DOCUMENTAÇÃO_UFMG:CampusCard

AUTHOR
Versão 1.0

Sumário

Table of contents

Namespaces

Lista de Namespaces

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

Índice Hierárquico

Hierarquia de Classes

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

Índice dos Componentes

Lista de Classes

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

Índice dos Arquivos

Lista de Arquivos

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

Namespace

Refência do Namespace media

Namespaces

- namespace **ui**

Refência do Namespace media::ui

Componentes

- class **Login**class **MapasCampus**
- class **MapasInterno**
- class **Menu**
- class **MenuCarteirinha**
- class **MenuDepartamento**
- class **MenuEventos**
- class **MenuGrade**
- class **MenuMapas**
- class **MenuOnibus**
- class **MenuPrincipal**
- class **MenuSalas**
- class **MenuTransacoesRU**
- class **MenuTransporte**
- class **MenuUteis**
- class **MenuVans**

Classes

Referência da Classe BancoDeDados

`#include <SimulaBancodedados.hpp>`

Membros Públicos

- **BancoDeDados** ()
- double **recupera_saldo_inicial** (long int _matricula) const

---

Construtores e Destrutores

BancoDeDados::BancoDeDados ()

Documentação das funções

double BancoDeDados::recupera_saldo_inicial (long int _*matricula*) const

---

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

SimulaBancodedados.hpp

Referência da Classe CadastroSala

`#include <SalasUteis.hpp>`

Membros Públicos

- void **cadastrarSala** (const std::string &nome, const std::string &predio, const std::string &numero)
- void **exibirSalas** () const
- void **salvarCadastro** () const
- void **carregarCadastro** ()
- void **alterarSala** (const std::string &nome)

Documentação das funções

void CadastroSala::alterarSala (const std::string & *nome*)

void CadastroSala::cadastrarSala (const std::string & *nome*, const std::string & *predio*, const std::string & *numero*)

void CadastroSala::carregarCadastro ()

void CadastroSala::exibirSalas () const

void CadastroSala::salvarCadastro () const

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

SalasUteis.hpp

Referência da Classe CarteiraEstudante

`#include <CarteiraEstudante.hpp>`

Membros Públicos

- **CarteiraEstudante** ()
- void **GerarCarteiraDigital** (const char *)
- bool **obterStatusValidade** () const
- void **definirStatusValidade** (bool novo_status)

Construtores e Destrutores

CarteiraEstudante::CarteiraEstudante ()

Documentação das funções

void CarteiraEstudante::definirStatusValidade (bool *novo_status*)

void CarteiraEstudante::GerarCarteiraDigital (const char *)

bool CarteiraEstudante::obterStatusValidade () const

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

CarteiraEstudante.hpp

Referência da Classe Eventos

`#include <eventos.hpp>`

Membros Públicos

- void **editarTarefas** (std::vector< **Eventos** > &tarefas)
  função que permite editar as tarefas

- void **adicionarTarefas** (std::vector< **Eventos** > &tarefas)
  função que que permite editar adicionar disciplinas em um vetor

- void **exibirPorPrazo** (std::vector< **Eventos** > &tarefas)
  função que permite exibir as informações e prazos das tarefas

- void **displayTarefa** () const
  função responsável pela impressão da lista

- std::pair< int, int > **tempo** () const
  função que calcula o tempo restante até o prazo da tarefa

- bool **operator<** (const **Eventos** &outros) const
  operador que compara a tarefa com o prazo

- void **salvarTarefasEmArquivo** (const std::vector< **Eventos** > &tarefas, const std::string &nomeArquivo)
  função que salva a tarefas cadastrada em um arquivo

- void **carregarTarefasDeArquivo** (std::vector< **Eventos** > &tarefas, const std::string &nomeArquivo)
  função que salva a tarefas cadastrada em um arquivo

- void **setNome** (const std::string &nome)
  funções set para alterar o parâmetro privado

- void **setDescricao** (const std::string &descricao)
- void **setData** (const std::string &data)
- void **setHora** (const std::string &hora)
- void **setDisciplina** (const std::string &disciplina)
- std::string **getNome** () const
  funções get obtem o parâmetro privado

- std::string **getDescricao** () const
- std::string **getData** () const
- std::string **getHora** () const
- std::string **getDisciplina** () const

Documentação das funções

void Eventos::adicionarTarefas (std::vector< Eventos > & *tarefas*)

função que que permite editar adicionar disciplinas em um vetor

Parâmetros

tarefas	vetor dos dados das tarefas

void Eventos::carregarTarefasDeArquivo (std::vector< Eventos > & *tarefas*, const std::string & *nomeArquivo*)

função que salva a tarefas cadastrada em um arquivo

Parâmetros

nomeArquivo	nome do arquivo em que os dados serão salvos
tarefas	vector com todas as tarefas

void Eventos::displayTarefa () const

função responsável pela impressão da lista

void Eventos::editarTarefas (std::vector< Eventos > & *tarefas*)

função que permite editar as tarefas

Parâmetros

tarefas	vetor dos dados das tarefas

void Eventos::exibirPorPrazo (std::vector< Eventos > & *tarefas*)

função que permite exibir as informações e prazos das tarefas

Parâmetros

tarefas	vetor dos dados das tarefas

std::string Eventos::getData () const

std::string Eventos::getDescricao () const

std::string Eventos::getDisciplina () const

std::string Eventos::getHora () const

std::string Eventos::getNome () const

funções get obtem o parâmetro privado

Retorna

o item privado desejado

bool Eventos::operator< (const Eventos & *outros*) const

operador que compara a tarefa com o prazo

Parâmetros

tarefas	vetor dos dados das tarefas

void Eventos::salvarTarefasEmArquivo (const std::vector< Eventos > & *tarefas*, const std::string & *nomeArquivo*)

função que salva a tarefas cadastrada em um arquivo

Parâmetros

nomeArquivo	nome do arquivo em que os dados serão salvos
tarefas	vector com todas as tarefas

void Eventos::setData (const std::string & *data*)

void Eventos::setDescricao (const std::string & *descricao*)

void Eventos::setDisciplina (const std::string & *disciplina*)

void Eventos::setHora (const std::string & *hora*)

void Eventos::setNome (const std::string & *nome*)

funções set para alterar o parâmetro privado

Parâmetros

nome	item privado refrente ao nome da tarefa
data	item privado refrente a data da tarefa
descricao	item privado refrente a descrição da tarefa
hora	item privado refrente a hora da tarefa
disciplina	item privado refrente a disciplina da tarefa

std::pair< int, int > Eventos::tempo () const

função que calcula o tempo restante até o prazo da tarefa

Retorna

o resultado em dias e horas.

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

eventos.hpp

Referência da Classe Grade_Semanal

`#include <gradeSemanal.hpp>`

Membros Públicos

- void **criarGrade** (**Rotina** &rotina)
  função que cria uma grade e os associa com as disciplinas

- void **exibir_grade_semanal** (std::vector< std::vector< std::string > > &parametro)
  função que responsavel por imprimir a grade

Documentação das funções

void Grade_Semanal::criarGrade (Rotina & *rotina*)

função que cria uma grade e os associa com as disciplinas

Parâmetros

rotina	classe rotina

void Grade_Semanal::exibir_grade_semanal (std::vector< std::vector< std::string > > & *parametro*)

função que responsavel por imprimir a grade

Parâmetros

parametro	vetor de vetor correspondente a grade

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

gradeSemanal.hpp

Referência da Classe Interface

`#include <interface.hpp>`

Membros Públicos

- void **tituloSessao** (const std::string &sessao) const
  função que imprime um quadro para cada sessão

- void **mensagemSaida** ()
  função que imprime uma mensagem de sáida/fim do programa

Documentação das funções

void Interface::mensagemSaida ()

função que imprime uma mensagem de sáida/fim do programa

void Interface::tituloSessao (const std::string & *sessao*) const

função que imprime um quadro para cada sessão

Parâmetros

sessao	nome da sessao

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

interface.hpp

Referência da Classe media::ui::Login

`#include <login.hpp>`

Membros Públicos

- void **exibirMenuLogin** (**Usuario** &usuario)
- void **renderLogin** ()

