DocumentaçãoProjetoFinal-PDS2

Generated by Doxygen 1.8.17

Chapter 1

Projeto Final - Programação e Desenvolvimento de Software 2

Nosso projeto final do curso de Programação e Desenvolvimento de Software 2 teve como objetivo a integração dos conceitos aprendidos, como orientação a objetos, boas práticas de programação e programação defensiva. Em nosso sistema para uma Locadora de Filmes, permitimos a automatização de processos fundamentais e repetitivos, tais como o cadastro, a listagem, a remoção, o aluguel e a devolução.

1.1 O andamento do projeto

Agora, abordando sobre o desenvolvimento do projeto e sobre as principais dificuldades encontradas pelo grupo, pode-se citar que foi de comum acordo à todos os membros a existência de obstáculos no que tange à elaboração e planejamento de como ocorreria a criação e funcionamento do código-fonte de uma forma geral, principalmente no quesito da utilização das classes (decidir o que cada classe faria, quais métodos seriam utilizados em cada uma, entre outras dúvidas). Além disso, a utilização de novas ferramentas, como o Doxygen5 para a elaboração de uma documentação de código mais profissional e a necessidade da realização de testes, submetendo o projeto à diversos ambientes de situações-problema com o obejtivo de verificar possíveis erros durante seu funcionamento também se configuraram como adversidades durante o desenvolvimento deste projeto final. Vale ressaltar também que, a tentativa da elaboração de uma interface mais agradável e dinâmica ao usúario fez com que o grupo se sentisse na necessidade de utilizar ferramentas mais avançadas para alcançar esse objetivo.

1.2 Nosso sistema

Falando mais sobre o funcionamento do código, foi desenvolvido um sistema de cadastro para os clientes de forma automatizada e dinâmica, um sistema de realização de listagem de filmes e clientes, de acordo com a opção de ordenação desejada pelo usuário e, também, um conjunto de sistemas integrados que auxiliam na remoção, no aluguel e na devolução de filmes de maneira prática, rápida e fácil pelos clientes da locadora.

Além disso, nosso grupo adionou algumas funcionalidades adicionais:

1.2.1 Sistema de Fidelidade:

Nosso sistema tem um sistema de fidelidade, que permite que a cada 10 filmes alugados pelo cliente, ele ganhe um desconto de 10% no valor total em sua próxima devolução. Essa iniciativa visa recompensar nossos clientes frequentes, oferecendo-lhes vantagens e incentivando os aluguéis.

1.2.2 Sistema de Bloqueio:

Em nosso sistema, implementamos uma política de segurança que visa garantir a integridade do processo de locação de filmes. Se um cliente alugou filmes e ainda não devolveu, ele não pode realizar novos aluguéis até que devolva os filmes anteriores.

1.2.3 Sistema de Avaliação:

Nosso sistema também tem uma funcionalidade de avaliação de filmes, que deve ser incentivada a ser utilizada pelos clientes. Na hora de listagem dos filmes, a avaliação do filme é exibida, calculada a partir de uma média de todas as avaliações, considerando sempre uma avaliação pioneira 4.

1.2.4 Sistema de Recomendação:

Por fim, nós desenvolvemos também um sistema de recomendação que permite a sugestão de filmes para clientes baseado em seu histórico de aluguéis na locadora. Nossa funcionalidade compara o gosto do nosso cliente com os demais clientes cadastrados na locadora, e a partir dos 3 clientes mais similares, define sugestões de filmes que ainda não foram assistidos pelo cliente.

1.3 Conclusão

Em conclusão, este projeto final do curso de Programação e Desenvolvimento de Software 2 representou uma oportunidade valiosa para o grupo aplicar os conhecimentos teóricos adquiridos em um contexto prático, enfrentando e superando desafios reais de desenvolvimento de software. Através da criação de um sistema para uma Locadora de Filmes, a equipe não só desenvolveu habilidades técnicas em programação e design de sistemas, mas também aprendeu a importância da colaboração, do planejamento eficaz e da adaptação a novas ferramentas e tecnologias.

O sucesso do projeto pode ser medido não apenas pela funcionalidade do sistema entregue, mas também pelo ganho de conhecimento e de habilidaes pessoais de cada membro da equipe. As adversidades encontradas serviram como lições valiosas, ensinando o grupo a buscar soluções criativas e a manter a resiliência diante dos obstáculos.

Este projeto também contribuiu significativamente para entendermos como seria a construção e o possível funcionamento automatizado de um software para uma locadora de filmes, o qual objetiva automatizar tarefas rotineiras e melhorar a experiência do usuário. A interface dinâmica e a facilidade de uso do sistema são testemunhos do esforço e da dedicação do grupo em fazer isso acontecer e, com certeza, todo o conhecimemto adquirido pode ser replicado em ambientes profissionais.

Dessa forma, se torna notório que, com a conclusão bem-sucedida deste projeto, os membros do grupo se sentem mais preparados e confiantes para enfrentar possíveis desafios futuros na área profissional de desenvolvimento de software.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

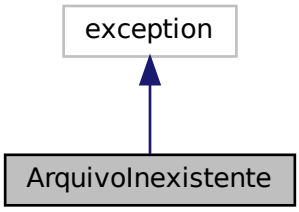
Class Documentation

5.1 ArquivoInexistente Class Reference

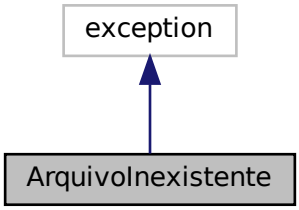
Classe de exceção para arquivos inexistentes.

`#include <erros.hpp>`

Inheritance diagram for ArquivoInexistente:



Collaboration diagram for ArquivoInexistente:



Public Member Functions

- virtual const char * what () const noexcept override

5.1.1 Detailed Description

Classe de exceção para arquivos inexistentes.

Essa classe é usada para exibir uma mensagem de erro para quando um arquivo inexistente é digitado pelo usuário.

5.1.2 Member Function Documentation

5.1.2.1 what()

```
virtual const char* ArquivoInexistente::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

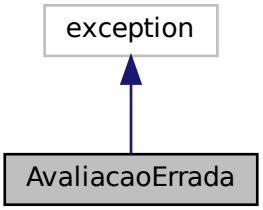
- errors/erros.hpp

5.2 AvaliacaoErrada Class Reference

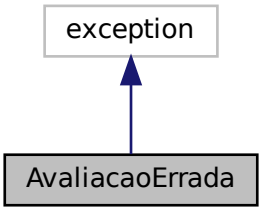
Classe de exceção para avaliações erradas.

```
#include <erros.hpp>
```

Inheritance diagram for AvaliacaoErrada:



Collaboration diagram for AvaliacaoErrada:



Public Member Functions

- virtual const char * what () const noexcept override

5.2.1 Detailed Description

Classe de exceção para avaliações erradas.

Essa classe é usada para exibir uma mensagem de erro para quando uma nota de avaliação errada é digitada pelo usuário.

5.2.2 Member Function Documentation

5.2.2.1 what()

```
virtual const char* AvaliacaoErrada::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

- errors/erros.hpp

5.3 Cliente Class Reference

```
#include <cliente.hpp>
```

Public Member Functions

- Cliente (long long cpf, string nome)

  Construtor do arquivo "cliente.hpp" para identificação dos clientes.
- ∼Cliente ()

  Destrutor da classe "Cliente".
- void lerCliente ()

  Método "lerCliente" classe "Cliente", utilizado especialmente para exibição.
- void alugar (Filme ∗filme)

  Método "alugar" da classe "Cliente", utilizado para registro.
- void devolver ()

  Método "devolver" da classe "Cliente", utilizado especialmente para controle.
- void recomendar (vector< Cliente ∗ > clientes)

  Método "recomendar" da classe "Cliente", utilizado para fins de recomendação.
- int calcularSimilaridade (Cliente ∗cliente)

  Método "calcularSimilaridade" da classe "Cliente", utilizado para controle.
- void definirSimilares (vector< Cliente ∗ > clientes)

  Método "definirSimilares" da classe "Cliente", utilizado para fins de recomendação.
- vector< Filme ∗ > recomendarPorSimilar (Cliente ∗cliente)

  Método "recomendarPorSimilar" da classe "Cliente", utilizado para fins de recomendação.

Public Attributes

- int pontos
- vector< Filme ∗ > filmesAlugados
- vector< Filme ∗ > historico
- vector< Cliente ∗ > similares
- vector< Filme ∗ > recomendados

Protected Attributes

- string nome
- long long cpf
- bool bloqueado

Friends

- class Locadora

5.3.1 Detailed Description

Na classe "Cliente", a qual possui como classe amiga "Locadora", existem como atributos protegidos o nome do cliente ("string nome") e seu cpf ("long long cpf"). A classe também contém como atributos públicos um construtor que recebe o cpf e o nome do cliente ("Cliente(long long cpf, string nome)") um destrutor ("∼Cliente()") e os pontos dos clientes com base nos filmes alugados ("int pontos"). Existe também um método para identificação de clientes ("void lerCliente()"), um método para controlar a quantidade e quais filmes foram alugados ("void alugar (Filme∗ filme)"), um método para controlar a quantidade e quais filmes foram devolvidos ("void devolver ()"), um método para recomendar filmes aos clientes ("void recomendar(vector <Cliente∗> clientes)"), um método para calcular a quantidade de filmes similares alugados pelos clientes ("int calcularSimilaridade (Cliente∗ cliente)"), um método para definir quais clientes possuem gostos similares ("void definirSimilares (vector <Cliente∗> clientes)"), um vetor de ponteiros para controlar quais filmes foram alugados ("vector<Filme∗> filmesAlugados") um vetor de ponteiros para adicionar filmes ao histórico dos clientes ("vector<Filme∗> historico"), um vetor de ponteiros de filmes recomendados ("vector<Filme∗> recomendados"), um vetor de ponteiros de filmes similares ("vector<↩ Cliente∗> similares") e, por fim, um vetor de ponteiros para recomendar os filmes aos clientes com base na similaridade entre os gostos dos mesmos ("vector <Filme∗> recomendarPorSimilar(Cliente∗ cliente)").

Parameters

nome	Registra o nome do cliente.
cpf	Registra o cpf do cliente.
recomendados	Usado na recomendação de filmes aos clientes.
similares	Usado no calculo de similaridade entre os filmes alugados pelos clientes.
recomendarPorSimilar	Usado na recomendação de filmes por similaridade entre os clientes.
pontos	Registra a quantidade de pontos de cada cliente.
filmesAlugados	Usado no controle dos filmes alugados.
historico	Usado no controle dos filmes adicionados ao histórico dos clientes.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Cliente()

```
Cliente::Cliente (
            long long cpf,
            string nome )
```

Construtor do arquivo "cliente.hpp" para identificação dos clientes.

Esse construtor registra o cpf, o nome e a quantidade de pontos que cada cliente tem, para controle da própria locadora, além de auxiliar em futuras promoções(pontos).

Parameters

cpf	Armazena o cpf do cliente para identificação.
nome	Armazena o nome do cliente.
pontos	Armazena a quantidade de pontos que cada cliente tem para falicitar o controle de futuras promoções.

5.3.2.2 ∼Cliente()

```
Cliente::∼Cliente ( )
```

Destrutor da classe "Cliente".

Esse destrutor trabalha na liberação de memória alocada nos vetores de ponteiros da classe "Cliente", fazendo o controle manual dessa desalocação. Para cada elemento dentro desses vetores, ele realiza uma desalocação, ou seja, deleta esse elemento do vetor.

5.3.3 Member Function Documentation

5.3.3.1 alugar()

```
void Cliente::alugar (
            Filme * filme )
```

Método "alugar" da classe "Cliente", utilizado para registro.

Esse método registra a quantidade de filmes alugados pelo cliente, lista esses filmes, adiciona pontos para uma possível futura promoção e registra o histórico de filmes que já foram alugados pelo cliente.

Parameters

unidades	Indica a quantidade de filmes alugados pelo cliente.
filme	Indica o filme alugado pelo cliente.
filmesAlugados	Armazena quais filmes estão alugados da locadora pelos clientes.
historico	Armazena os filmes alugados pelo cliente em seu historico.

5.3.3.2 calcularSimilaridade()

```
int Cliente::calcularSimilaridade (
            Cliente * cliente2 )
```

Método "calcularSimilaridade" da classe "Cliente", utilizado para controle.

Esse método analisa o filme alugado por um clinte diretamente do seu histórico e compara esse filme com os filmes alugados por outro cliente, também de acordo com seu histórico. Caso esses filmes sejam iguais, existe uma similaridade entre os gostos do clientes, a qual aumenta (a similaridade será importante para o sistema de recomendação).

