - switchSong, 63
- core::Song, 65
 - _album, 78
 - _album_id, 78
 - _artist, 78
 - _artist_id, 78
 - _duration, 78
 - _featuring_artists_ids, 79
 - _file_path, 79
 - _genre, 79
 - _metadata_loaded, 79
 - _title, 79
 - _track_number, 79
 - _user, 80
 - _year, 80
 - ~Song, 69
 - albumLoader, 80
 - artistLoader, 80
 - decoderConfig, 80
 - featuringArtistsLoader, 80
 - getAlbum, 69
 - getArtist, 69
 - getAudioFilePath, 70
 - getDuration, 70
 - getFeaturingArtists, 70
 - getFeaturingArtistsId, 70
 - getFilePath, 71
 - getFormattedDuration, 71
 - getGenre, 71
 - getPlayableObjects, 71
 - getTitle, 72
 - getTrackNumber, 72
 - getUser, 72
 - getYear, 72
 - operator!=, 73
 - operator==, 74
 - setAlbum, 74
 - setAlbumLoader, 74
 - setArtist, 75
 - setArtistLoader, 75
 - setDuration, 75
 - setFeaturingArtists, 75, 76
 - setFeaturingArtistsLoader, 76
 - setGenre, 76
 - setTitle, 76
 - setTrackNumber, 77
 - setUser, 77
 - setYear, 77
 - Song, 67–69
 - toString, 77
 - user_id, 81

core::User, [81](#)
 _home_path, [87](#)
 _input_path, [87](#)
 _is_current_user, [87](#)
 _uid, [87](#)
 _username, [88](#)
 ~User, [83](#)
 getHomePath, [84](#)
 getInputPath, [84](#)
 getUID, [84](#)
 getUsername, [84](#)
 isCurrentUser, [84](#)
 operator!=, [85](#)
 operator==, [85](#)
 setHomePath, [85](#)
 setInputPath, [86](#)
 setIsCurrentUser, [86](#)
 setUID, [86](#)
 setUsername, [86](#)
 User, [83](#)

 decoderConfig
 core::Song, [80](#)

 Entity
 core::Entity, [46](#)

 featuringArtistsLoader
 core::Album, [27](#)
 core::Song, [80](#)
 findAlbumByName
 core::Artist, [33](#)
 findSongById
 core::Album, [15](#)
 core::Artist, [33](#)
 core::Playlist, [58](#)
 findSongByTitle
 core::Album, [16](#)
 core::Artist, [34](#)
 core::Playlist, [59](#)

 getAlbum
 core::Song, [69](#)
 getAlbums
 core::Artist, [34](#)
 getAlbumsCount
 core::Artist, [34](#)
 getArtist
 core::Album, [16](#)
 core::Song, [69](#)
 getAudioFilePath
 core::Album, [16](#)
 core::Song, [70](#)
 getDataCriacao
 core::Entity, [46](#)
 getDuration
 core::Song, [70](#)
 getFeaturingArtists
 core::Album, [17](#)
 core::Song, [70](#)
 getFeaturingArtistsId
 core::Song, [70](#)
 getFilePath
 core::Song, [71](#)
 getFormattedDuration
 core::Album, [17](#)
 core::Artist, [34](#)
 core::Playlist, [59](#)
 core::Song, [71](#)
 getGenre
 core::Album, [17](#)
 core::Artist, [35](#)
 core::Song, [71](#)
 getHomePath
 core::User, [84](#)
 getId
 core::Entity, [47](#)
 getInputPath
 core::User, [84](#)
 getName
 core::Album, [17](#)
 core::Artist, [35](#)
 getNextSong
 core::Album, [18](#)
 core::Artist, [35](#)
 core::Playlist, [59](#)
 getNextSongAlbum
 core::Artist, [36](#)
 getPlayableObjects
 core::Album, [19](#)
 core::Artist, [36](#)
 core::Playlist, [60](#)
 core::Song, [71](#)
 getPlayedAt
 core::HistoryPlayback, [52](#)
 getPreviousSong
 core::Album, [19](#)
 core::Artist, [36](#)
 core::Playlist, [60](#)
 getPreviousSongAlbum
 core::Artist, [37](#)
 getSong
 core::HistoryPlayback, [52](#)
 getSongAt
 core::Album, [19](#)
 core::Artist, [37](#)
 core::Playlist, [60](#)
 getSongAtAlbum
 core::Artist, [37](#)
 getSongCount
 core::Album, [20](#)
 getSongs
 core::Album, [20](#)
 core::Artist, [38](#)
 core::Playlist, [61](#)
 getSongsAlbum
 core::Artist, [38](#)