Documentação das funções

void media::ui::Login::exibirMenuLogin (Usuario & *usuario*)

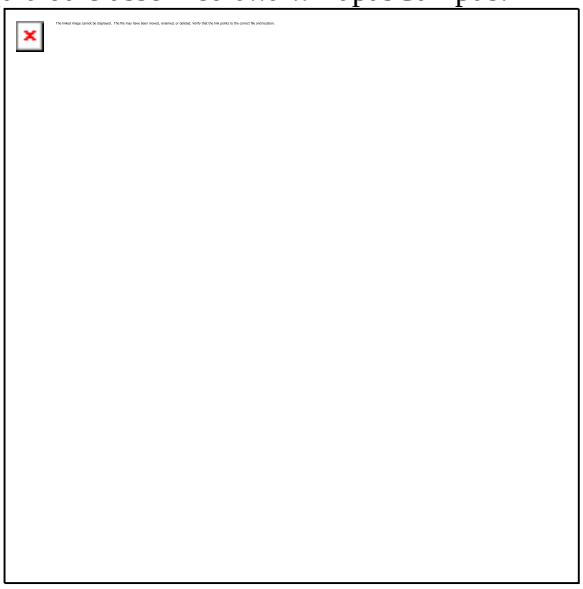
void media::ui::Login::renderLogin ()

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

login.hpp

Referência da Classe media::ui::MapasCampus

`#include <MapasCampus.hpp>`
Diagrama de hierarquia da classe media::ui::MapasCampus:



Membros Públicos

- **MapasCampus** ()
  : constroi um menu de opções

- **Menu** * **next** (unsigned option) override
  abre as opções dos usuários

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MapasCampus::MapasCampus ()

: constroi um menu de opções

Documentação das funções

Menu * media::ui::MapasCampus::next (unsigned *option*) `[override]`, `[virtual]`

abre as opções dos usuários

Retorna

o mapa escolhido pelo usuario

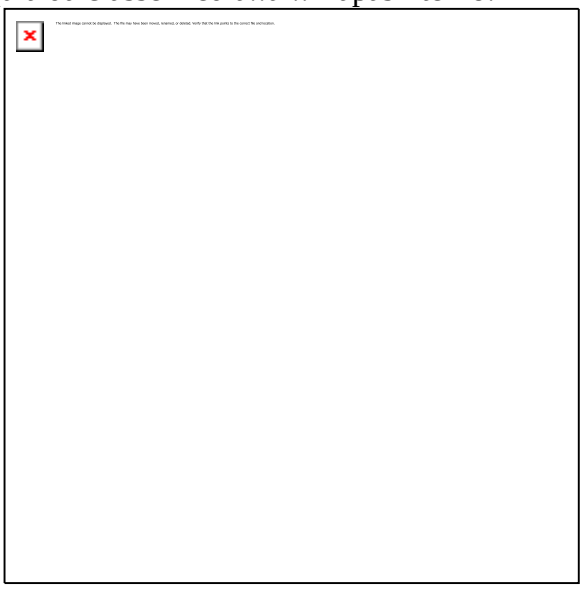
Implementa **media::ui::Menu** (*p.22*).

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

MapasCampus.hpp

Referência da Classe media::ui::MapasInterno

`#include <MapasInterno.hpp>`
Diagrama de hierarquia da classe media::ui::MapasInterno:



Membros Públicos

- **MapasInterno** ()
  : constroi um menu de opções

- **Menu** * **next** (unsigned option) override
  abre as opções dos usuários

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MapasInterno::MapasInterno ()

: constroi um menu de opções

Documentação das funções

Menu * media::ui::MapasInterno::next (unsigned *option*) `[override], [virtual]`

abre as opções dos usuários

Retorna

o mapa escolhido pelo usuario

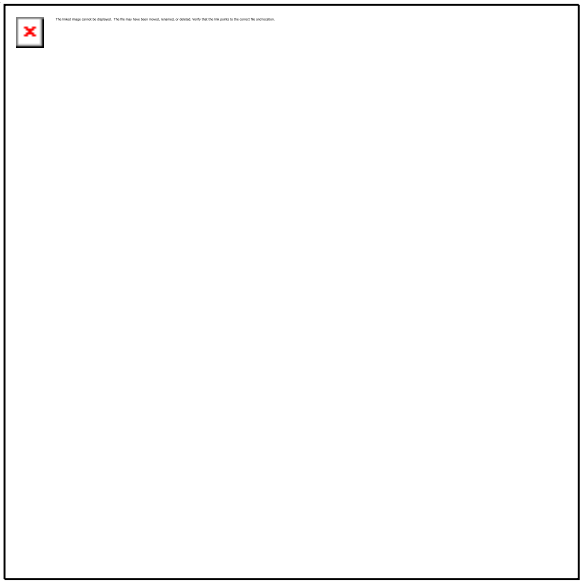
Implementa **media::ui::Menu** (*p.22*).

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

MapasInterno.hpp

Referência da Classe media::ui::Menu

`#include <menu.hpp>`
Diagrama de hierarquia da classe media::ui::Menu:



Membros Públicos

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

- virtual **Menu** * **next** (unsigned option)=0

Atributos Protegidos

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

virtual media::ui::Menu::~Menu () `[virtual], [default]`

Destrutor virtual, para correta destruição de subclasses.

Documentação das funções

virtual Menu * media::ui::Menu::next (unsigned *option*) `[pure virtual]`

Retorna

O próximo menu, a partir da seleção do usuário.

Implementado por **media::ui::MenuPrincipal** (*p.37*), **media::ui::MapasCampus** (*p.19*), **media::ui::MapasInterno** (*p.21*), **media::ui::MenuCarteirinha** (*p.25*), **media::ui::MenuDepartamento** (*p.27*), **media::ui::MenuEventos** (*p.29*), **media::ui::MenuGrade** (*p.31*), **media::ui::MenuMapas** (*p.33*), **media::ui::MenuOnibus** (*p.35*), **media::ui::MenuSalas** (*p.39*), **media::ui::MenuTransacoesRU** (*p.41*), **media::ui::MenuTransporte** (*p.43*), **media::ui::MenuUteis** (*p.45*) e **media::ui::MenuVans** (*p.47*).

virtual void media::ui::Menu::render () const `[virtual]`

Renderiza as opções do menu.

Atributos

std::vector<std::string> media::ui::Menu::_options = {"0 - Sair"} `[protected]`

```
21 {"0 - Sair"};
```

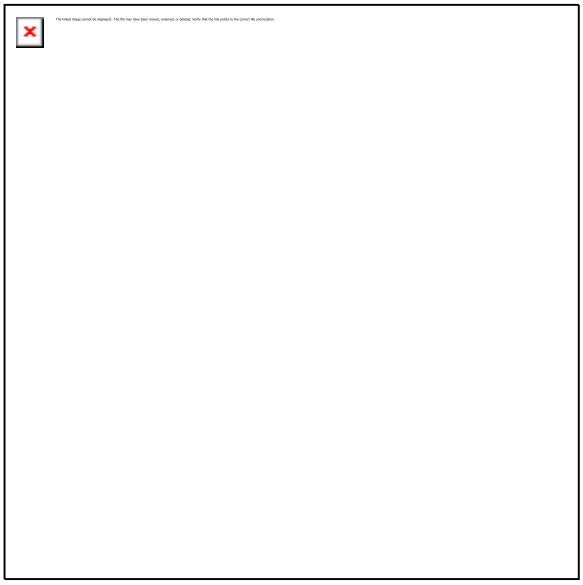
std::string media::ui::Menu::_title = "Menu Principal" `[protected]`

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

menu.hpp

Referência da Classe media::ui::MenuCarteirinha

`#include <menuCarteirinha.hpp>`
Diagrama de hierarquia da classe media::ui::MenuCarteirinha:



Membros Públicos

- **MenuCarteirinha** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuCarteirinha::MenuCarteirinha ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuCarteirinha::next (unsigned *option*) `[override]`, `[virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