Parameters

similaridade	Registra a quantidade de filmes similares com base no histórico dos clientes.
filme1	Filme pertencente ao histórico do cliente 1.
filme2	Filme pertencente ao histórico do cliente 2.

Returns

Retorna a quantidade de filmes similares entre os dois clientes.

5.3.3.3 definirSimilares()

```
void Cliente::definirSimilares (
            vector< Cliente * > clientes )
```

Método "definirSimilares" da classe "Cliente", utilizado para fins de recomendação.

Esse método recebe um vetor de clientes e, a partir daí, realiza uma filtragem, em busca de selecionar os 3 usuários com maior similaridade com o cliente atual, e colocá-los no seu atributo "similares". Caso esse vetor parâmetro possua menos que três clientes, essa comparação não precisa ser realizada, e os clientes já são alocados nos similares imediatamente. Caso haja mais de 3, a seleção prossegue. Para cada cliente do parâmetro, é calculado o grau de similaridade, e comparado com os 3 clientes mais similares. Se o grau de similaridade for maior que algum, esse ranking é reformulado para encaixar o novo cliente similar. O cliente com maior similaridade fica em primeiro lugar e assim por diante. Depois de percorrer todo o vetor, os 3 clientes são adicionados ao vetor "similares" de acordo com o ranking previamente feito.

Parameters

clientes	Armazena os clientes.
similares	Armazena os 3 clientes de acordo com a similaridade deles, utilizando como base o "ranking".
c1	Cliente com maior grau de similaridade àquele recebido pelo método.
c2	Cliente com grau de similaridade intermediário àquele recebido pelo método.
c3	Cliente com menor grau de similaridade àquele recebido pelo método.
s1	Maior grau de similaridade.
s2	Grau de similaridade intermediário.
s3	Menor grau de similaridade.

5.3.3.4 devolver()

```
void Cliente::devolver ( )
```

Método "devolver" da classe "Cliente", utilizado especialmente para controle.

Esse método controla a quantidade e quais filmes foram alugados pelos clientes (o filme devolvido à locadora é retirado do vetor filmesAlugados, para controle).

5.3.3.5 lerCliente()

```
void Cliente::lerCliente ( )
```

Método "lerCliente" classe "Cliente", utilizado especialmente para exibição.

Esse método exibe para o usuário o cpf e o nome do cliente registrado na locadora.

5.3.3.6 recomendar()

```
void Cliente::recomendar (
            vector< Cliente * > clientes )
```

Método "recomendar" da classe "Cliente", utilizado para fins de recomendação.

Esse método recebe como parâmetro um vetor de clientes e retorna, a partir deles, um vetor de filmes recomendados. Para isso, ele começa definindo os clientes de gosto mais similar ao cliente atual, pelo método "definir↩ Similares". Caso depois dessa chamada não existam clientes similares ou o cliente atual ainda não tenha alugado nenhum filme na locadora, o método retorna que ainda é impossível recomendar filmes à esse cliente. Se não, é utilizado o método "recomendarPorSimilar" para que cada um dos clientes no vetor "similares" forneça um vetor de filmes, que se juntarão em um só vetor de filmes recomendados, que fica no atributo "recomendados".

Parameters

definirSimilares	Define os filmes similares com base em um cliente específico.

5.3.3.7 recomendarPorSimilar()

```
vector< Filme * > Cliente::recomendarPorSimilar (
            Cliente * cliente )
```

Método "recomendarPorSimilar" da classe "Cliente", utilizado para fins de recomendação.

Esse método recebe um cliente como parâmetro e percorre os filmes assistidos por ele, comparando-os com os filmes no histórico do cliente atual. Caso o cliente atual não tenha visto o filme, o filme é considerado inédito. Caso contrário, o filme é ignorado. Pós isso, os filmes inéditos são adiconados à um outro vetor ("conjunto↩ Recomendados"). Caso existam filmes repetidos, eles são excluidos utilizando "set".

Parameters

recomendados	Armazena os filmes do cliente que foram considerados recomendados.
conjuntoRecomendados	Armazena os filmes recomendados considerados inétidos.
aindaTemFilmes	Determina se ainda existem filmes no vetor "historico".
inedito	Determina se o filme é inétido ou não com base nos filmes do cliente com elevada similaridade.
filmeVisto	Determina se o filme foi visto ou não por algum outro cliente específico.
recomendado	Registra o filme recomendado com base no histórico do cliente de elevada similaridade.

5.3.4 Friends And Related Function Documentation

5.3.4.1 Locadora

```
friend class Locadora  [friend]
```

5.3.5 Member Data Documentation

5.3.5.1 bloqueado

```
bool Cliente::bloqueado  [protected]
```

5.3.5.2 cpf

```
long long Cliente::cpf  [protected]
```

5.3.5.3 filmesAlugados

```
vector<Filme*> Cliente::filmesAlugados
```

5.3.5.4 historico

```
vector<Filme*> Cliente::historico
```

5.3.5.5 nome

```
string Cliente::nome  [protected]
```

5.3.5.6 pontos

```
int Cliente::pontos
```

5.3.5.7 recomendados

```
vector<Filme*> Cliente::recomendados
```

5.3.5.8 similares

```
vector<Cliente*> Cliente::similares
```

The documentation for this class was generated from the following files:

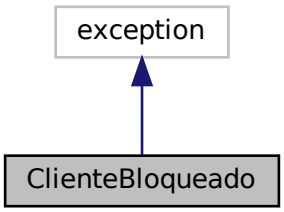
- Cliente/cliente.hpp
- Cliente/cliente.cpp

5.4 ClienteBloqueado Class Reference

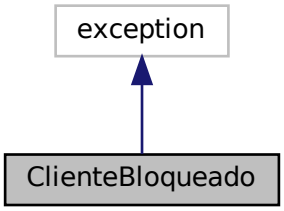
Classe de exceção para clientes bloqueados.

`#include <erros.hpp>`

Inheritance diagram for ClienteBloqueado:



Collaboration diagram for ClienteBloqueado:



Public Member Functions

- virtual const char * what () const noexcept override

5.4.1 Detailed Description

Classe de exceção para clientes bloqueados.

Essa classe é usada para exibir uma mensagem de erro para quando um cliente bloqueado deseja alugar mais filmes.

5.4.2 Member Function Documentation

5.4.2.1 what()

```
virtual const char* ClienteBloqueado::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

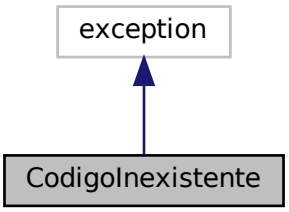
- errors/erros.hpp

5.5 CodigoInexistente Class Reference

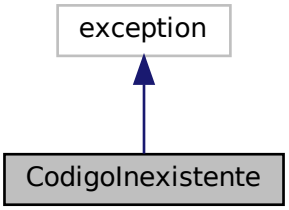
Classe de exceção para códigos inexistentes.

```
#include <erros.hpp>
```

Inheritance diagram for CodigoInexistente:



Collaboration diagram for CodigoInexistente:



Public Member Functions

- virtual const char * what () const noexcept override

5.5.1 Detailed Description

Classe de exceção para códigos inexistentes.

Essa classe é usada para exibir uma mensagem de erro para quando códigos inexistentes são digitados pelo usuário.

5.5.2 Member Function Documentation

5.5.2.1 what()

```
virtual const char* CodigoInexistente::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

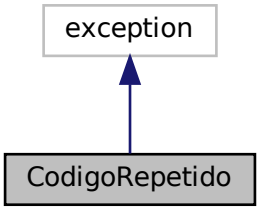
- errors/erros.hpp

5.6 CodigoRepetido Class Reference

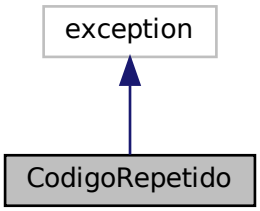
Classe de exceção para códigos repetidos.

```
#include <erros.hpp>
```

Inheritance diagram for CodigoRepetido:



Collaboration diagram for CodigoRepetido:



Public Member Functions

- virtual const char * what () const noexcept override

5.6.1 Detailed Description

Classe de exceção para códigos repetidos.

Essa classe é usada para exibir uma mensagem de erro para quando códigos repetidos são digitados pelo usuário.

5.6.2 Member Function Documentation

5.6.2.1 what()

```
virtual const char* CodigoRepetido::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

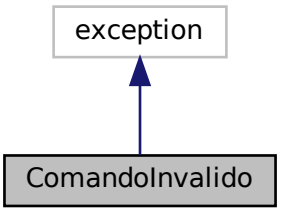
- errors/erros.hpp

5.7 ComandoInvalido Class Reference

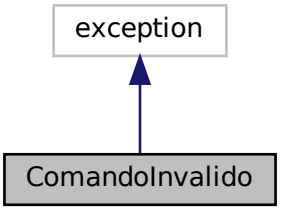
Classe de exceção para comandos inválidos.

`#include <erros.hpp>`

Inheritance diagram for ComandoInvalido:



Collaboration diagram for ComandoInvalido:



Public Member Functions

- virtual const char * what () const noexcept override

5.7.1 Detailed Description

Classe de exceção para comandos inválidos.

Essa classe é usada para exibir uma mensagem de erro para quando um comando inválido é digitado pelo usuário.

5.7.2 Member Function Documentation

5.7.2.1 what()

```
virtual const char* ComandoInvalido::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

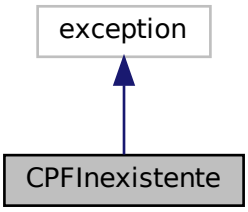
- errors/erros.hpp

5.8 CPFInexistente Class Reference

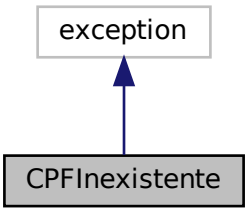
Classe de exceção para CPF's inexistentes.

```
#include <erros.hpp>
```

Inheritance diagram for CPFInexistente:



Collaboration diagram for CPFInexistente:



Public Member Functions

- virtual const char * what () const noexcept override

5.8.1 Detailed Description

Classe de exceção para CPF's inexistentes.

Essa classe é usada para exibir uma mensagem de erro para quando um CPF inexistente é digitado pelo usuário.

5.8.2 Member Function Documentation

5.8.2.1 what()

```
virtual const char* CPFInexistente::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

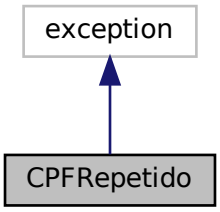
- errors/erros.hpp

5.9 CPFRepetido Class Reference

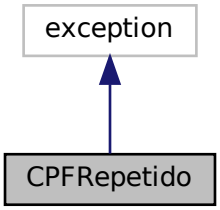
Classe de exceção para CPF's repetidos.

```
#include <erros.hpp>
```

Inheritance diagram for CPFRepetido:



Collaboration diagram for CPFRepetido:



Public Member Functions

- virtual const char * what () const noexcept override

5.9.1 Detailed Description

Classe de exceção para CPF's repetidos.

Essa classe é usada para exibir uma mensagem de erro para quando um CPF repetido é digitados pelo usuário.

5.9.2 Member Function Documentation

5.9.2.1 what()

```
virtual const char* CPFRepetido::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

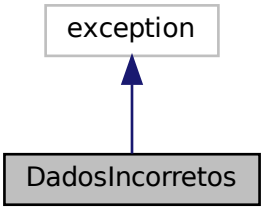
- errors/erros.hpp

5.10 DadosIncorretos Class Reference

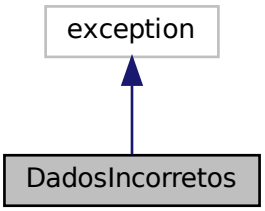
Classe de exceção para dados incorretos.

```
#include <erros.hpp>
```

Inheritance diagram for DadosIncorretos:



Collaboration diagram for DadosIncorretos:



Public Member Functions

- virtual const char ∗ what () const noexcept override

5.10.1 Detailed Description

Classe de exceção para dados incorretos.

Essa classe é usada para exibir uma mensagem de erro para quando dados incorretos são digitados pelo usuário.