getSongsCount
 core::Artist, 39
getTitle
 core::Playlist, 61
 core::Song, 72
getTitulo
 core::Playlist, 61
getTotalDuration
 core::Artist, 39
getTrackNumber
 core::Song, 72
getUID
 core::User, 84
getUser
 core::Album, 20
 core::Artist, 39
 core::HistoryPlayback, 52
 core::Playlist, 61
 core::Song, 72
getUsername
 core::User, 84
getYear
 core::Album, 20
 core::Song, 72

hasAlbum
 core::Artist, 39
hasSong
 core::Artist, 39
HistoryPlayback
 core::HistoryPlayback, 51

isCurrentUser
 core::User, 84

operator!=
 core::Album, 20
 core::Artist, 40
 core::Entity, 47
 core::HistoryPlayback, 52
 core::Playlist, 61
 core::Song, 73
 core::User, 85
operator<
 core::Entity, 47
operator==
 core::Album, 21
 core::Artist, 40
 core::Entity, 48
 core::HistoryPlayback, 53
 core::Playlist, 62
 core::Song, 74
 core::User, 85

Playlist
 core::Playlist, 57

removeAlbum
 core::Artist, 40

removeSong
 core::Album, 21
 core::Artist, 41
 core::Playlist, 62
removeSongAlbum
 core::Artist, 41

setAlbum
 core::Song, 74
setAlbumLoader
 core::Song, 74
setAlbumsLoader
 core::Artist, 42
setArtist
 core::Album, 21
 core::Song, 75
setArtistLoader
 core::Album, 22
 core::Song, 75
setDataCriacao
 core::Entity, 48
setDuration
 core::Song, 75
setFeaturingArtists
 core::Album, 22
 core::Song, 75, 76
setFeaturingArtistsLoader
 core::Album, 22
 core::Song, 76
setFilePath
 core::Album, 23
setGenre
 core::Album, 23
 core::Artist, 42
 core::Song, 76
setHomePath
 core::User, 85
setId
 core::Entity, 48
setInputPath
 core::User, 86
setIsCurrentUser
 core::User, 86
setName
 core::Artist, 42
setPlayedAt
 core::HistoryPlayback, 53
setSong
 core::HistoryPlayback, 53
setSongsLoader
 core::Album, 23
 core::Artist, 42
 core::Playlist, 62
setTitle
 core::Song, 76
setTitulo
 core::Playlist, 63
setTrackNumber
 core::Song, 77

- setUID
 - core::User, [86](#)
- setUser
 - core::Album, [24](#)
 - core::Artist, [43](#)
 - core::HistoryPlayback, [54](#)
 - core::Playlist, [63](#)
 - core::Song, [77](#)
- setUsername
 - core::User, [86](#)
- setYear
 - core::Album, [24](#)
 - core::Song, [77](#)
- Song
 - core::Song, [67–69](#)
- songsLoader
 - core::Album, [27](#)
 - core::Artist, [45](#)
- switchSong
 - core::Album, [24](#)
 - core::Artist, [43](#)
 - core::Playlist, [63](#)
- toString
 - core::Album, [25](#)
 - core::Artist, [43](#)
 - core::HistoryPlayback, [54](#)
 - core::Song, [77](#)
- User
 - core::User, [83](#)
- user_id
 - core::Song, [81](#)
- userid
 - core, [9](#)