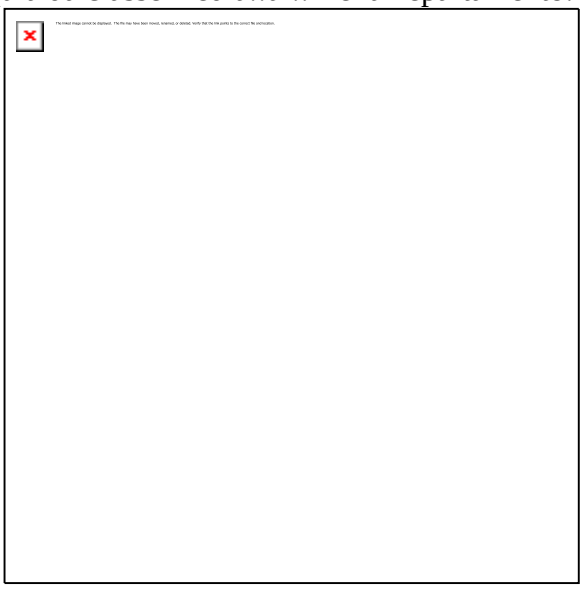
A documentação para essa classe foi gerada a partir do seguinte arquivo:

menuCarteirinha.hpp

Referência da Classe media::ui::MenuDepartamento

`#include <menuDepartamento.hpp>`

Diagrama de hierarquia da classe media::ui::MenuDepartamento:



Membros Públicos

- **MenuDepartamento** ()
  : constroi um menu de opcoes

- **Menu** * **next** (unsigned option) override
  abre as opcoes dos usuarios

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuDepartamento::MenuDepartamento ()

: constroi um menu de opcoes

Documentação das funções

Menu * media::ui::MenuDepartamento::next (unsigned *option*) `[override], [virtual]`

abre as opcoes dos usuarios

Retorna

a opcaoo escolhido pelo usuario

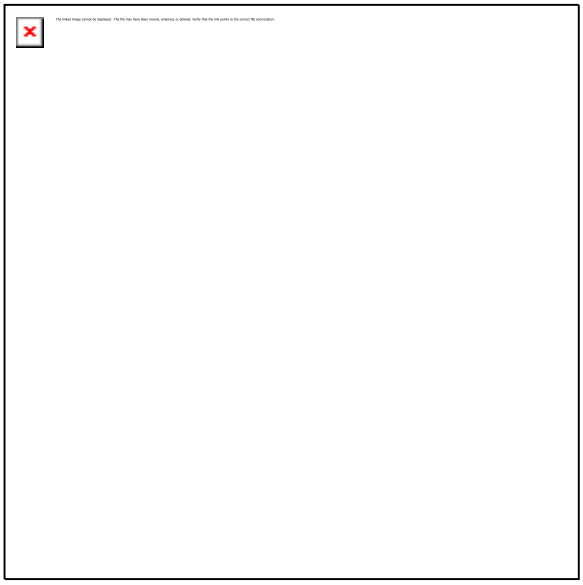
Implementa **media::ui::Menu** (*p.22*).

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

menuDepartamento.hpp

Referência da Classe media::ui::MenuEventos

`#include <menuEventos.hpp>`
Diagrama de hierarquia da classe media::ui::MenuEventos:



Membros Públicos

- **MenuEventos** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuEventos::MenuEventos ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuEventos::next (unsigned *option*) [override], [virtual]

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

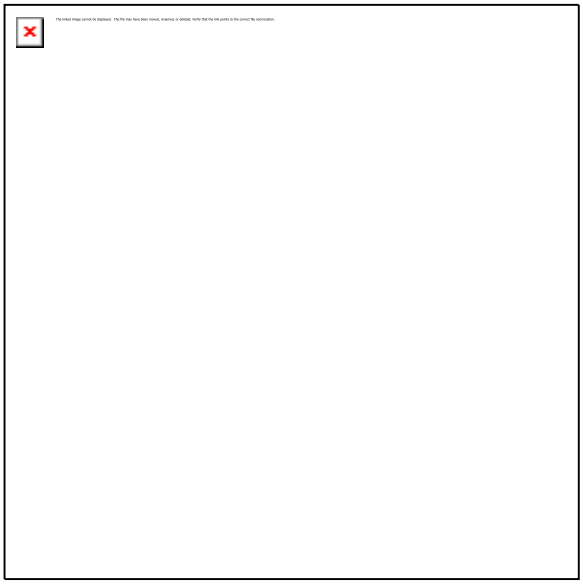
Implementa **media::ui::Menu** (*p.22*).

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

menuEventos.hpp

Referência da Classe media::ui::MenuGrade

`#include <menuGradeSemanal.hpp>`
Diagrama de hierarquia da classe media::ui::MenuGrade:



Membros Públicos

- **MenuGrade** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuGrade::MenuGrade ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuGrade::next (unsigned *option*) `[override], [virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

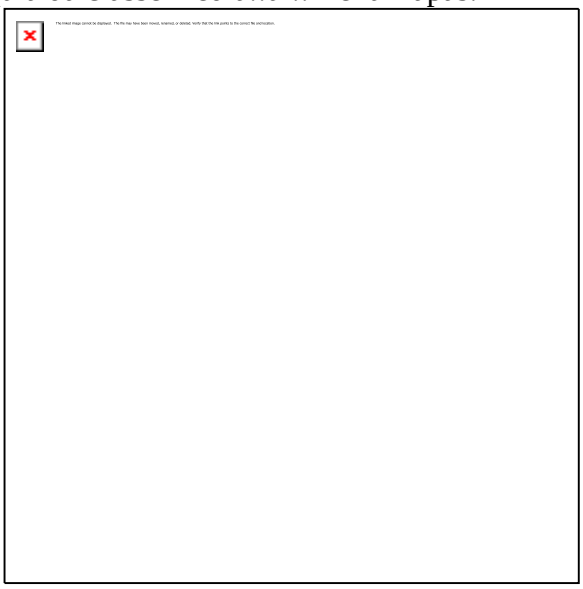
A documentação para essa classe foi gerada a partir do seguinte arquivo:

menuGradeSemanal.hpp

Referência da Classe media::ui::MenuMapas

`#include <menuMapas.hpp>`

Diagrama de hierarquia da classe media::ui::MenuMapas:



Membros Públicos

- **MenuMapas** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (mapas dos campi ou de interno)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuMapas::MenuMapas ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuMapas::next (unsigned *option*) `[override], [virtual]`

Constrói próximo menu (mapas dos campi ou de interno)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

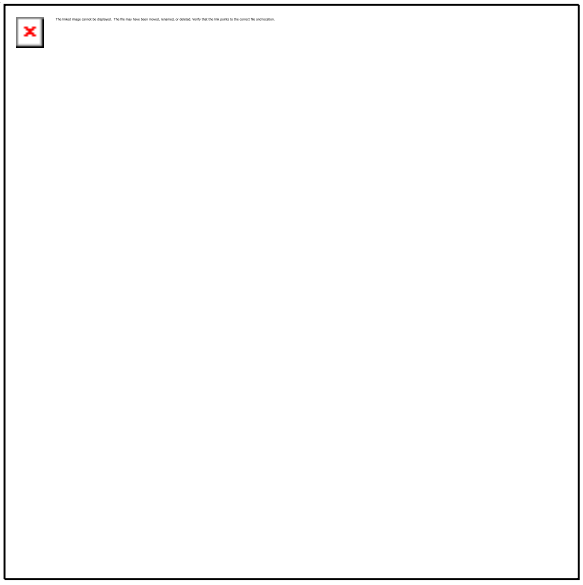
A documentação para essa classe foi gerada a partir do seguinte arquivo:

menuMapas.hpp

Referência da Classe media::ui::MenuOnibus

`#include <menuOnibus.hpp>`

Diagrama de hierarquia da classe media::ui::MenuOnibus:



Membros Públicos

- **MenuOnibus** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuOnibus::MenuOnibus ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuOnibus::next (unsigned *option*) `[override], [virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