5.10.2 Member Function Documentation

5.10.2.1 what()

```
virtual const char* DadosIncorretos::what ( ) const  [inline], [override], [virtual], [noexcept]
```

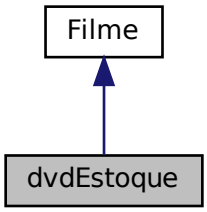
The documentation for this class was generated from the following file:

- errors/erros.hpp

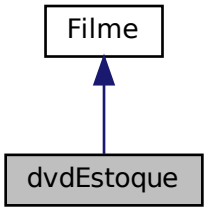
5.11 dvdEstoque Class Reference

`#include <DVDestoque.hpp>`

Inheritance diagram for dvdEstoque:



Collaboration diagram for dvdEstoque:



Public Member Functions

- dvdEstoque (int id, string titulo, int unidades)

  Construtor do arquivo "DVDestoque.hpp", para identificação e controle dos filmes da locadora.
- int calcularValor (int dias) override

  Método "calcularValor" da classe "dvdEstoque", utilizado especialmente para controle e registro.

Additional Inherited Members

5.11.1 Detailed Description

Na classe "dvdEstoque", que herda publicamente a classe "Filme", existem como atributos públicos um construtor, o qual recebe como parâmetros o id de identificação de um filme, seu título e a quantidade de unidades disponíveis dele para locação ("dvdEstoque(int id, string titulo, int unidades)"). Existe também um método para calcular quanto será pago pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado, o qual substitui uma função virtual na classe base, no caso, "Filme" ("int calcularValor(int dias) override").

Parameters

id	Determina o id do filme para identificação.
titulo	Armazena o nome do filme.
unidades	Registra quantas unidades de um filme estão disponíveis para locação.
dias	Registra quantos dias um determinado filme ficou alugado.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 dvdEstoque()

```
dvdEstoque::dvdEstoque (
            int id,
            string titulo,
            int unidades )
```

Construtor do arquivo "DVDestoque.hpp", para identificação e controle dos filmes da locadora.

Esse construtor inclui uma chamada para o construtor da classe base ("Filme"), o qual contém os mesmos parâmetros para identificação do filme ("(int id, string titulo, int unidades)"). Além disso, define o valor de "tipo" como "DVD".

Parameters

tipo	Armazena o tipo do filme.

5.11.3 Member Function Documentation

5.11.3.1 calcularValor()

```
int dvdEstoque::calcularValor (
            int dias )  [override], [virtual]
```

Método "calcularValor" da classe "dvdEstoque", utilizado especialmente para controle e registro.

Esse método calcula quanto o cliente terá de pagar pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado.

Parameters

dias	Armazena quantos dias um determinado filme ficou alugado.

Returns

Retorna o preço que o cliente terá de pagar pelo aluguel do filme, sendo 10 reais por dia o preço da locação de um DVD da categoria "Estoque".

Implements Filme.

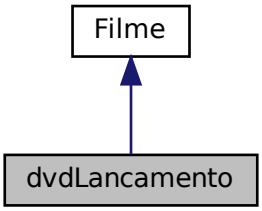
The documentation for this class was generated from the following files:

- Filmes/Subclasses/DVDestoque.hpp
- Filmes/Subclasses/DVDestoque.cpp

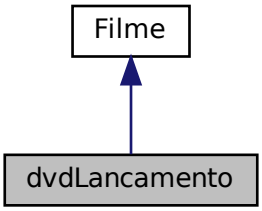
5.12 dvdLancamento Class Reference

`#include <DVDlancamento.hpp>`

Inheritance diagram for dvdLancamento:



Collaboration diagram for dvdLancamento:



Public Member Functions

- dvdLancamento (int id, string titulo, int unidades)

  Construtor do arquivo "DVDlancamento.hpp", para identificação e controle dos filmes da locadora.
- int calcularValor (int dias) override

  Método "calcularValor" da classe "dvdLancamento", utilizado especialmente para controle e registro.

Additional Inherited Members

5.12.1 Detailed Description

Na classe "dvdLancamento", que herda publicamente a classe "Filme", existem como atributos públicos um construtor, o qual recebe como parâmetros o id de identificação de um filme, seu título e a quantidade de unidades disponíveis dele para locação ("dvdLancamento(int id, string titulo, int unidades)"). Existe também um método para calcular quanto será pago pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado, o qual substitui uma função virtual na classe base, no caso, "Filme" ("int calcularValor(int dias) override").

Parameters

id	Determina o id do filme para identificação.
titulo	Armazena o nome do filme.
unidades	Registra quantas unidades de um filme estão disponíveis para locação.
dias	Registra quantos dias um determinado filme ficou alugado.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 dvdLancamento()

```
dvdLancamento::dvdLancamento (
            int id,
            string titulo,
            int unidades )
```

Construtor do arquivo "DVDlancamento.hpp", para identificação e controle dos filmes da locadora.

Esse construtor inclui uma chamada para o construtor da classe base ("Filme"), o qual contém os mesmos parâmetros para identificação do filme ("(int id, string titulo, int unidades)"). Além disso, define o valor de "tipo" como "DVD".

Parameters

tipo	Armazena o tipo do filme.

5.12.3 Member Function Documentation

5.12.3.1 calcularValor()

```
int dvdLancamento::calcularValor (
            int dias )  [override], [virtual]
```

Método "calcularValor" da classe "dvdLancamento", utilizado especialmente para controle e registro.

Esse método calcula quanto o cliente terá de pagar pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado.

Parameters

dias	Armazena quantos dias um determinado filme ficou alugado.

Returns

Retorna o preço que o cliente terá de pagar pelo aluguel do filme, sendo 20 reais por dia o preço da locação de um DVD da categoria "Lançamento".

Implements Filme.

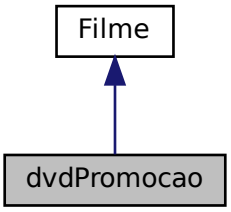
The documentation for this class was generated from the following files:

- Filmes/Subclasses/DVDlancamento.hpp
- Filmes/Subclasses/DVDlancamento.cpp

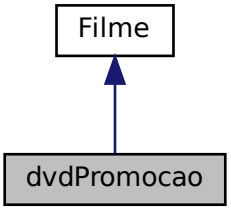
5.13 dvdPromocao Class Reference

`#include <DVDpromocao.hpp>`

Inheritance diagram for dvdPromocao:



Collaboration diagram for dvdPromocao:



Public Member Functions

- dvdPromocao (int id, string titulo, int unidades)

  Construtor do arquivo "DVDpromocao.hpp", para identificação e controle dos filmes da locadora.
- int calcularValor (int dias) override

  Método "calcularValor" da classe "dvdPromocao", utilizado especialmente para controle e registro.

Additional Inherited Members

5.13.1 Detailed Description

Na classe "dvdPromocao", que herda publicamente a classe "Filme", existem como atributos públicos um construtor, o qual recebe como parâmetros o id de identificação de um filme, seu título e a quantidade de unidades disponíveis dele para locação ("dvdPromocao(int id, string titulo, int unidades)"). Existe também um método para calcular quanto será pago pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado, o qual substitui uma função virtual na classe base, no caso, "Filme" ("int calcularValor(int dias) override").

Parameters

id	Determina o id do filme para identificação.
titulo	Armazena o nome do filme.
unidades	Registra quantas unidades de um filme estão disponíveis para locação.
dias	Registra quantos dias um determinado filme ficou alugado.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 dvdPromocao()

```
dvdPromocao::dvdPromocao (
            int id,
            string titulo,
            int unidades )
```

Construtor do arquivo "DVDpromocao.hpp", para identificação e controle dos filmes da locadora.

Esse construtor inclui uma chamada para o construtor da classe base ("Filme"), o qual contém os mesmos parâmetros para identificação do filme ("(int id, string titulo, int unidades)"). Além disso, define o valor de "tipo" como "DVD".

Parameters

tipo	Armazena o tipo do filme.

5.13.3 Member Function Documentation

5.13.3.1 calcularValor()

```
int dvdPromocao::calcularValor (
            int dias )  [override], [virtual]
```

Método "calcularValor" da classe "dvdPromocao", utilizado especialmente para controle e registro.

Esse método calcula quanto o cliente terá de pagar pelo aluguel do filme de acordo com a quantidade de dias que ele ficou alugado. Como o DVD está é da categoria "promoção", seu valor é fixo em 10 reais.

Parameters

dias	Armazena quantos dias um determinado filme ficou alugado.

Returns

Retorna o preço que o cliente terá de pagar pelo aluguel do filme, sendo 10 reais fixos o preço da locação de um DVD da categoria "Promoção".

Implements Filme.

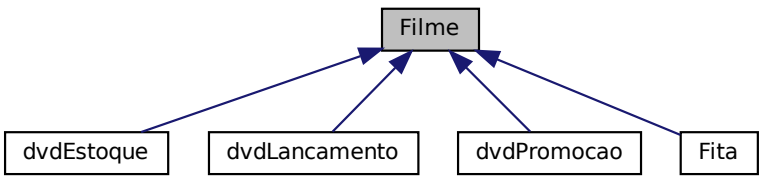
The documentation for this class was generated from the following files:

- Filmes/Subclasses/DVDpromocao.hpp
- Filmes/Subclasses/DVDpromocao.cpp

5.14 Filme Class Reference

```
#include <filme.hpp>
```

Inheritance diagram for Filme:



Public Member Functions

- Filme (int id, string titulo, int unidades)

  Construtor do arquivo "filme.hpp" para identificação e controle dos filmes da locadora.
- virtual ∼Filme ()=default
- virtual int calcularValor (int dias)=0
- void lerFilme ()

  Método "lerFilme" da classe "Filme", utilizado especialmente para exibição.
- virtual bool serAlugado ()

  Método "serAlugado" da classe "Filme", utilizado especialmente para controle.
- void serDevolvido ()

  Método "serDevolvido" da classe "Filme", utilizado especialmente para controle.
- void serAvaliado (float nota)

  Método "serAvaliado" da classe "Filme", utilizado para avaliação dos filmes.

Public Attributes

- int unidades
- float avaliacao
- bool erro
- int id

Protected Attributes

- int vezesAvaliado
- string titulo
- string tipo

Friends

- class Locadora
- class Cliente

5.14.1 Detailed Description

Na classe "Filme", a qual possui como classe amiga "Locadora" e "Cliente", existem como atributos protegidos quantas vezes um determinado filme foi avaliado ("int vezesAvaliado"), o nome do filme ("string titulo") e o tipo desse filme, seja fita, DVD, etc ("string tipo"). A classe também contém como atributos públicos um construtor que recebe o id, o nome e a quantidade disponível de determinado filme ("Filme(int id, string titulo, int unidades)") e um destrutor virtual, inicializado como default ("∼Filme()"). Existe também um método para calcular quanto será pago pelo aluguel do filme, de acordo com o tempo de aluguel ("virtual int calcularValor(int dias) = 0"), um método para identificar os filmes ("void lerFilme()"), um método para alugar os filmes disponíveis ("virtual void serAlugado()"), um método para devolver os filmes que foram alugados ("void serDevolvido()") e um método para avaliar o filme alugado ("void serAvaliado(int nota)"). Além disso, contém um atributo público para controlar a quantidade de filmes disponíveis na locadora ("int unidades"), outro atributo para registrar a nota de avaliação recebida pelo filme ("float avaliação") e, por fim, um atributo que para registrar o id de identificação de um filme ("int id").

Parameters

id	Determina o id do filme para identificação.
avaliacao	Registra a avaliação recebida pelo filme.
vezesAvaliado	Registra quantas vezes um filme foi avaliado.
titulo	Armazena o nome do filme.
tipo	Armazena o tipo do filme.
unidades	Registra quantas unidades de um filme estão disponíveis para locação

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Filme()

```
Filme::Filme (
            int id,
            string titulo,
            int unidades )
```

Construtor do arquivo "filme.hpp" para identificação e controle dos filmes da locadora.

Esse construtor registra o id, o nome e a quantidade de filmes disponíveis para locação, auxiliando a locadora no controle. Além disso, registra as avaliações (inicializada em 4) e quantas vezes determinado filme foi avaliado (inicializada em 1).

Parameters

id	Identifica o filme.
titulo	Armzena o nome do filme.
unidades	Registra quantas unidades de um filme estão disponíveis para locação.
vezesAvaliado	Registra quantas vezes um filme foi avaliado.
avaliacao	Regista a nota de avaliação recebida por um filme.

5.14.2.2 ∼Filme()

```
virtual Filme::∼Filme ( )  [virtual], [default]
```

5.14.3 Member Function Documentation

5.14.3.1 calcularValor()

```
virtual int Filme::calcularValor (
            int dias )  [pure virtual]
```

Implemented in Fita, dvdEstoque, dvdLancamento, and dvdPromocao.