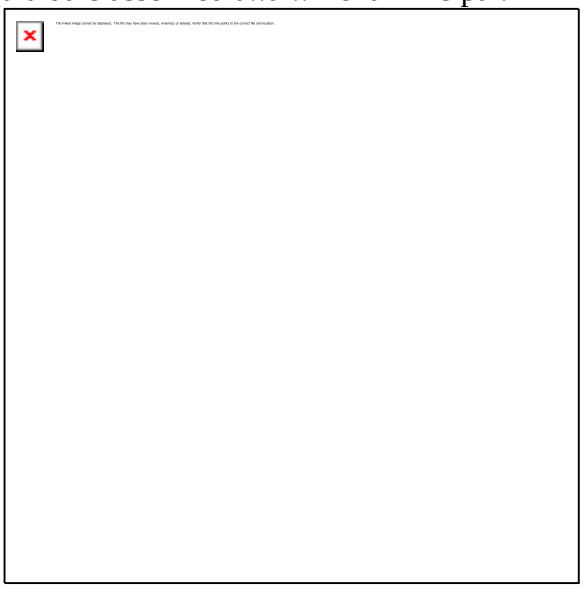
A documentação para essa classe foi gerada a partir do seguinte arquivo:

menuOnibus.hpp

Referência da Classe media::ui::MenuPrincipal

`#include <main_menu.hpp>`

Diagrama de hierarquia da classe media::ui::MenuPrincipal:



Membros Públicos

- **MenuPrincipal** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuPrincipal::MenuPrincipal ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuPrincipal::next (unsigned *option*) `[override], [virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

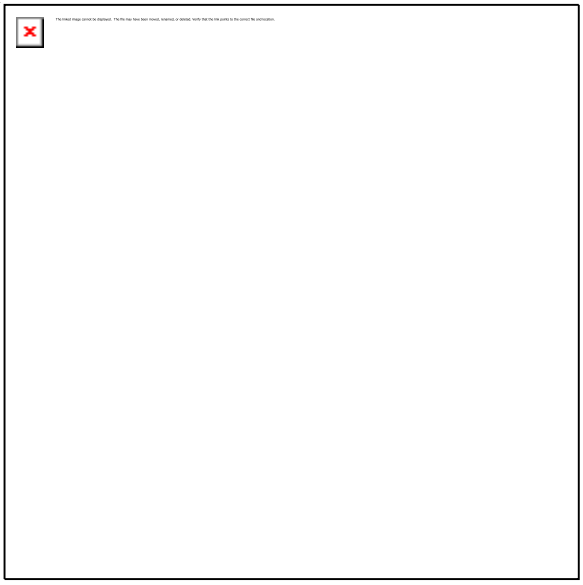
A documentação para essa classe foi gerada a partir do seguinte arquivo:

main_menu.hpp

Referência da Classe media::ui::MenuSalas

`#include <menuSalas.hpp>`

Diagrama de hierarquia da classe media::ui::MenuSalas:



Membros Públicos

- **MenuSalas** ()
  : constroi um menu principal das salas

- **Menu** * **next** (unsigned option) override
  Constroi proximo menu (salas uteis ou salas por depto)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuSalas::MenuSalas ()

: constroi um menu principal das salas

Documentação das funções

Menu * media::ui::MenuSalas::next (unsigned *option*) `[override], [virtual]`

Constroi proximo menu (salas uteis ou salas por depto)

Retorna

O proximo menu, a partir da selecao do usuario.

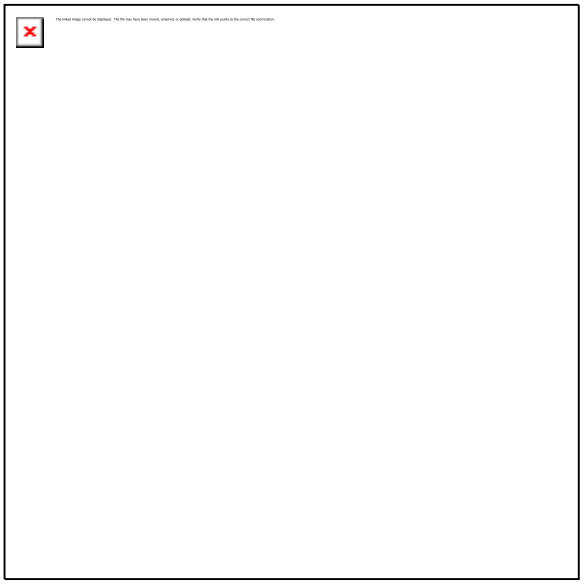
Implementa **media::ui::Menu** (*p.22*).

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

menuSalas.hpp

Referência da Classe media::ui::MenuTransacoesRU

`#include <menuTransacoesRU.hpp>`
Diagrama de hierarquia da classe media::ui::MenuTransacoesRU:



Membros Públicos

- **MenuTransacoesRU** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuTransacoesRU::MenuTransacoesRU ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuTransacoesRU::next (unsigned *option*) [override], [virtual]

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

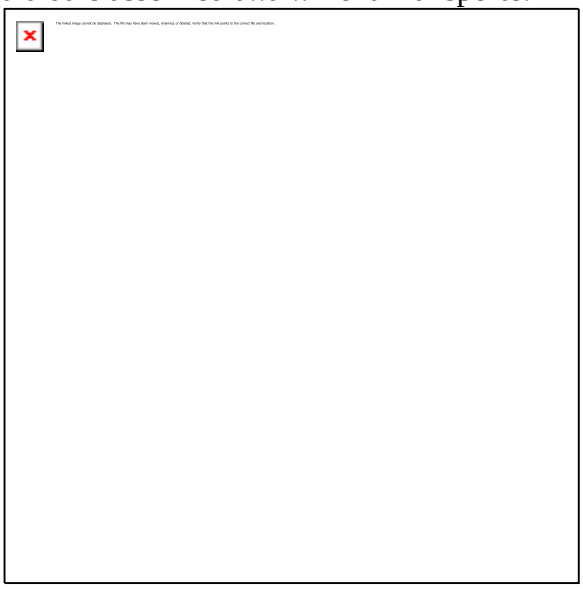
Implementa **media::ui::Menu** (*p.22*).

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

menuTransacoesRU.hpp

Referência da Classe media::ui::MenuTransporte

`#include <menuTransporte.hpp>`
Diagrama de hierarquia da classe media::ui::MenuTransporte:



Membros Públicos

- **MenuTransporte** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuTransporte::MenuTransporte ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuTransporte::next (unsigned *option*) `[override], [virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

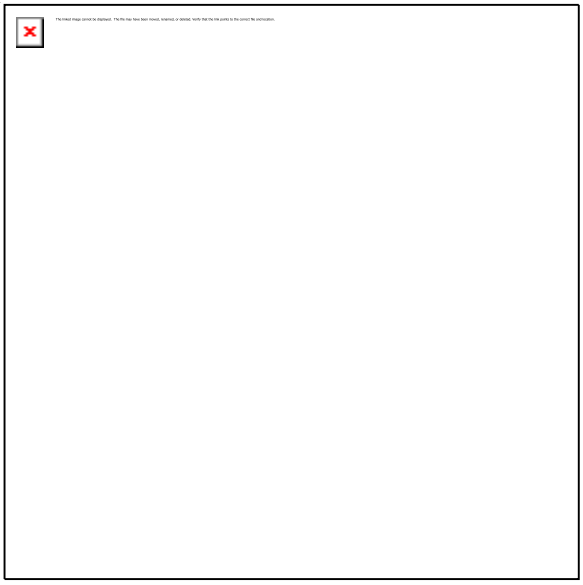
Implementa **media::ui::Menu** (*p.22*).

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

menuTransporte.hpp

Referência da Classe media::ui::MenuUteis

`#include <menuUteis.hpp>`
Diagrama de hierarquia da classe media::ui::MenuUteis:



Membros Públicos

- **MenuUteis** ()
  : constroi um menu de opcoes

- **Menu** * **next** (unsigned option) override
  abre as opcoes dos usuarios

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuUteis::MenuUteis ()

: constroi um menu de opcoes

Documentação das funções

Menu * media::ui::MenuUteis::next (unsigned *option*) `[override]`, `[virtual]`

abre as opcoes dos usuarios

Retorna

a opcaoo escolhido pelo usuario

Implementa **media::ui::Menu** (*p.22*).

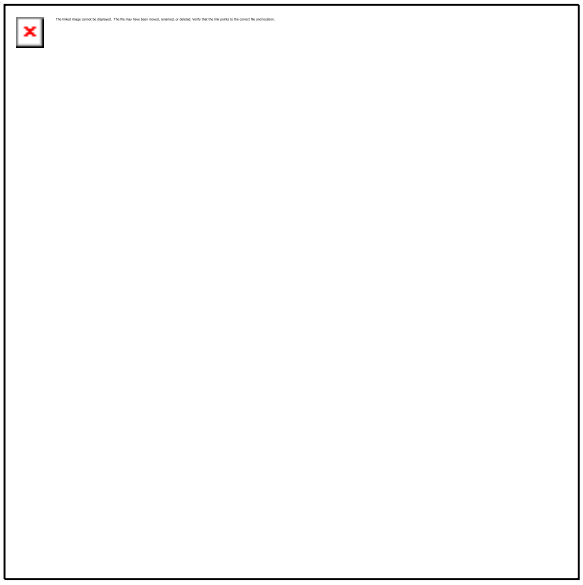
A documentação para essa classe foi gerada a partir do seguinte arquivo:

menuUteis.hpp

Referência da Classe media::ui::MenuVans

`#include <menuVans.hpp>`

Diagrama de hierarquia da classe media::ui::MenuVans:



Membros Públicos

- **MenuVans** ()
  : constrói um menu principal

- **Menu** * **next** (unsigned option) override
  Constrói próximo menu (criação de conta ou login)

Membros Públicos herdados de media::ui::Menu

- virtual **~Menu** ()=default
  Destrutor virtual, para correta destruição de subclasses.

- virtual void **render** () const
  Renderiza as opções do menu.

Outros membros herdados

Atributos Protegidos herdados de media::ui::Menu

- std::string **_title** = "Menu Principal"
- std::vector< std::string > **_options** = {"0 - Sair"}

Construtores e Destrutores

media::ui::MenuVans::MenuVans ()

: constrói um menu principal

Documentação das funções

Menu * media::ui::MenuVans::next (unsigned *option*) `[override]`, `[virtual]`

Constrói próximo menu (criação de conta ou login)

Retorna

O próximo menu, a partir da seleção do usuário.

Implementa **media::ui::Menu** (*p.22*).

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

menuVans.hpp

Referência da Classe Onibus

`#include <onibus.hpp>`

Membros Públicos

- void **cadastrarOnibus** ()
- void **exibirInformacao** () const
- void **editarOnibus** ()
- void **exibirOnibusCadastrados** () const
- void **carregarOnibus** ()

Documentação das funções

void Onibus::cadastrarOnibus ()

void Onibus::carregarOnibus ()

void Onibus::editarOnibus ()

void Onibus::exibirInformacao () const

void Onibus::exibirOnibusCadastrados () const

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

onibus.hpp

Referência da Classe Redirecionamento

`#include <redirecionamento.hpp>`

Membros públicos estáticos

- static void **redirecionarLink** (const std::string &link)

Documentação das funções

static void Redirecionamento::redirecionarLink (const std::string & *link*) `[static]`

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

redirecionamento.hpp

Referência da Classe Rotina

`#include <rotina.hpp>`

Membros Públicos

- const std::vector< Disciplina > & **getDisciplinas** () const
  função que permite que classes selecionadas acessem o vetor disciplina

- void **cadastrarDisciplina** ()
  função que adiciona disciplinas ao vetor

- void **editar_disciplina** ()
  função que permite alterar valores dos dados da disciplina

- void **informacoes_disciplina** ()
  função que exibe as informações de disciplina

- bool **existeDisciplinaNoHorario** (int dia, int hora) const
  função que permite a consulta se há uma disciplina cadastrada em um dia e hora

- void **cadastrarHorarios** (Horario &novoHorario)
  função que cadastra os horarios das disciplinas

- void **cadastrarDetalhes** (Disciplina &novadisciplina)
  função que cadastra as informações das disciplinas

- void **salvarDisciplinasEmArquivo** (const std::string &nomeArquivo) const
  função que salva a disciplina cadastrada em um arquivo

- void **carregarDisciplinasDeArquivo** (const std::string &nomeArquivo)
  função que adiciona as disciplinas do arquivo no vetor

Documentação das funções

void Rotina::cadastrarDetalhes (Disciplina & *novadisciplina*)

função que cadastra as informações das disciplinas

Parâmetros

novoDisciplina	vetor de struct

void Rotina::cadastrarDisciplina ()

função que adiciona disciplinas ao vetor

void Rotina::cadastrarHorarios (Horario & *novoHorario*)

função que cadastra os horarios das disciplinas

Parâmetros

novoHorario	vetor de struct

void Rotina::carregarDisciplinasDeArquivo (const std::string & *nomeArquivo*)

função que adiciona as disciplinas do arquivo no vetor

Parâmetros

nomeArquivo	nome do arquivo em que os dados serão salvos

void Rotina::editar_disciplina ()

função que permite alterar valores dos dados da disciplina

bool Rotina::existeDisciplinaNoHorario (int *dia*, int *hora*) const

função que permite a consulta se há uma disciplina cadastrada em um dia e hora

Parâmetros

hora	inteiro associado ao horario
dia	inteiro associado ao dia da semana

Retorna

retorna verdadeiro se há uma disciplina cadastrada no horário

retorna false se não há disciplina cadastrada no horário

const std::vector< Disciplina > & Rotina::getDisciplinas () const

função que permite que classes selecionadas acessem o vetor disciplina

Retorna

retorna o vetor privado

void Rotina::informacoes_disciplina ()

função que exibe as informações de disciplina

void Rotina::salvarDisciplinasEmArquivo (const std::string & *nomeArquivo*) const

função que salva a disciplina cadastrada em um arquivo

Parâmetros

nomeArquivo	nome do arquivo em que os dados serão salvos

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

rotina.hpp

Referência da Classe Sala

`#include <SalasUteis.hpp>`

Atributos Públicos

- std::string **nome**
- std::string **predio**
- std::string **numero**

Atributos

std::string Sala::nome

std::string Sala::numero

std::string Sala::predio

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

SalasUteis.hpp

Referência da Classe Saldo

`#include <Saldo.hpp>`

Membros Públicos

- **Saldo** ()
- **Saldo** (long int _matricula, **BancoDeDados** &banco)
- void **diminuir_saldo** (int _fump)
- void **adicionar_saldo** (double valor)
- double **retornar_saldo_atual** () const

Construtores e Destrutores

Saldo::Saldo ()

Saldo::Saldo (long int _*matricula*, BancoDeDados & *banco*)

Documentação das funções

void Saldo::adicionar_saldo (double *valor*)

void Saldo::diminuir_saldo (int _*fump*)

double Saldo::retornar_saldo_atual () const

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

Saldo.hpp

Referência da Classe Transacoes

`#include <Transacoes.hpp>`

Membros Públicos

- **Transacoes** ()
- void **set_pagamento** (bool **pagamento**)
- void **set_deposito** (bool **deposito**)
- void **pagamento** ()
- void **deposito** (double valor)
- void **consultar_saldo** ()
- void **preenche_pdf** (HPDF_Page page)
- void **gerar_GRU** (const char *guia)
- void **ler_qrcode** ()

Construtores e Destrutores

Transacoes::Transacoes ()

Documentação das funções

void Transacoes::consultar_saldo ()

void Transacoes::deposito (double *valor*)

void Transacoes::gerar_GRU (const char * *guia*)

void Transacoes::ler_qrcode ()

void Transacoes::pagamento ()

void Transacoes::preenche_pdf (HPDF_Page *page*)

void Transacoes::set_deposito (bool *deposito*)

void Transacoes::set_pagamento (bool *pagamento*)

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

Transacoes.hpp

Referência da Classe Usuario

`#include <usuario.hpp>`

Membros Públicos

- **Usuario** ()
  Constructor.

- long int **getMatricula** () const
  Getters.

- std::string **getNome** () const
- std::string **getCurso** () const
- std::string **getEmail** () const
- int **getNivelFump** () const
- long long int **getCPF** () const
- std::string **getEndereco** () const
- void **setMatricula** (long int matricula)
  Setters.