5.14.3.2 lerFilme()

```
void Filme::lerFilme ( )
```

Método "lerFilme" da classe "Filme", utilizado especialmente para exibição.

Esse método exibe para o usuário o id do filme, seu título, a quantidade de unidades restantes para locação na locadora e o tipo do filme.

Parameters

tipo	Armazena o tipo do filme.

5.14.3.3 serAlugado()

```
bool Filme::serAlugado ( )  [virtual]
```

Método "serAlugado" da classe "Filme", utilizado especialmente para controle.

Esse método controla a quantidade de exemplares de um determinado filme que está na locadora para ser alugado. Caso não existam mais unidades para serem alugadas, o método exibe uma mensagem de erro, informando sobre isso.

Reimplemented in Fita.

5.14.3.4 serAvaliado()

```
void Filme::serAvaliado (
            float nota )
```

Método "serAvaliado" da classe "Filme", utilizado para avaliação dos filmes.

Esse método registra as avalizações dos filmes e também calcula quantas vezes esse filme foi avaliado com uma determinada nota. Além disso, atribui uma média dessas avaliações para esse mesmo filme.

Parameters

nota	Armazena uma nota recebida pelo filme.
soma	Armazena a soma das notas recebidas pelo filme.

5.14.3.5 serDevolvido()

```
void Filme::serDevolvido ( )
```

Método "serDevolvido" da classe "Filme", utilizado especialmente para controle.

Esse método controla as unidades dos filmes da locadora conforme são devolvidos pelos clientes pós aluguel.

5.14.4 Friends And Related Function Documentation

5.14.4.1 Cliente

```
friend class Cliente  [friend]
```

5.14.4.2 Locadora

```
friend class Locadora  [friend]
```

5.14.5 Member Data Documentation

5.14.5.1 avaliacao

```
float Filme::avaliacao
```

5.14.5.2 erro

```
bool Filme::erro
```

5.14.5.3 id

```
int Filme::id
```

5.14.5.4 tipo

```
string Filme::tipo  [protected]
```

5.14.5.5 titulo

```
string Filme::titulo  [protected]
```

5.14.5.6 unidades

```
int Filme::unidades
```

5.14.5.7 vezesAvaliado

```
int Filme::vezesAvaliado  [protected]
```

The documentation for this class was generated from the following files:

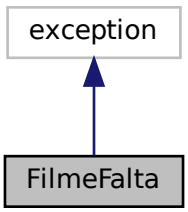
- Filmes/filme.hpp
- Filmes/filme.cpp

5.15 FilmeFalta Class Reference

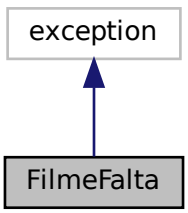
Classe de exceção para falta de filmes no estoque.

```
#include <erros.hpp>
```

Inheritance diagram for FilmeFalta:



Collaboration diagram for FilmeFalta:



Public Member Functions

- FilmeFalta (int id)
- virtual const char * what () const noexcept override

5.15.1 Detailed Description

Classe de exceção para falta de filmes no estoque.

Essa classe é usada para exibir uma mensagem de erro para quando um id de um filme que está em falta no estoque é digitado pelo usuário.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 FilmeFalta()

```
FilmeFalta::FilmeFalta (
            int id )  [inline]
```

5.15.3 Member Function Documentation

5.15.3.1 what()

```
virtual const char* FilmeFalta::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

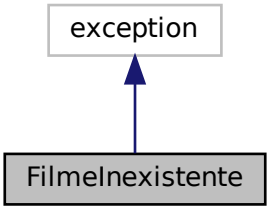
- errors/erros.hpp

5.16 FilmeInexistente Class Reference

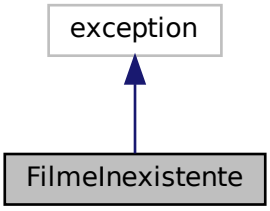
Classe de exceção para filmes inexistentes.

`#include <erros.hpp>`

Inheritance diagram for FilmeInexistente:



Collaboration diagram for FilmeInexistente:



Public Member Functions

- FilmeInexistente (int id)
- virtual const char * what () const noexcept override

5.16.1 Detailed Description

Classe de exceção para filmes inexistentes.

Essa classe é usada para exibir uma mensagem de erro para quando um id de um filme inexistente é digitado pelo usuário.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 FilmeInexistente()

```
FilmeInexistente::FilmeInexistente (
            int id )  [inline]
```

5.16.3 Member Function Documentation

5.16.3.1 what()

```
virtual const char* FilmeInexistente::what ( ) const  [inline], [override], [virtual], [noexcept]
```

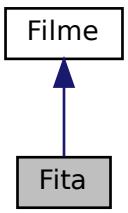
The documentation for this class was generated from the following file:

- errors/erros.hpp

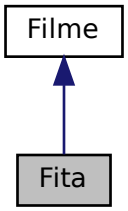
5.17 Fita Class Reference

```
#include <fita.hpp>
```

Inheritance diagram for Fita:



Collaboration diagram for Fita:



Public Member Functions

- Fita (int id, string titulo, int unidades)

  Construtor do arquivo "fita.hpp", para identificação e controle dos filmes da locadora.
- int calcularValor (int dias) override
- bool serAlugado () override

  Método "serAlugado" da classe "Filme", utilizado especialmente para controle.

Additional Inherited Members

5.17.1 Detailed Description

Na classe "Fita", que herda publicamente a classe "Filme", existe como atributo privado a variável booleana "rebobinada", que determina se a fita foi rebobinada ou não. A classe também contém como atributos públicos um construtor, o qual recebe como parâmetros um id de identificação de uma fita, seu título e a quantidade de unidades dela disponíveis para locação ("Fita(int id, string titulo, int unidades)"), um método para calcular quanto será pago pelo aluguel da fita de acordo com a quantidade de dias que ela ficou alugada ("("int calcularValor(int dias) override")") e um método para alugar as fitas disponíveis na locadora ("void serAlugado() override"). Ambos métodos substituem funções virtuais na classe base, no caso,"Filme".

Parameters

rebobinada	Determina se a fita foi rebobinada ou não.
id	Determina o id do filme (no caso, fita) para identificação.
titulo	Armazena o nome do filme (no caso, fita).
unidades	Registra quantas unidades de um filme (no caso, fita) estão disponíveis para locação.
dias	Registra quantos dias um determinado filme (no caso, fita) ficou alugado.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 Fita()

```
Fita::Fita (
            int id,
            string titulo,
            int unidades )
```

Construtor do arquivo "fita.hpp", para identificação e controle dos filmes da locadora.

Esse construtor inclui uma chamada para o construtor da classe base ("Filme"), o qual contém os mesmos parâmetros para identificação da fita ("(int id, string titulo, int unidades)"). Além disso, define o valor de "tipo" como "FITA" e o valor de "rebobinada" como "true".

Parameters

tipo	Armazena o tipo do filme (no caso, filme).
rebobinada	Determina se a fita foi rebobinada ou não.

5.17.3 Member Function Documentation

5.17.3.1 calcularValor()

```
int Fita::calcularValor (
            int dias )  [override], [virtual]
```

Implements Filme.

5.17.3.2 serAlugado()

```
bool Fita::serAlugado ( )  [override], [virtual]
```

Método "serAlugado" da classe "Filme", utilizado especialmente para controle.

Esse método determina que, sempre que uma fita for alugada, ela será automaticamente considerada desrebobinada.

Reimplemented from Filme.

The documentation for this class was generated from the following files:

- Filmes/Subclasses/fita.hpp
- Filmes/Subclasses/fita.cpp

5.18 Locadora Class Reference

`#include <locadora.hpp>`

Public Member Functions

- Locadora ()

  Construtor do arquivo "locadora.hpp" para armazenagem dos clientes e filmes da locadora.
- ~Locadora ()

  Destrutor do arquivo "locadora.hpp".
- void cadastrarFilme (char tipo, int quantidade, int id, const string &titulo, const char &categoria)

  Método "cadastrarFilme" da classe "Locadora", utilizado especialmente para controle e registro.
- void removerFilme (int codigo)

  Método "removerFilme" da classe "Locadora", utilizado para registro e controle.
- void listarFilmes (char opcao)

  Método "listarFilmes" da classe "Locadora", utilizado especialmente para exibição.
- void cadastrarCliente (long long cpf, string nome)

  Método "cadastrarFilme" da classe "Locadora", utilizado para controle e registro.
- void removerCliente (long long cpf)

  Método "removerCliente" da classe "Locadora", utilizado para controle e registro.
- void listarClientes (char opcao)

  Método "listarClientes" da classe "Locadora", utilizado para exibição.
- void listarBloqueados ()

  Método "listarBloqueados" da classe "Locadora".
- void aluguel (long long cpf, vector< int > id)

  Método "aluguel" da classe "locadora", utilizado especialmente para controle e registro.
- void devolucao (long long cpf, int dias)

  Método "devolucao" da classe "Locadora", utilizado especialmente para controle e registro.
- void recomendarFilmes (long long cpf)

  Método "recomendarFilmes" da classe "Locadora".
- void avaliarFilme (int id, float nota)

  Método "avaliarFilme" da classe "Locadora".

Public Attributes

- vector< Filme * > filmes
- vector< Cliente * > clientes
- vector< Cliente * > bloqueados
- vector< Cliente * > clientesOrdenados
- vector< Filme * > filmesOrdenados
- bool erro

5.18.1 Detailed Description

Na classe "Locadora",existem como atributos públicos um construtor ("Locadora()") e um destrutor ("∼Locadora()"). Existe também um método para cadastrar filmes, que recebe como parâmetros o tipo do filme, a quantidade dele, seu id de identificação, seu título e sua e sua categoria, seja estoque, promoção ou lançamento ("void cadastrar↩ Filme(char tipo, int quantidade, int id, const string& titulo, const char& categoria)"), um método para remover filmes de acordo com seu código ("void removerFilme(int codigo)"), um método para listar filmes de de acordo com a opção selecionada, seja a partir do id de identificação ou título do filme ("void listarFilmes(char opcao)"), um método para cadastrar clientes, de acordo com o cpf e o nome informados ("void cadastrarCliente(long long cpf, string nome)"), um método para remover clientes utilizando somente o cpf informado ("void removerCliente(long long cpf)"), um método para listar clientes a partir da opção selecionada, seja por nome ou cpf cadastrados ("void listarClientes(char opcao)"), um método para exibir uma lista de sobre clientes bloqueados na locadora ("void listarBloqueados()"), um método para registrar os clientes que alugaram filmes e quais filmes foram alugados, com base em seus id's ("void aluguel(long long cpf, vector <int> id)"), um método para controlar quais clientes fazem a devolução dos filmes e com quantos dias esse cliente ficou com o filme alugado ("void devolucao(long long cpf, int dias)"), um método para recomendar filmes aos clientes, de acordo com o cpf informado ("void recomendar↩ Filmes(long long cpf)") e um método para avaliar os filmes, de acordo com seu id e a nota dada ("void avaliar↩ Filme(int id, int nota)"). Além disso, a classe conta um vetor de ponteiros para armazenar os filmes da locadora ("vector<Filme∗> filmes") e um vetor de ponteiros para armazenar os clientes da locadora ("vector<Cliente∗> clientes"). Por fim, existe um vetor de ponteiros que armazena uma lista de clientes ordenados ("vector<Cliente∗> clientesOrdenados") e outro vetor, também de ponteiros, que armazena uma lista de filmes ordenados ("vector<↩ Filme∗> filmesOrdenados").

Parameters

tipo	Armazena o tipo do filme (fita ou DVD).
quantidade	Armazena a quantidade de filmes que serão cadastrados na locadora.
id	Armazena o id de identificação do filme.
titulo	Armazena o título do filme.
categoria	Armazena a categoria do filme (estoque, promoção ou lançamento).
codigo	Armazena o código do filme (id).
opcao	Determina o tipo de listagem de filmes ou clientes a ser selecionada.
cpf	Armazena o cpf do cliente.
nome	Armazena o nome do cliente.
nota	Armazena a nota recebida por um filme.
filmes	Armazena todos os filmes da locadora.
clientes	Armazena todos os clientes da locadora.
erro	Informa caso haja erros no código.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Locadora()

```
Locadora::Locadora ( )
```

Construtor do arquivo "locadora.hpp" para armazenagem dos clientes e filmes da locadora.

Esse construtor utiliza os vetores "filmes" e "clientes" para armazenar os filmes e clientes da locadora, respectivavente. Depois atribui a esses vetores os membros de dados correspondentes da classe, no caso, atribui os membros "filmes" e "clientes" à eles.

Parameters

filmes	Armazena os filmes da locadora.
clientes	Armazena os clientes da locadora.

5.18.2.2 ∼Locadora()