- void **setNome** (const std::string &nome)
- void **setCurso** (const std::string &curso)
- void **setEmail** (const std::string &email)
- void **setNivelFump** (int fump)
- void **setCPF** (long long int cpf)
- void **setEndereco** (const std::string &endereco)

Construtores e Destrutores

Usuario::Usuario ()

Constructor.

Documentação das funções

long long int Usuario::getCPF () const

std::string Usuario::getCurso () const

std::string Usuario::getEmail () const

std::string Usuario::getEndereco () const

long int Usuario::getMatricula () const

Getters.

int Usuario::getNivelFump () const

std::string Usuario::getNome () const

void Usuario::setCPF (long long int *cpf*)

void Usuario::setCurso (const std::string & *curso*)

void Usuario::setEmail (const std::string & *email*)

void Usuario::setEndereco (const std::string & *endereco*)

void Usuario::setMatricula (long int *matricula*)

Setters.

void Usuario::setNivelFump (int *fump*)

void Usuario::setNome (const std::string & *nome*)

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

usuario.hpp

Referência da Classe Validacao

`#include <validação.hpp>`

Membros Públicos

- int **obterNumeroInteiro** ()
- char **validarSN** (char escolha)
  função que confere e altera o S/N se o valor não estiver correto

- int **validarNumero** (int escolha)
  função que confere se o que foi digitado é um inteiro, e o altera se não for

- int **validarDia** (int escolha)
  função que confere se o que foi digitado é um valor correspondente a data, se não for o altera

- int **validarHorario** (int escolha)
  função que confere se o que foi digitado é um valor válido aos dias, se não for o altera

- std::string **transformarEmDia** (int dia)
  função que altera o valor digitado em inteiro para um char correspondente ao dia da semana

- std::string **transformarEmHora** (int hora)
  função que altera o valor digitado em inteiro para um char correspondente ao horario

- bool **validarFormatoData** (const std::string &data) const
  função que confere se o formato digitado pelo usuário é DD-MM-AA

- bool **validarFormatoHora** (const std::string &hora) const
  função que confere se o formato digitado pelo usuário é HH:MM

Documentação das funções

int Validacao::obterNumeroInteiro ()

std::string Validacao::transformarEmDia (int *dia*)

função que altera o valor digitado em inteiro para um char correspondente ao dia da semana

Parâmetros

dia	recebe o inteiro digitado associado ao dia

Retorna

retorna o dia associado ao número

std::string Validacao::transformarEmHora (int *hora*)

função que altera o valor digitado em inteiro para um char correspondente ao horario

Parâmetros

hora	recebe o inteiro digitado associado ao horario

Retorna

retorna o horario associado ao número

int Validacao::validarDia (int *escolha*)

função que confere se o que foi digitado é um valor correspondente a data, se não for o altera

Parâmetros

escolha	refere-se ao que foi digitado pelo usuário

Retorna

retorna um valor válido em formato inteiro e da data

bool Validacao::validarFormatoData (const std::string & *data*) const

função que confere se o formato digitado pelo usuário é DD-MM-AA

Parâmetros

data	recebe a data cadastrada

Retorna

verdadeiro ou falso

bool Validacao::validarFormatoHora (const std::string & *hora*) const

função que confere se o formato digitado pelo usuário é HH:MM

Parâmetros

hora	recebe a hora cadastrada

Retorna

verdadeiro ou falso

int Validacao::validarHorario (int *escolha*)

função que confere se o que foi digitado é um valor válido aos dias, se não for o altera

Parâmetros

escolha	refere-se ao que foi digitado pelo usuário

Retorna

retorna um valor válido em formato inteiro

int Validacao::validarNumero (int *escolha*)

função que confere se o que foi digitado é um inteiro, e o altera se não for

Parâmetros

escolha	refere-se ao que foi digitado pelo usuário

Retorna

retorna um valor válido em formato inteiro

char Validacao::validarSN (char *escolha*)

função que confere e altera o S/N se o valor não estiver correto

Parâmetros

escolha	refere-se ao que foi digitado pelo usuário

Retorna

retorna um valor válido em formato S/N

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

validação.hpp

Referência da Classe Van

`#include <van.hpp>`

Membros Públicos

- void **cadastrarVan** ()
- void **exibirInformacao** () const
- void **editarVan** ()
- void **exibirVansCadastradas** () const
- void **carregarVans** ()

Documentação das funções

void Van::cadastrarVan ()

void Van::carregarVans ()

void Van::editarVan ()

void Van::exibirInformacao () const

void Van::exibirVansCadastradas () const

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

Arquivos

Referência do Arquivo CarteiraEstudante.hpp

```
#include <iostream>
#include <string>
#include <sstream>
#include <hpdf.h>
#include "usuario.hpp"
```

Componentes

class **CarteiraEstudante**

CarteiraEstudante.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <sstream>
6 #include <hpdf.h>
7 #include "usuario.hpp"
8
9
10 class CarteiraEstudante{
11 public:
12     CarteiraEstudante();
13
14     void GerarCarteiraDigital(const char*);
15
16     bool obterStatusValidade() const;
17
18     void definirStatusValidade(bool novo_status);
19
20 private:
21     bool ValidadeCarteira;
22 };
```

Referência do Arquivo eventos.hpp

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

Componentes

class **Eventos**

eventos.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4 #include <chrono>
5 #include <iostream>
6 #include <algorithm>
7 #include <vector>
8
9 class Eventos{
10
11     private:
12
13         //informações das tarefas
14         std::string _nome;
15         std::string  descricao;
16         std::string  data;
17         std::string _hora;
18         std::string _disciplina;
19         std::chrono::system_clock::time_point deadline;
20
27         std::chrono::system clock::time point parseDateTime(const std::string&
 data, const std::string&  hora) const;
28
29     public:
30
35         void editarTarefas(std::vector<Eventos>& tarefas);
36
41         void adicionarTarefas(std::vector<Eventos>& tarefas);
42
47         void exibirPorPrazo(std::vector<Eventos>& tarefas);
48
52         void displayTarefa() const;
53
58         std::pair<int, int> tempo() const;
59
64         bool operator<(const Eventos& outros) const;
65
66
72         void salvarTarefasEmArquivo(const std::vector<Eventos>& tarefas, const
std::string& nomeArquivo);
73
79         void carregarTarefasDeArquivo(std::vector<Eventos>& tarefas, const
std::string& nomeArquivo);
80
89         void setNome(const std::string& nome);
90         void setDescricao(const std::string& descricao);
91         void setData(const std::string& data);
92         void setHora(const std::string& hora);
93         void setDisciplina(const std::string& disciplina);
94
99         std::string getNome() const;
100         std::string getDescricao() const;
101         std::string getData() const;
102         std::string getHora() const;
103         std::string getDisciplina() const;
104
105 };
```

Referência do Arquivo gradeSemanal.hpp

```
#include <iostream>
#include <string>
#include <chrono>
#include <vector>
#include <sstream>
#include <iomanip>
#include "rotina.hpp"
```

Componentes

class **Grade_Semanal**

gradeSemanal.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <chrono>
6 #include <vector>
7 #include <sstream>
8 #include <iomanip>
9 #include "rotina.hpp"
10
11
12     //classe em que se cria uma grade semanal, que receberá disciplinas
13     class Grade_Semanal{
14
15         private:
17             Rotina rotina;
18
19         public:
24             void criarGrade(Rotina &rotina);
25
30             void exibir grade semanal(std::vector<std::vector<std::string>>&
parametro);
31
32     };
```

Referência do Arquivo interface.hpp

```
#include <iostream>
```

Componentes

class **Interface**

interface.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4
5     class Interface{
6         public:
7
12          void tituloSessao(const std::string& sessao) const;
13
17          void mensagemSaida();
18
19      };
```

Referência do Arquivo login.hpp

```
#include "usuario.hpp"
```

Componentes

class media::ui::Login**Namespaces**

- namespace **media**
- namespace **media::ui**

login.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "usuario.hpp"
4
5 namespace media::ui{
6     class Login{
7     public:
8         void exibirMenuLogin(Usuario& usuario);
9         void renderLogin();
10     };
11
12 }
```

Referência do Arquivo main_menu.hpp

```
#include "menu.hpp"
#include <string>
#include "menuGradeSemanal.hpp"
#include "menuMapas.hpp"
#include "menuTransporte.hpp"
#include "menuOnibus.hpp"
#include "menuVans.hpp"
#include "menuEventos.hpp"
#include "menuSalas.hpp"
#include "menuTransacoesRU.hpp"
#include "menuCarteirinha.hpp"
```

Componentes

class media::ui::MenuPrincipal**Namespaces**

- namespace **media**
- namespace **media::ui**

main_menu.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5 #include "menuGradeSemanal.hpp"
6 #include "menuMapas.hpp"
7 #include "menuTransporte.hpp"
8 #include "menuOnibus.hpp"
9 #include "menuVans.hpp"
10 #include "menuEventos.hpp"
11 #include "menuSalas.hpp"
12 #include "menuTransacoesRU.hpp"
13 #include "menuCarteirinha.hpp"
14
15
16 namespace media::ui{
17     class MenuPrincipal : public Menu{
18     public:
20         MenuPrincipal();
21
24         Menu *next(unsigned option) override;
25
26     };
27 }
```

Referência do Arquivo MapasCampus.hpp

```
#include "menuMapas.hpp"
#include "menu.hpp"
#include <string>
#include <iostream>
```

Componentes

class media::ui::MapasCampus**Namespaces**

- namespace **media**
- namespace **media::ui**

MapasCampus.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menuMapas.hpp"
4 #include "menu.hpp"
5 #include <string>
6 #include <iostream>
7
8 namespace media::ui {
9     class MapasCampus : public Menu {
10     public:
12         MapasCampus();
13
16         Menu *next(unsigned option) override;
17     };
18
19
20 }
21
```

Referência do Arquivo MapasInterno.hpp

```
#include "menuMapas.hpp"
#include "menu.hpp"
#include <string>
#include <iostream>
```

Componentes

class media::ui::MapasInterno**Namespaces**

- namespace **media**
- namespace **media::ui**

MapasInterno.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menuMapas.hpp"
4 #include "menu.hpp"
5 #include <string>
6 #include <iostream>
7
8 namespace media::ui {
9     class MapasInterno : public Menu {
10     public:
12         MapasInterno();
13
16         Menu *next(unsigned option) override;
17     };
18
19
20 }
```

Referência do Arquivo menu.hpp

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

Componentes

class media::ui::Menu**Namespaces**

- namespace **media**
- namespace **media::ui**

menu.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4 #include <vector>
5 #include "usuario.hpp"
6
7 namespace media::ui{
8     class Menu{
9     public:
11         virtual ~Menu() = default;
12
14         virtual void render() const;
15
17         virtual Menu *next(unsigned option) = 0;
18
19     protected:
20         std::string _title = "Menu Principal";
21         std::vector<std::string> _options = {"0 - Sair"};
22     };
23 };
```

Referência do Arquivo menuCarteirinha.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuCarteirinha**Namespaces**

- namespace **media**
- namespace **media::ui**

menuCarteirinha.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui{
7     class MenuCarteirinha : public Menu{
8     public:
10         MenuCarteirinha();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuDepartamento.hpp

```
#include "menu.hpp"
#include "menuSalas.hpp"
#include <string>
#include <iostream>
```

Componentes

class media::ui::MenuDepartamento**Namespaces**

- namespace **media**
- namespace **media::ui**

menuDepartamento.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include "menuSalas.hpp"
5 #include <string>
6 #include <iostream>
7
8 namespace media::ui {
9     class MenuDepartamento : public Menu {
10     public:
12         MenuDepartamento();
13
16         Menu *next(unsigned option) override;
17     };
18
19
20 }
```

Referência do Arquivo menuEventos.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuEventos**Namespaces**

- namespace **media**
- namespace **media::ui**

menuEventos.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui{
7     class MenuEventos : public Menu{
8     public:
10         MenuEventos();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuGradeSemanal.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuGrade**Namespaces**

- namespace **media**
- namespace **media::ui**

menuGradeSemanal.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui{
7     class MenuGrade : public Menu{
8     public:
10         MenuGrade();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuMapas.hpp

```
#include "menu.hpp"
#include "MapasCampus.hpp"
#include "MapasInterno.hpp"
#include <string>
```

Componentes

class media::ui::MenuMapas**Namespaces**

- namespace **media**
- namespace **media::ui**

menuMapas.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include "MapasCampus.hpp"
5 #include "MapasInterno.hpp"
6 #include <string>
7
8 namespace media::ui{
9     class MenuMapas : public Menu{
10     public:
12         MenuMapas();
13
16         Menu *next(unsigned option) override;
17
18     };
19 }
```

Referência do Arquivo menuOnibus.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuOnibus**Namespaces**

- namespace **media**
- namespace **media::ui**

menuOnibus.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui {
7     class MenuOnibus : public Menu {
8     public:
10         MenuOnibus();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuSalas.hpp

```
#include "menu.hpp"
#include "menuUteis.hpp"
#include "menuDepartamento.hpp"
#include <string>
```

Componentes

class media::ui::MenuSalas**Namespaces**

- namespace **media**
- namespace **media::ui**

menuSalas.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include "menuUteis.hpp"
5 #include "menuDepartamento.hpp"
6 #include <string>
7
8 namespace media::ui {
9     class MenuSalas : public Menu {
10     public:
12         MenuSalas();
13
16         Menu *next(unsigned option) override;
17
18     };
19 }
```

Referência do Arquivo menuTransacoesRU.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuTransacoesRU**Namespaces**

- namespace **media**
- namespace **media::ui**

menuTransacoesRU.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui{
7     class MenuTransacoesRU : public Menu{
8     public:
10         MenuTransacoesRU();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuTransporte.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuTransporte**Namespaces**

- namespace **media**
- namespace **media::ui**

menuTransporte.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui {
7     class MenuTransporte : public Menu {
8     public:
10         MenuTransporte();
11
14         Menu *next(unsigned option) override;
15
16     };
17 }
```

Referência do Arquivo menuUteis.hpp

```
#include "menu.hpp"
#include "menuSalas.hpp"
#include <string>
#include <iostream>
```

Componentes

class media::ui::MenuUteis**Namespaces**

- namespace **media**
- namespace **media::ui**

menuUteis.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include "menuSalas.hpp"
5 #include <string>
6 #include <iostream>
7
8 namespace media::ui {
9     class MenuUteis : public Menu {
10     public:
12         MenuUteis();
13
16         Menu *next(unsigned option) override;
17     };
18
19
20 }
```

Referência do Arquivo menuVans.hpp

```
#include "menu.hpp"
#include <string>
```

Componentes

class media::ui::MenuVans**Namespaces**

- namespace **media**
- namespace **media::ui**

menuVans.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include "menu.hpp"
4 #include <string>
5
6 namespace media::ui {
7     class MenuVans : public Menu {
8     public:
10          MenuVans();
11
14          Menu *next(unsigned option) override;
15
16      };
17 }
```

Referência do Arquivo onibus.hpp

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
```

Componentes

class **Onibus**

onibus.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4 #include <iostream>
5 #include <fstream>
6 #include <vector>
7
8 #include <sstream>
9
10 class Onibus{
11 private:
12     struct HorarioOnibus{
13         std::string placa;
14         std::string tipo;
15         std::string marca;
16         std::string linha;
17         std::vector<std::string> horarios;
18     };
19
20     std::string linha;
21     std::string placa;
22     std::string marca;
23     std::string tipo;
24     std::string placaAtual;
25
26     std::vector<HorarioOnibus> horarios;  // Modificado para armazenar HorarioOnibus
27
28     void salvarOnibus() const;
29     void cadastrarOnibusNovo();
30     std::string formatarOnibus(const HorarioOnibus& horario) const;
31
32     void salvarOnibus(const std::vector<HorarioOnibus>& onibusCadastrados) const;
33     void cadastrarOnibusNovo(const std::vector<HorarioOnibus>& onibusCadastrados);
34
35 public:
36     void cadastrarOnibus();
37     void exibirInformacao() const;
38     void editarOnibus();
39     void exibirOnibusCadastrados() const;
40     void carregarOnibus();
41 };
```

Referência do Arquivo redirecionamento.hpp