```
Locadora::∼Locadora ( )
```

Destrutor do arquivo "locadora.hpp".

Esse destrutor é responsável pela liberação manual de mémoria alocada para os vetores "filmes" e "clientes" a partir de um loop de repetição "for". O destrutor percorre os vetores e para cado elemento deles, ele realiza um delete, liberando a memória anteriormente alocada.

Parameters

filme	Elemento do vetor "filmes".
cliente	Elemento do vetor "clientes".

5.18.3 Member Function Documentation

5.18.3.1 aluguel()

```
void Locadora::aluguel (
            long long cpf,
            vector< int > id )
```

Método "aluguel" da classe "locadora", utilizado especialmente para controle e registro.

O método recebe como parâmetros o cpf de um cliente que deseja alugar um filme e o id de identificação desse mesmo filme. A partir daí, ele percorre o vetor "clientes", procurando pelo cpf informado e o vetor "filmes", procurando pelo id do filme informado. Caso o cliente ou o filme não existirem, o método exibe uma mensagem de erro. Caso contrário, ocorre uma verificação para saber se ainda existem cópias do filme disponíveis para locação. Caso existam, o método registra que o filme foi alugado pelo cliente. Caso contrário, o método informa que não existem mais unidades do filme disponiveis para aluguel. Além disso, o método possibilita o aluguel de vários filmes ao mesmo tempo.

5.18.3.2 avaliarFilme()

```
void Locadora::avaliarFilme (
            int id,
            float nota )
```

Método "avaliarFilme" da classe "Locadora".

O método recebe como parâmetros o id de identificação de um filme e a nota dada à ele por um cliente. A partir daí, o método percorre o vetor "filmes" procurando pelo id informado. Caso o filme não seja encontrado, o método exibe uma mensagem de erro. Caso contrário, ocorre a chamada do método "serAvaliado()", o qual atribui a nota dada pelo cliente à esse filme em questão.

Parameters

nota	Armazena a nota recebida por um filme.

5.18.3.3 cadastrarCliente()

```
void Locadora::cadastrarCliente (
            long long cpf,
            string nome )
```

Método "cadastrarFilme" da classe "Locadora", utilizado para controle e registro.

Esse método cadastra um novo cliente à locadora. Ele recebe como parâmetros o nome e o cpf de um determinado cliente. A partir daí, ele percorre o vetor "clientes" da locadora e verifica se o cpf do cliente já está cadastrado. Caso esteja, o método exibe uma mensagem de erro informando sobre isso e retorna. Caso o cpf informado seja inválido, o método também exibe uma mensagem de erro e retorna. Por fim, caso o cpf ainda não esteja cadastrado na locadora e seja válido, o cliente é adicionado ao vetor "clientes" da locadora e, portanto, cadastrado na mesma, de forma que o método exibe uma mensagem informando que o cadastro foi um sucesso!.

Parameters

cpf	Armazena o cpf do cliente.
nome	Armazena o nome do cliente.

5.18.3.4 cadastrarFilme()

```
void Locadora::cadastrarFilme (
            char tipo,
            int unidades,
            int id,
            const string & titulo,
            const char & categoria )
```

Método "cadastrarFilme" da classe "Locadora", utilizado especialmente para controle e registro.

Esse método é responsável por adicionar um novo filme à locadora. Ele recebe como parâmetros o tipo do filme, a quantidade de unidades que serão registradas, o id de identificação do filme, seu título e também a categoria na qual ele pertence. A partir daí, o método verifica o id do filme, seu tipo, categoria e unidades. Caso o id do filme ainda não esteja registrado na locadora, o tipo e categoria do filme forem válidos e existirem unidades suficientes, o filme é registrado, caso contrário, não.

Parameters

tipo	Armazena o tipo do filme (DVD ou fita).
unidades	Armazena a quantidade de unidades do filme.
id	Armazena o id de identificação do filme.
titulo	Armazena o título do filme.
categoria	Armazena a categoria do filme.

5.18.3.5 devolucao()

```
void Locadora::devolucao (
            long long cpf,
            int dias )
```

Método "devolucao" da classe "Locadora", utilizado especialmente para controle e registro.

O método recebe como parâmetros o cpf de um cliente e os dias nos quais ele ficou com um filme alugado. A partir daí, a função procura pelo cpf informado no vetor "clientes". Caso esse cliente não exista, o método exibe uma mensagem de erro. Caso contrário, ocorre uma busca pela lista de filmes alugados por esse cliente e calcula quanto ele deve pagar com base nos dias que ele ficou com o filme alugado. Além disso, o método verifica se o cliente em questão possui mais de 10 pontos. Caso possua, ocorre uma aplicação de 10% de desconto sobre o valor do aluguel. Por fim, o método exibe o valor que o cliente terá que pagar e chama outros dois métodos: "ser↩ Devolvido()", para registrar que o filme foi devolvido à locadora e "devolver()", para registrar que o cliente devolveu o filme à locadora.

Parameters

dias	Armazena a quantidade de dias que um cliente ficou com o filme alugado.

5.18.3.6 listarBloqueados()

```
void Locadora::listarBloqueados ( )
```

Método "listarBloqueados" da classe "Locadora".

O método lista os clientes bloqueados da locadora pelo fato de ainda não terem realizado a devolução dos filmes alugados. caso o vetor "bloqueados" esteja vazio, uma mensagem informando que não há clientes bloqueados é exibida. Caso contrário, uma lista dos clientes bloqueados é exibida.

5.18.3.7 listarClientes()

```
void Locadora::listarClientes (
            char opcao )
```

Método "listarClientes" da classe "Locadora", utilizado para exibição.

Esse método lista os clientes cadastrados na locadora. Ele recebe o parâmetro "opção" e verifica se a opção de ordenação é valida. Se não for válida, o método exibe uma mensagem de erro. Caso o vetor "clientes" esteja vazio, significa que não há clientes cadastrado na locadora, então o método exibe uma mensagem avisando sobre isso. Caso a opção de ordenação seja válida e existirem clientes cadastrados na locadora, o método lista esses clientes de acordo com o cpf ou nome.

Parameters

opcao	Armazena a opção de listagem dos clientes (nome ou cpf).

5.18.3.8 listarFilmes()

```
void Locadora::listarFilmes (
            char opcao )
```

Método "listarFilmes" da classe "Locadora", utilizado especialmente para exibição.

Esse método é responsável por listar os filmes da locadora. Ele recebe como parâmetros a opção de listagem desejada pelo usuário. Caso essa opção seja inválida, o método exibe uma mensagem de erro e retorna. O mesmo ocorre caso ainda não existam filmes para listar. Caso a opção de listagem seja válida e existam filmes para serem listados, o método cria umma cópia do vetor "filmes" da locadora e ordena com base na opção fornecida, seja por id ou título.

Parameters

opcao	Armazena a opção de ordenação dos filmes.

5.18.3.9 recomendarFilmes()

```
void Locadora::recomendarFilmes (
            long long cpf )
```

Método "recomendarFilmes" da classe "Locadora".

O método recebe como parâmetro um cpf e procura no vetor "clientes" pelo cliente com o cpf informado. Se o cliente não for encontrado, uma mensagem de erro é exibida. Caso contrário, ocorre a chamada do método "recomendar()" para o cliente em questão, o qual recomenda filmes com base nos gostos desse cliente e nos gostos de outros clientes da locadora que são considerados similares à ele. Por fim, a função percorre a lista de filmes recomendados e chama o método "lerFilme()", o qual exibe informações sobre os filmes recomendados na tela.

5.18.3.10 removerCliente()

```
void Locadora::removerCliente (
            long long cpf )
```

Método "removerCliente" da classe "Locadora", utilizado para controle e registro.

Esse método remove um cliente já cadastrado na locadora. Ele recebe como parâmetro o cpf de um cliente, o qual indica o cpf do cliente a ser removido. A partir daí, o método percorre o vetor "clientes" da locadora à procura do cliente com o cpf informado, utilizando a função "find_if". Caso esse cliente seja encontrado, ele é removido do vetor e, portanto, da locadora, de forma que o método exibe uma mensagem informando sobre isso. Caso contrário, uma mensagem de erro é exibida.

5.18.3.11 removerFilme()

```
void Locadora::removerFilme (
            int id )
```

Método "removerFilme" da classe "Locadora", utilizado para registro e controle.

Esse método remove um filme da locadora. Ele recebe como parâmetro o id de identificação do filme e, a partir daí, percorre o vetor "filmes" procurando o filme com esse id, utilizando a funcão "find_if". Caso esse filme seja encontrado, o método o retira do vetor "filmes" e, portanto, da locadora. Caso contrário, o método exibe uma mensagem de erro e não retira nenhum filme da locadora.

5.18.4 Member Data Documentation

5.18.4.1 bloqueados

```
vector<Cliente*> Locadora::bloqueados
```

5.18.4.2 clientes

```
vector<Cliente*> Locadora::clientes
```

5.18.4.3 clientesOrdenados

```
vector<Cliente*> Locadora::clientesOrdenados
```

5.18.4.4 erro

```
bool Locadora::erro
```

5.18.4.5 filmes

```
vector<Filme*> Locadora::filmes
```

5.18.4.6 filmesOrdenados

```
vector<Filme*> Locadora::filmesOrdenados
```

The documentation for this class was generated from the following files:

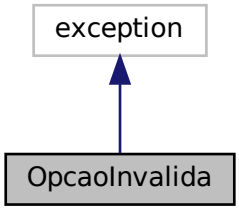
- Locadora/locadora.hpp
- Locadora/locadora.cpp

5.19 OpcaoInvalida Class Reference

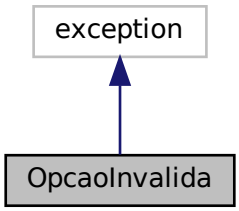
Classe de exceção para opções inválidas.

```
#include <erros.hpp>
```

Inheritance diagram for OpcaoInvalida:



Collaboration diagram for OpcaoInvalida:



Public Member Functions

- virtual const char * what () const noexcept override

5.19.1 Detailed Description

Classe de exceção para opções inválidas.

Essa classe é usada para exibir uma mensagem de erro para quando uma opção inválido é digitado pelo usuário.

5.19.2 Member Function Documentation

5.19.2.1 what()

```
virtual const char* OpcaoInvalida::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

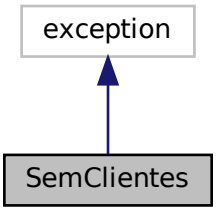
- errors/erros.hpp

5.20 SemClientes Class Reference

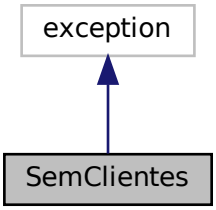
Classe de exceção para falta de clientes.

`#include <erros.hpp>`

Inheritance diagram for SemClientes:



Collaboration diagram for SemClientes:



Public Member Functions

- virtual const char * what () const noexcept override

5.20.1 Detailed Description

Classe de exceção para falta de clientes.

Essa classe é usada para exibir uma mensagem de erro para quando a locadora não possui clientes cadastrados.

5.20.2 Member Function Documentation

5.20.2.1 what()

```
virtual const char* SemClientes::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

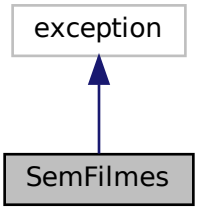
- errors/erros.hpp

5.21 SemFilmes Class Reference

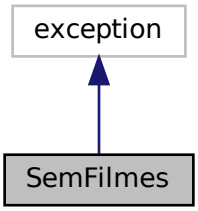
Classe de exceção para filmes inexistentes.

```
#include <erros.hpp>
```

Inheritance diagram for SemFilmes:



Collaboration diagram for SemFilmes:



Public Member Functions

- virtual const char * what () const noexcept override

5.21.1 Detailed Description

Classe de exceção para filmes inexistentes.

Essa classe é usada para exibir uma mensagem de erro para quando a locadora ainda não possui filmes cadastrados.

5.21.2 Member Function Documentation

5.21.2.1 what()

```
virtual const char* SemFilmes::what ( ) const  [inline], [override], [virtual], [noexcept]
```

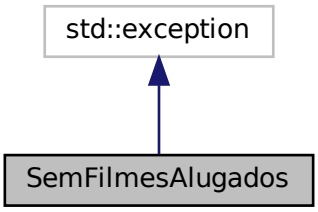
The documentation for this class was generated from the following file:

- errors/erros.hpp

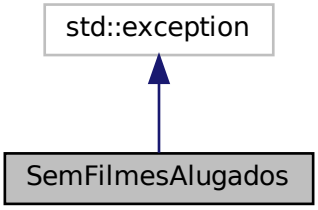
5.22 SemFilmesAlugados Class Reference

`#include <erros.hpp>`

Inheritance diagram for SemFilmesAlugados:



Collaboration diagram for SemFilmesAlugados:



Public Member Functions

- virtual const char * what () const noexcept override

5.22.1 Member Function Documentation

5.22.1.1 what()

`virtual const char* SemFilmesAlugados::what ( ) const [inline], [override], [virtual], [noexcept]`

The documentation for this class was generated from the following file:

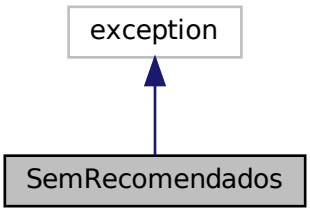
- errors/erros.hpp

5.23 SemRecomendados Class Reference

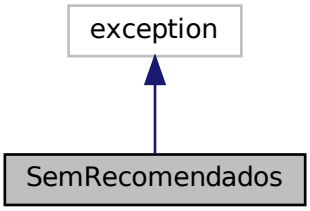
Classe de exceção para clientes sem filmes recomendados.

`#include <erros.hpp>`

Inheritance diagram for SemRecomendados:



Collaboration diagram for SemRecomendados:



Public Member Functions

- virtual const char * what () const noexcept override

5.23.1 Detailed Description

Classe de exceção para clientes sem filmes recomendados.

Essa classe é usada para exibir uma mensagem de erro para quando um cliente ainda não possui filmes recomendados.

5.23.2 Member Function Documentation

5.23.2.1 what()

```
virtual const char* SemRecomendados::what ( ) const  [inline], [override], [virtual], [noexcept]
```

The documentation for this class was generated from the following file:

- errors/erros.hpp

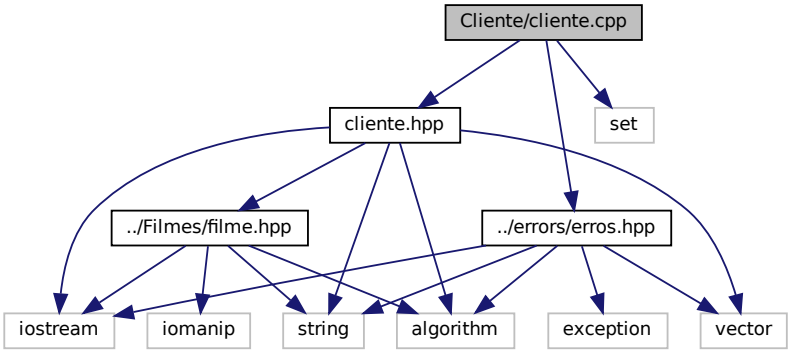
Chapter 6

File Documentation

6.1 Cliente/cliente.cpp File Reference

```
#include "cliente.hpp"
#include "../errors/erros.hpp"
#include <set>
```

Include dependency graph for cliente.cpp:

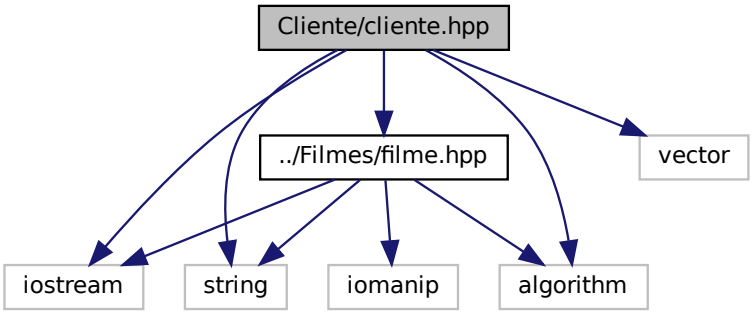


6.2 Cliente/cliente.hpp File Reference

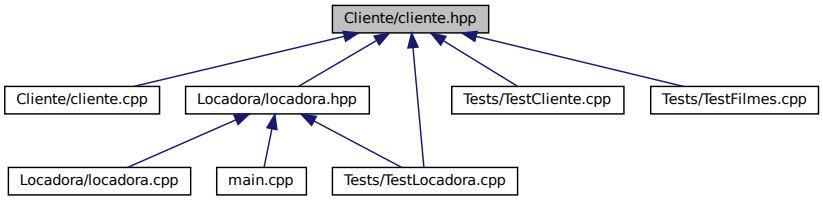
Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "cliente.cpp".

```
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
```

```
#include "../Filmes/filme.hpp"
```
Include dependency graph for cliente.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Cliente

6.2.1 Detailed Description

Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "cliente.cpp".

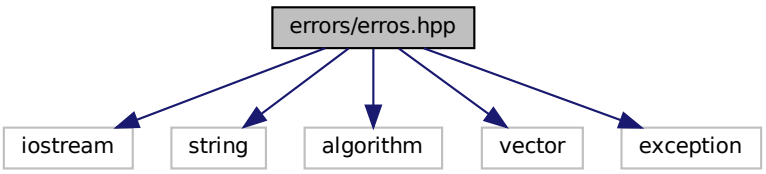
Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "cliente.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.3 errors/erros.hpp File Reference

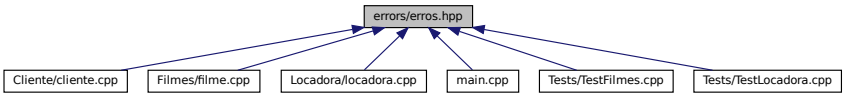
```
#include <iostream>
#include <string>
#include <algorithm>
```

```
#include <vector>
#include <exception>
```

Include dependency graph for erros.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class ComandoInvalido

  Classe de exceção para comandos inválidos.
- class OpcaoInvalida

  Classe de exceção para opções inválidas.
- class DadosIncorretos

  Classe de exceção para dados incorretos.
- class CodigoRepetido

  Classe de exceção para códigos repetidos.
- class CodigoInexistente

  Classe de exceção para códigos inexistentes.
- class CPFRepetido

  Classe de exceção para CPF's repetidos.
- class CPFInexistente

  Classe de exceção para CPF's inexistentes.
- class ArquivoInexistente

  Classe de exceção para arquivos inexistentes.
- class SemFilmesAlugados
- class SemFilmes

  Classe de exceção para filmes inexistentes.
- class FilmeInexistente

  Classe de exceção para filmes inexistentes.
- class FilmeFalta

  Classe de exceção para falta de filmes no estoque.
- class SemClientes

Classe de exceção para falta de clientes.

- class ClienteBloqueado

  Classe de exceção para clientes bloqueados.

- class AvaliacaoErrada

  Classe de exceção para avaliações erradas.

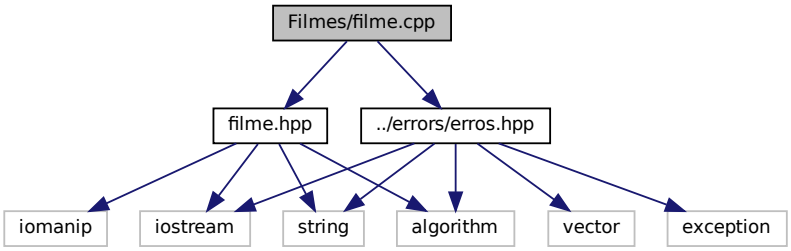
- class SemRecomendados

  Classe de exceção para clientes sem filmes recomendados.

6.4 Filmes/filme.cpp File Reference

```
#include "filme.hpp"
#include "../errors/erros.hpp"
```

Include dependency graph for filme.cpp:

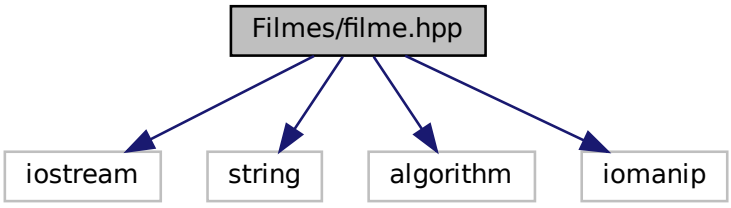


6.5 Filmes/filme.hpp File Reference

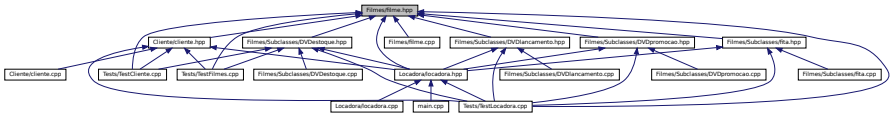
Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "filme.cpp".

```
#include <iostream>
#include <string>
#include <algorithm>
#include <iomanip>
```

Include dependency graph for filme.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Filme

6.5.1 Detailed Description

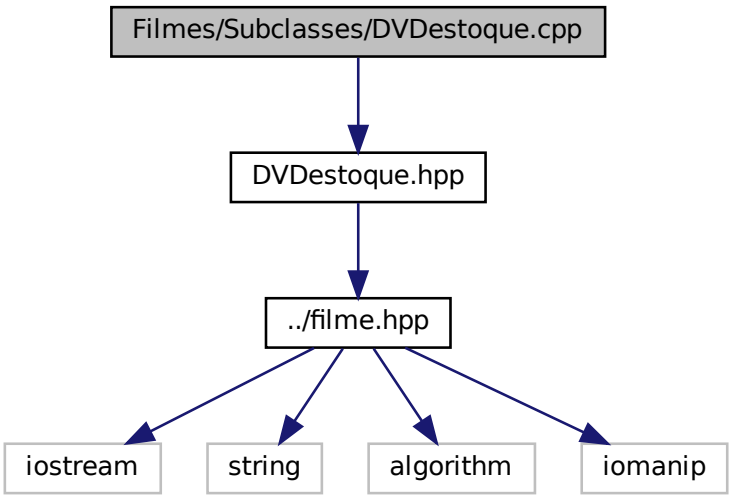
Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "filme.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "filme.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.6 Filmes/Subclasses/DVDestoque.cpp File Reference

```
#include "DVDestoque.hpp"
```

Include dependency graph for DVDestoque.cpp:

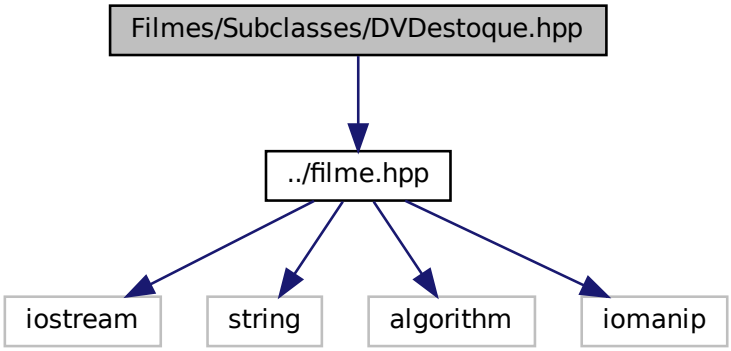


6.7 Filmes/Subclasses/DVDestoque.hpp File Reference

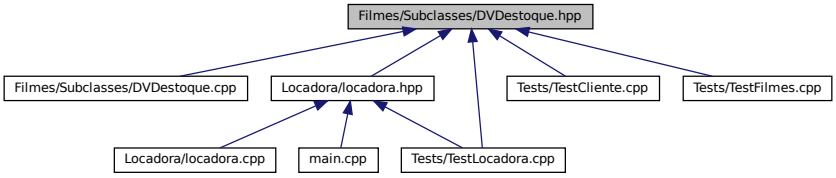
Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "DV↩ Destoque.cpp".