```
#include <string>
```

Componentes

class **Redirecionamento**

redirecionamento.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4
5 class Redirecionamento{
6 public:
7     static void redirecionarLink(const std::string& link);
8 };
```

Referência do Arquivo rotina.hpp

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <iomanip>
#include <limits>
#include <fstream>
```

Componentes

class **Rotina**

rotina.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <vector>
6 #include <sstream>
7 #include <iomanip>
8 #include <limits>
9 #include <fstream>
10
11
12
13     //classe em que se cadastra uma disciplina no quadro semanal
14     class Rotina{
15
16         private:
18
19             // Definição da estrutura Horario
20             struct Horario{
21                 int  dia;
22                 int  hora;
23
24                 //construtor que inicaliza com 0
25                 Horario() : _dia(0), _hora(0) {}
26
27                 };
28
29             // Definição da estrutura Disciplina
30             struct Disciplina{
31                 std::string _nome;
32                 std::string _codigo;
33                 std::string  predio;
34                 std::string  sala;
35                 std::vector<Horario> horarios;
36
37                 //construtor que inicializa com 0
38                 Disciplina() : _nome("0"), _codigo("0"), _predio("0"), _sala("0") {}
39             };
40
41             //criação vetor do struct disciplina
42             std::vector<Disciplina> disciplinas;
43
44         public:
45
46
51             const std::vector<Disciplina>& getDisciplinas() const;
52
56             void cadastrarDisciplina();
57
61             void editar disciplina();
62
66             void informacoes_disciplina();
67
68
76             bool existeDisciplinaNoHorario(int dia, int hora) const;
77
82             void cadastrarHorarios(Horario &novoHorario);
83
88             void cadastrarDetalhes(Disciplina &novadisciplina);
89
94             void salvarDisciplinasEmArquivo(const std::string& nomeArquivo) const;
95
100             void carregarDisciplinasDeArquivo(const std::string& nomeArquivo);
101
102     };
```

Referência do Arquivo SalasUteis.hpp

```
#include <string>
#include <vector>
```

Componentes

- class **Sala**class **CadastroSala**

SalasUteis.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4 #include <vector>
5
6 class Sala {
7 public:
8     std::string nome;
9     std::string predio;
10     std::string numero;
11 };
12
13 class CadastroSala {
14 private:
15     std::vector<Sala> salas;
16
17 public:
18     // Métodos para manipulação do cadastro
19     void cadastrarSala(const std::string& nome, const std::string& predio, const
std::string& numero);
20     void exibirSalas() const;
21     void salvarCadastro() const;
22     void carregarCadastro();
23     void alterarSala(const std::string& nome);
24 };
```

Referência do Arquivo Saldo.hpp

```
#include <iostream>
#include "SimulaBancodedados.hpp"
#include "usuario.hpp"
```

Componentes

class **Saldo**

Saldo.hpp

Ir para a documentação desse arquivo.

```
1 #ifndef SALDO_HPP
2 #define SALDO_HPP
3 #include <iostream>
4 #include "SimulaBancodedados.hpp"
5 #include "usuario.hpp"
6
7 class Saldo{
8     private:
9         double saldo atual;
10
11
12     public:
13         Saldo();
14         Saldo(long int _matricula, BancoDeDados &banco);
15         void diminuir saldo(int  fump);
16         void adicionar saldo(double valor);
17         double retornar_saldo_atual() const;
18
19 };
20
21 #endif
```

Referência do Arquivo SimulaBancodedados.hpp

```
#include <iostream>
#include <unordered_map>
#include "usuario.hpp"
```

Componentes

class **BancoDeDados**

SimulaBancodedados.hpp

Ir para a documentação desse arquivo.

```
1 #ifndef SIMULABANCODEDADOS
2 #define SIMULABANCODEDADOS
3 #include <iostream>
4 #include <unordered map>
5 #include "usuario.hpp"
6
7 class BancoDeDados{
8 private:
9     std::unordered map<int, double> saldos iniciais; /* Simulação de um banco de dados
com ID de usuário no formato de
10     número de matrícula e saldo inicial */
11
12 public:
13     BancoDeDados();
14     double recupera saldo inicial(long int  matricula) const;
15 };
16
17
18 #endif
```

Referência do Arquivo Transacoes.hpp

```
#include <iostream>
#include "./usuario.hpp"
#include "Saldo.hpp"
#include <hpdf.h>
```

Componentes

class **Transacoes**

Transacoes.hpp

Ir para a documentação desse arquivo.

```
1 #ifndef TRANSACOES
2 #define TRANSACOES
3 #include <iostream>
4 #include "./usuario.hpp"
5 #include "Saldo.hpp"
6 #include <hpdf.h>
7
8 /* @brief: Nesta classe busca-se efetuar funcionalidades da transacao em si, como
confirmar pagamento, deposito
9 e gerar e ler documentos para pagamento.  */
10
11 class Transacoes {
12     private:
13         bool confirma_pagamento;
14         bool confirma deposito;
15     public:
16         Transacoes();
17         void set_pagamento(bool pagamento);
18         void set_deposito(bool deposito);
19         void pagamento();
20         void deposito(double valor);
21         void consultar saldo();
22         void preenche_pdf(HPDF_Page page);
23         void gerar_GRU(const char* guia);
24         void ler_qrcode();
25 };
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42 #endif
```

Referência do Arquivo usuario.hpp

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

Componentes

class **Usuario**

usuario.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <vector>
6
7 class Usuario{
8 private:
9     long int  matricula;
10     std::string  nome;
11     std::string  curso;
12     std::string  email;
13     int _fump;
14     long long int _cpf;
15     std::string  endereco;
16
17 public:
19     Usuario();
20
22     long int getMatricula() const;
23     std::string getNome() const;
24     std::string getCurso() const;
25     std::string getEmail() const;
26     int getNivelFump() const;
27     long long int getCPF() const;
28     std::string getEndereco() const;
29
31     void setMatricula(long int matricula);
32     void setNome(const std::string& nome);
33     void setCurso(const std::string& curso);
34     void setEmail(const std::string& email);
35     void setNivelFump(int fump);
36     void setCPF(long long int cpf);
37     void setEndereco(const std::string& endereco);
38 };
```

Referência do Arquivo validação.hpp

```
#include <iostream>
#include <string>
#include <chrono>
#include <vector>
#include <sstream>
#include <iomanip>
```

Componentes

class **Validacao**

validação.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <iostream>
4 #include <string>
5 #include <chrono>
6 #include <vector>
7 #include <sstream>
8 #include <iomanip>
9
10
11     class Validacao{
12         public:
13     int obterNumeroInteiro();
14
20                 char validarSN(char escolha);
21
27                 int validarNumero(int escolha);
28
34                 int validarDia(int escolha);
35
41                 int validarHorario(int escolha);
42
48                 std::string transformarEmDia(int dia);
49
55                 std::string transformarEmHora(int hora);
56
62                 bool validarFormatoData(const std::string& data) const;
63
69                 bool validarFormatoHora(const std::string& hora) const;
70
71
72
73
74
75     };
```

Referência do Arquivo van.hpp

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
```

Componentes

class **Van**

van.hpp

Ir para a documentação desse arquivo.

```
1 #pragma once
2
3 #include <string>
4 #include <iostream>
5 #include <fstream>
6 #include <vector>
7 #include <sstream>
8
9 class Van{
10 private:
11     struct HorarioVan{
12         std::string placa;
13         std::string tipo;
14         std::string marca;
15         std::string linha;
16         std::vector<std::string> horarios;
17     };
18
19     std::string linha;
20     std::string placa;
21     std::string marca;
22     std::string tipo;
23
24     void salvarVans(const std::vector<HorarioVan>& vansCadastradas) const;
25     void cadastrarVanNova(const std::vector<HorarioVan>& vansCadastradas);
26
27 public:
28     void cadastrarVan();
29     void exibirInformacao() const;
30     void editarVan();
31     void exibirVansCadastradas() const;
32     void carregarVans();
33 };
```

Sumário

INDEX