```
#include "../filme.hpp"
```

Include dependency graph for DVDestoque.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class dvdEstoque

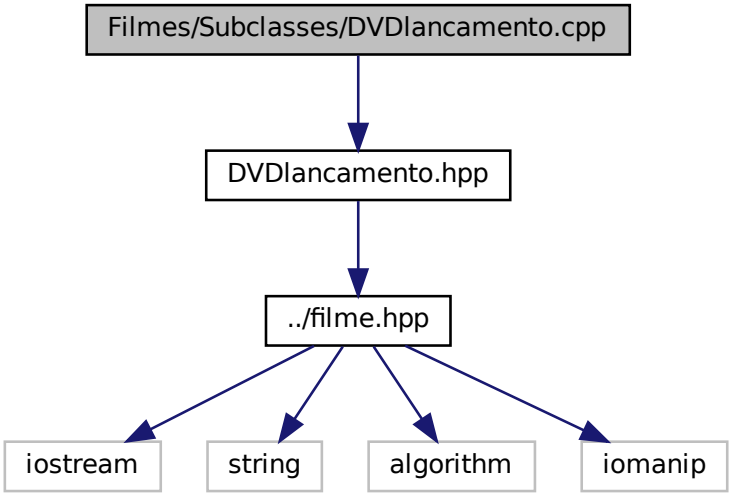
6.7.1 Detailed Description

Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "DV↩ Destoque.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "DVDestoque.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.8 Filmes/Subclasses/DVDlancamento.cpp File Reference

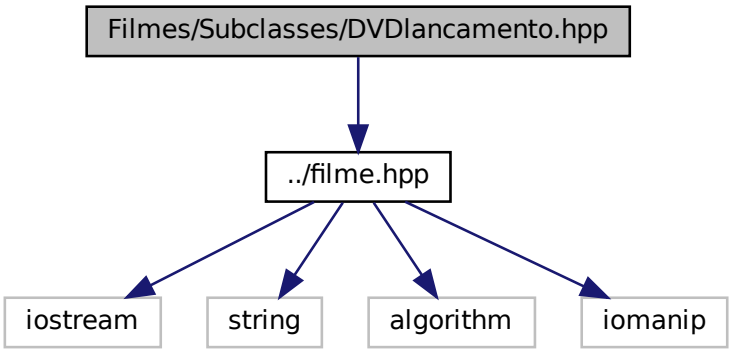
`#include "DVDlancamento.hpp"`
Include dependency graph for DVDlancamento.cpp:



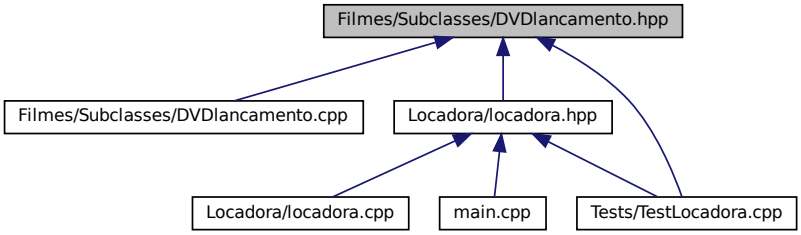
6.9 Filmes/Subclasses/DVDlancamento.hpp File Reference

Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "DV↩ Dlancamemto.cpp".

`#include "../filme.hpp"`
Include dependency graph for DVDlancamento.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class dvdLancamento

6.9.1 Detailed Description

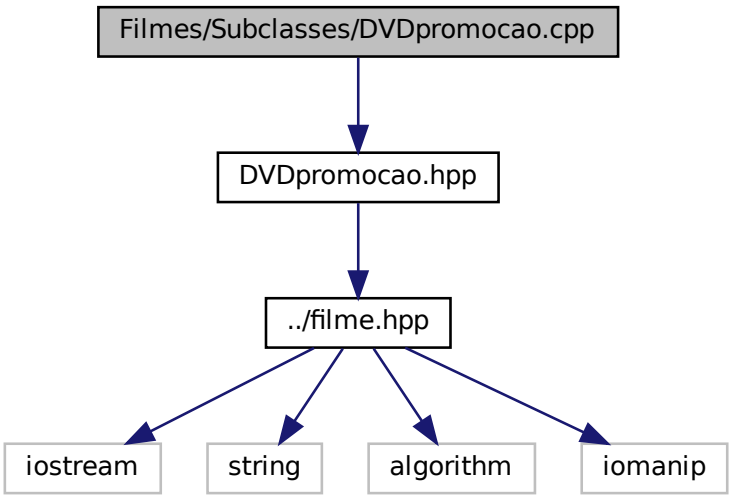
Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "DV↩ Dlancamemto.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "DVDlancamento.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.10 Filmes/Subclasses/DVDpromocao.cpp File Reference

```
#include "DVDpromocao.hpp"
```

Include dependency graph for DVDpromocao.cpp:

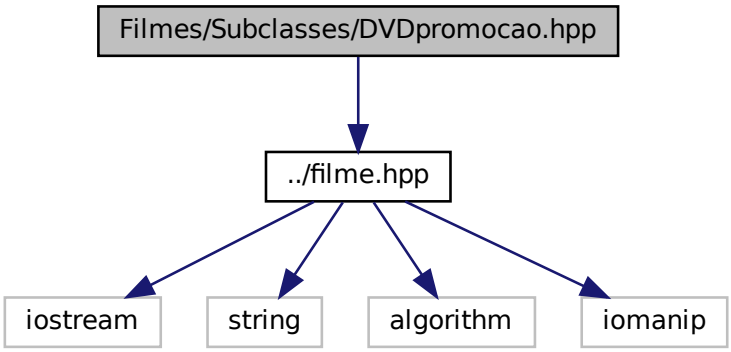


6.11 Filmes/Subclasses/DVDpromocao.hpp File Reference

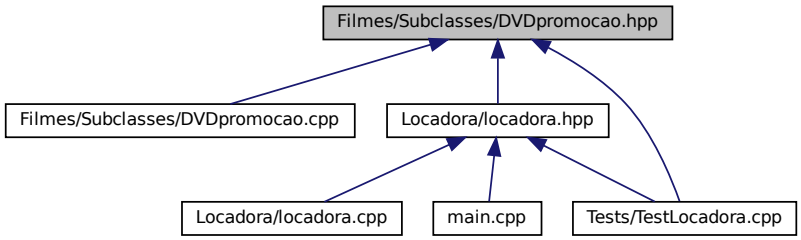
Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "DV↩ Dpromocao.cpp".

```
#include "../filme.hpp"
```

Include dependency graph for DVDpromocao.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class dvdPromocao

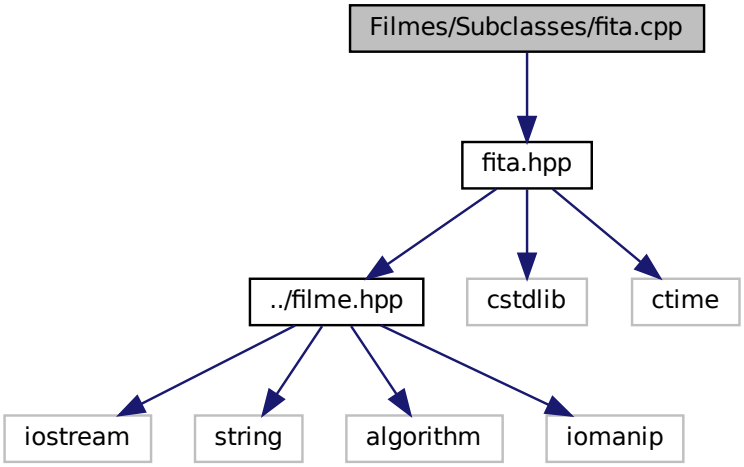
6.11.1 Detailed Description

Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "DV↩ Dpromocao.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "DVDpromocao.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.12 Filmes/Subclasses/fita.cpp File Reference

```
#include "fita.hpp"
```
Include dependency graph for fita.cpp:



Functions

- bool gerarValorAleatorio ()

  Método "calcularValor" da classe "Fita", utilizado especialmente para controle e registro.

6.12.1 Function Documentation

6.12.1.1 gerarValorAleatorio()

```
bool gerarValorAleatorio ( )
```

Método "calcularValor" da classe "Fita", utilizado especialmente para controle e registro.

Esse método calcula quanto o cliente terá de pagar pelo aluguel da fita de acordo com a quantidade de dias que ela ficou alugada. Caso o cliente devolva a fita já rebobinada, o aluguel é fixo em 5 reais, caso contrário, é fixo em 7 reais. O atributo rebobinado é gerado aleatoriamente pela função gerarValorAleatorio().

Parameters

dias	Armazena quantos dias um determinado filme (no caso, fita) ficou alugado.

Returns

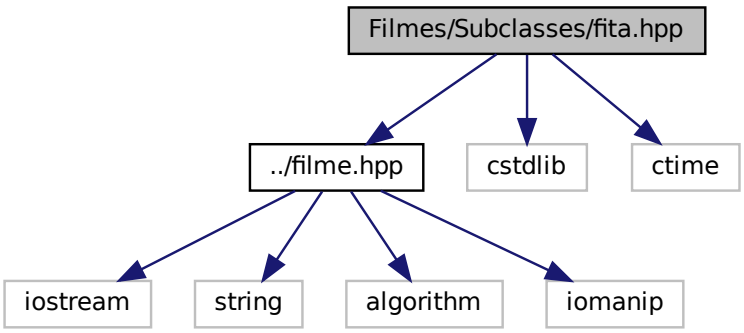
Retorna o preço que o cliente terá de pagar pelo aluguel da fita, sendo 5 reais fixos o preço da locação de uma fita rebobinada e 7 reais fixos o de uma não rebobinada.

6.13 Filmes/Subclasses/fita.hpp File Reference

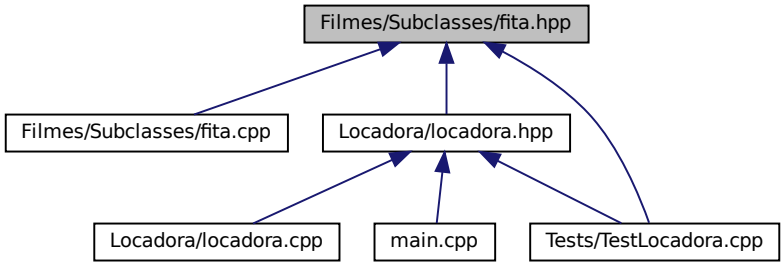
Arquivo de cabeçalho no qual contém os atributos e a especifição dos métodos utilizados no arquivo "fita.cpp".

```
#include "../filme.hpp"
#include <cstdlib>
#include <ctime>
```

Include dependency graph for fita.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Fita

6.13.1 Detailed Description

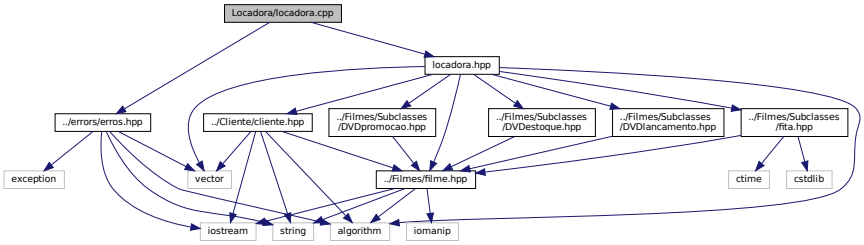
Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "fita.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "fita.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.14 Locadora/locadora.cpp File Reference

```
#include "locadora.hpp"
#include "../errors/erros.hpp"
```

Include dependency graph for locadora.cpp:

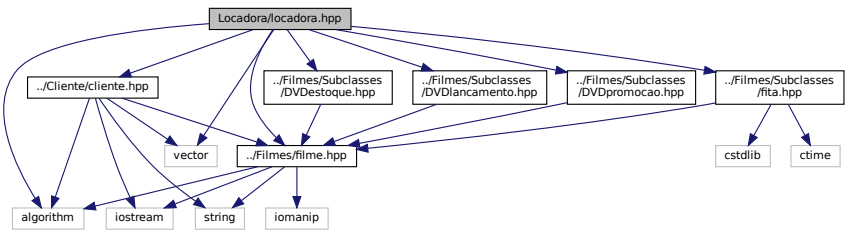


6.15 Locadora/locadora.hpp File Reference

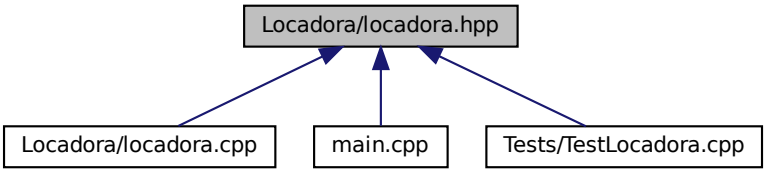
Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "locadora.cpp".

```
#include "../Filmes/filme.hpp"
#include "../Cliente/cliente.hpp"
#include "../Filmes/Subclasses/DVDestoque.hpp"
#include "../Filmes/Subclasses/DVDlancamento.hpp"
#include "../Filmes/Subclasses/DVDpromocao.hpp"
#include "../Filmes/Subclasses/fita.hpp"
#include <vector>
#include <algorithm>
```

Include dependency graph for locadora.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Locadora

6.15.1 Detailed Description

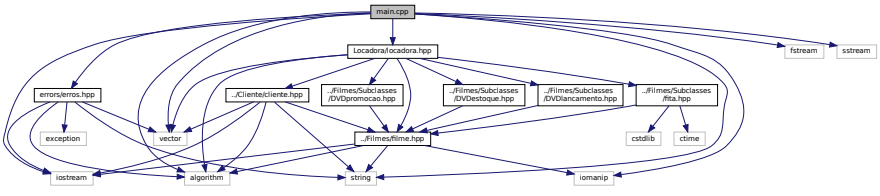
Arquivo de cabeçalho no qual contém os atributos e a especificação dos métodos utilizados no arquivo "locadora.cpp".

Esse arquivo define a classe na qual contém os métodos que serão utilizados no desenvolvimento do código do arquivo "locadora.cpp", especifica o tipo desses métodos (o que eles irão retornar) e os parâmetros que serão recebidos por eles.

6.16 main.cpp File Reference

```
#include "Locadora/locadora.hpp"
#include "errors/erros.hpp"
#include <fstream>
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
#include <iomanip>
#include <sstream>
```

Include dependency graph for main.cpp:



Functions

- void printColoredText (const std::string &text, int colorCode)
- void menu ()

  Função "void menu()", utilizada especialmente para exibição.

6.16.1 Function Documentation

6.16.1.1 menu()

```
void menu ( )
```

Função "void menu()", utilizada especialmente para exibição.

Essa função exibe todos os comandos disponíveis para o usuário.

6.16.1.2 printColoredText()

```
void printColoredText (
            const std::string & text,
            int colorCode )
```

Função que colore a interface do sistema para o usuário.

6.17 README.md File Reference

6.18 testla.txt File Reference

6.19 Tests/TestCliente.cpp File Reference

```
#include "../doctest.h"
#include "../Cliente/cliente.hpp"
#include "../Filmes/Subclasses/DVDestoque.hpp"
#include "../Filmes/filme.hpp"
```

Include dependency graph for TestCliente.cpp:

Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

Functions

- string saida_cliente (Cliente *lercliente)
- void zerar_v (Cliente *cliente)
- TEST_CASE ("Cliente-lerCliente")
- TEST_CASE ("Cliente-alugar")
- TEST_CASE ("Cliente-devolver")
- TEST_CASE ("Cliente-calcularSimilaridade")
- TEST_CASE ("Cliente-definirSimilares")
- TEST_CASE ("Cliente-recomendarPorSimiliar")

Variables

- dvdEstoque dvdE (1, "Divergente", 10)
- dvdEstoque dvdE2 (2, "Feliz", 11)
- dvdEstoque dvdE3 (3, "CBUM", 4)
- Filme * filme =&dvdE
- Filme * filme2 =&dvdE2
- Filme * filme3 =&dvdE3
- Cliente cliente1 (1, "luisa")
- Cliente cliente2 (2, "bruna")
- Cliente cliente3 (3, "warley")
- Cliente cliente4 (4, "samuel")
- Cliente * pointerCliente1 = &cliente1
- Cliente * pointerCliente2 = &cliente2
- Cliente * pointerCliente3 = &cliente3
- Cliente * pointerCliente4 = &cliente4
- vector< Cliente * > clientes = {pointerCliente2, pointerCliente3, pointerCliente4}

6.19.1 Macro Definition Documentation

6.19.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

6.19.2 Function Documentation

6.19.2.1 saida_cliente()

```
string saida_cliente (
            Cliente * lercliente )
```

6.19.2.2 TEST_CASE() [1/6]

```
TEST_CASE (
             "Cliente-alugar"  )
```

6.19.2.3 TEST_CASE() [2/6]

```
TEST_CASE (
             "Cliente-calcularSimilaridade"  )
```

6.19.2.4 TEST_CASE() [3/6]

```
TEST_CASE (
             "Cliente-definirSimilares"  )
```

6.19.2.5 TEST_CASE() [4/6]

```
TEST_CASE (
             "Cliente-devolver"  )
```

6.19.2.6 TEST_CASE() [5/6]

```
TEST_CASE (
             "Cliente-lerCliente"  )
```

6.19.2.7 TEST_CASE() [6/6]

```
TEST_CASE (
             "Cliente-recomendarPorSimiliar"  )
```

6.19.2.8 zerar_v()

```
void zerar_v (
             Cliente * cliente )
```

6.19.3 Variable Documentation

6.19.3.1 cliente1

```
Cliente cliente1(1, "luisa")
```

6.19.3.2 cliente2

```
Cliente cliente2(2, "bruna")
```

6.19.3.3 cliente3

```
Cliente cliente3(3, "warley")
```

6.19.3.4 cliente4

```
Cliente cliente4(4, "samuel")
```

6.19.3.5 clientes

```
vector<Cliente*> clientes = {pointerCliente2, pointerCliente3, pointerCliente4}
```

6.19.3.6 dvdE

```
dvdEstoque dvdE(1, "Divergente", 10)
```

6.19.3.7 dvdE2

```
dvdEstoque dvdE2(2, "Feliz", 11)
```

6.19.3.8 dvdE3

```
dvdEstoque dvdE3(3, "CBUM", 4)
```

6.19.3.9 filme

```
Filme* filme =&dvdE
```

6.19.3.10 filme2

```
Filme* filme2 =&dvdE2
```

6.19.3.11 filme3

```
Filme* filme3 =&dvdE3
```

6.19.3.12 pointerCliente1

```
Cliente* pointerCliente1 = &cliente1
```

6.19.3.13 pointerCliente2

```
Cliente* pointerCliente2 = &cliente2
```

6.19.3.14 pointerCliente3

```
Cliente* pointerCliente3 = &cliente3
```

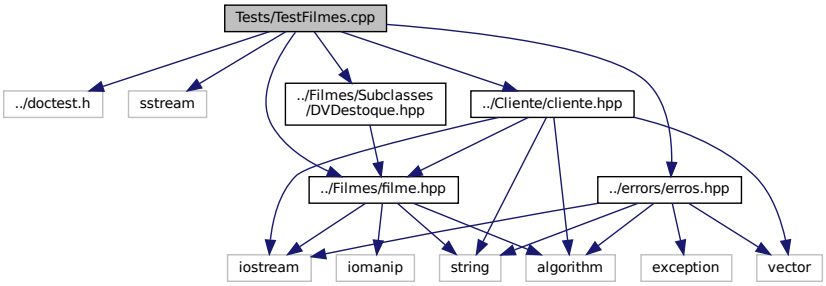
6.19.3.15 pointerCliente4

```
Cliente* pointerCliente4 = &cliente4
```

6.20 Tests/TestFilmes.cpp File Reference

```
#include "../doctest.h"
#include <sstream>
#include "../Filmes/filme.hpp"
#include "../Cliente/cliente.hpp"
#include "../Filmes/Subclasses/DVDestoque.hpp"
#include "../errors/erros.hpp"
```

Include dependency graph for TestFilmes.cpp:



Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

Enumerations

- enum Metodo { ler_filme, ser_alugado, ser_avaliado }

Functions

- string saida_filme (Filme *filme, Metodo metodo)
- TEST_CASE ("Filmes-lerFilme")
- TEST_CASE ("Filmes-serAlugado")
- TEST_CASE ("Filmes-serDevolvido")
- TEST_CASE ("Filmes-serAvaliado")

Variables

- dvdEstoque dvdE (1, "Divergente", 0)
- Filme * filme1 =&dvdE
- dvdEstoque dvdE2 (1, "Divergente", 2)
- Filme * filme2 =&dvdE2

6.20.1 Macro Definition Documentation

6.20.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

6.20.2 Enumeration Type Documentation

6.20.2.1 Metodo

```
enum Metodo
```

Enumerator

ler_filme	
ser_alugado	
ser_avaliado	

6.20.3 Function Documentation

6.20.3.1 saida_filme()

```
string saida_filme (
            Filme * filme,
            Metodo metodo )
```

6.20.3.2 TEST_CASE() [1/4]

```
TEST_CASE (
            "Filmes-lerFilme"  )
```

6.20.3.3 TEST_CASE() [2/4]

```
TEST_CASE (
            "Filmes-serAlugado"  )
```

6.20.3.4 TEST_CASE() [3/4]

```
TEST_CASE (
             "Filmes-serAvaliado"  )
```

6.20.3.5 TEST_CASE() [4/4]

```
TEST_CASE (
             "Filmes-serDevolvido"  )
```

6.20.4 Variable Documentation

6.20.4.1 dvdE

```
dvdEstoque dvdE(1, "Divergente", 0)
```

6.20.4.2 dvdE2

```
dvdEstoque dvdE2(1, "Divergente", 2)
```

6.20.4.3 filme1

```
Filme* filme1 =&dvdE
```

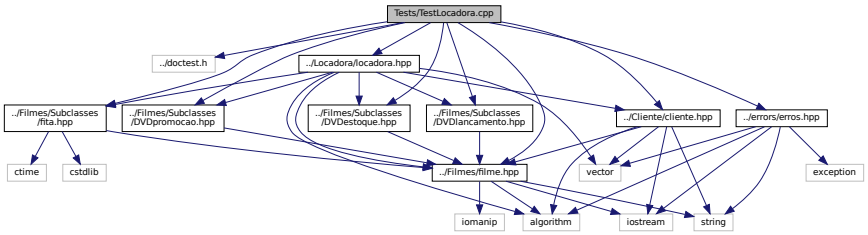
6.20.4.4 filme2

```
Filme* filme2 =&dvdE2
```

6.21 Tests/TestLocadora.cpp File Reference

```
#include "../doctest.h"
#include "../Locadora/locadora.hpp"
#include "../Filmes/filme.hpp"
#include "../Cliente/cliente.hpp"
#include "../Filmes/Subclasses/DVDestoque.hpp"
#include "../Filmes/Subclasses/DVDlancamento.hpp"
#include "../Filmes/Subclasses/DVDpromocao.hpp"
#include "../Filmes/Subclasses/fita.hpp"
#include "../errors/erros.hpp"
```

Include dependency graph for TestLocadora.cpp:



Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

Functions

- string saida_filme (Filme *filme)
- string saida_cliente (Cliente *lercliente)
- TEST_CASE ("Locadora-cadastrarFilme")
- TEST_CASE ("Locadora-removerFilme")
- TEST_CASE ("Locadora-listarFilmes")
- TEST_CASE ("Locadora-cadasstrarCliente")
- TEST_CASE ("Locadora-removerCliente")
- TEST_CASE ("Locadora-listarClientes")
- TEST_CASE ("Locadora-aluguel")
- TEST_CASE ("Locadora-devolução")
- TEST_CASE ("Locadora-avaliarFilme")

Variables

- Locadora minhalocadora
- Locadora locadorazerada
- vector< int > id = {minhalocadora.filmes[0]->id, minhalocadora.filmes[1]->id}

6.21.1 Macro Definition Documentation

6.21.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

6.21.2 Function Documentation

6.21.2.1 saida_cliente()

```
string saida_cliente (
            Cliente * lercliente )
```

6.21.2.2 saida_filme()

```
string saida_filme (
            Filme * filme )
```

6.21.2.3 TEST_CASE() [1/9]

```
TEST_CASE (
            "Locadora-aluguel"  )
```

6.21.2.4 TEST_CASE() [2/9]

```
TEST_CASE (
            "Locadora-avaliarFilme"  )
```

6.21.2.5 TEST_CASE() [3/9]

```
TEST_CASE (
            "Locadora-cadasstrarCliente"  )
```

6.21.2.6 TEST_CASE() [4/9]

```
TEST_CASE (
             "Locadora-cadastrarFilme"  )
```

6.21.2.7 TEST_CASE() [5/9]

```
TEST_CASE (
             "Locadora-devolução"  )
```

6.21.2.8 TEST_CASE() [6/9]

```
TEST_CASE (
             "Locadora-listarClientes"  )
```

6.21.2.9 TEST_CASE() [7/9]

```
TEST_CASE (
             "Locadora-listarFilmes"  )
```

6.21.2.10 TEST_CASE() [8/9]

```
TEST_CASE (
             "Locadora-removerCliente"  )
```

6.21.2.11 TEST_CASE() [9/9]

```
TEST_CASE (
             "Locadora-removerFilme"  )
```

6.21.3 Variable Documentation

6.21.3.1 id

```
vector<int> id = {minhalocadora.filmes[0]->id, minhalocadora.filmes[1]->id}
```

6.21.3.2 locadorazerada

```
Locadora locadorazerada
```

6.21.3.3 minhalocadora

```
Locadora minhalocadora
```