Concordo

Generated by Doxygen 1.9.7

Chapter 1

README

Linguagens de Programação 1 Progamming Languages 1 Discord Clone - Concordo @subsection autotoc_md0 Projeto / Project Project#### O *projeto* consiste na criação de um clone do Discord utilizando o terminal, sem interface gráfica e sem acesso de rede..#### The *project* consists of creating a clone of Discord using the terminal without a graphical interface and without network access. @subsection autotoc_md1 Como rodar o projeto Clone o projeto utilizando o git e entre no diretório: @icode{bash} git clone https://github.com/edurs2602/discord cd discord @endicode Agora, para compilar o Concordo, siga os seguintes passos: @icode{bash} mkdir build cd build cmake .. cmake --build . @endicode Dessa forma, o Concordo está pronto para rodar: @icode{bash} ./Concordo @endicode Para rodar o Concordo com o script de teste atualizado para a parte 2, utilize o seguinte código: @icode{bash} ./Concordo < ../script-test.txt @endicode @subsection autotoc_md2 Technologies Technologies C++ @paragraph autotoc_md3 O projeto utiliza-se somente da linguagem C++. // The project uses only the C++ language. @subsection autotoc_md4 Funcionalidades O Concordo, apresenta as seguintes funcionalidades: funcionalidadesSair do sistemasistemaCriar usuáriousuárioFazer loginloginDeslogarDeslogarInterações com Servidores:ServidoresCriar servidorservidorMudar a descrição do servidorservidorSetar código de convite do servidorservidorListar servidoresservidoresRemover servidorservidorEntrar em um servidorservidorSair de um servidorservidorListar participantes do servidor Funcionalidades da Parte 2: :Listar CanaisCanaisCriar CanaisCanaisEntrar em algum CanalCanalSair de algum CanalCanalMandar alguma mensagem de textotextoListar as mensagens do Servidor @subsection autotoc_md5 Informações do Aluno Colaborador: @edurs2602

Luis Eduardo - 20220028973

:phone: *C O N T A T O S / C O N T A C T S* :phone:

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

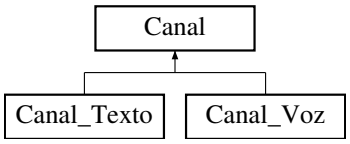
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Canal Class Reference

Inheritance diagram for Canal:



Public Member Functions

- Canal ()
- virtual ~Canal ()
- Canal (std::string nome)
- void setNome (std::string nome)
- std::string getNome ()
- bool operator== (const Canal &other) const

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Canal() [1/2]

```
Canal::Canal ( )
```

Construtor padrão da classe Canal.

5.1.1.2 ~Canal()

```
Canal::~Canal ( )  [virtual]
```

Destrutor da classe Canal.

5.1.1.3 Canal() [2/2]

```
Canal::Canal (
            std::string nome )
```

Construtor da classe Canal que recebe o nome do canal.

Parameters

nome	O nome do canal.

5.1.2 Member Function Documentation

5.1.2.1 getNome()

```
std::string Canal::getNome ( )
```

Obtém o nome do canal.

Returns

O nome do canal.

5.1.2.2 operator==()

```
bool Canal::operator== (
            const Canal & other ) const
```

Sobrecarga do operador de igualdade para comparar dois objetos Canal.

Parameters

other	O objeto Canal a ser comparado.

Returns

True se os objetos têm o mesmo nome, False caso contrário.

5.1.2.3 setNome()

```
void Canal::setNome (
            std::string nome )
```

Define o nome do canal.

Parameters

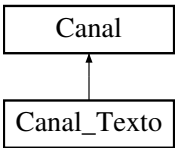
nome	O nome do canal.

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

- include/canal.h
- src/canal.cpp

5.2 Canal_Texto Class Reference

Inheritance diagram for Canal_Texto:



Public Member Functions

- Canal_Texto ()
- ~Canal_Texto ()

Public Member Functions inherited from Canal

- Canal ()
- virtual ~Canal ()
- Canal (std::string nome)
- void setNome (std::string nome)
- std::string getNome ()
- bool operator== (const Canal &other) const

Public Attributes

- std::vector< Mensagem > **texto**

5.2.1 Constructor & Destructor Documentation

5.2.1.1 Canal_Texto()

```
Canal_Texto::Canal_Texto ( )
```

Construtor padrão da classe Canal_Texto.

5.2.1.2 ~Canal_Texto()

```
Canal_Texto::~Canal_Texto ( )
```

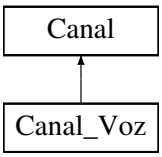
Destrutor da classe Canal_Texto.

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

- include/canal_texto.h
- src/canal_texto.cpp

5.3 Canal_Voz Class Reference

Inheritance diagram for Canal_Voz:



Public Member Functions

- Canal_Voz ()
- ~Canal_Voz ()
- void setUltima (Mensagem x)
- std::string getUltima ()

Public Member Functions inherited from Canal

- Canal ()
- virtual ~Canal ()
- Canal (std::string nome)
- void setNome (std::string nome)
- std::string getNome ()
- bool operator== (const Canal &other) const

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Canal_Voz()

```
Canal_Voz::Canal_Voz ( )
```

Construtor padrão da classe Canal_Voz.

5.3.1.2 ~Canal_Voz()

```
Canal_Voz::~Canal_Voz ( )
```

Destrutor da classe Canal_Voz.

5.3.2 Member Function Documentation

5.3.2.1 getUltima()

```
std::string Canal_Voz::getUltima ( )
```

Obtém o conteúdo da última mensagem enviada no canal de voz.

Returns

O conteúdo da última mensagem.

5.3.2.2 setUltima()

```
void Canal_Voz::setUltima (
            Mensagem x )
```

Define a última mensagem enviada no canal de voz.

Parameters

x	A última mensagem enviada.

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

- include/canal_voz.h
- src/canal_voz.cpp

5.4 Mensagem Class Reference

`#include <mensagem.h>`

Public Member Functions

- Mensagem ()
- ~Mensagem ()
- Mensagem (std::string dataHora, int enviadaPor, std::string conteudo)
- void setEnviadoPor (int id)
- void setDataHora (std::string dataHora)
- void setConteudo (std::string conteudo)
- int getEnviadoPor () const
- std::string getDataHora () const
- std::string getConteudo () const
- bool operator== (Mensagem &mensagem)

5.4.1 Detailed Description

Classe que representa uma mensagem.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Mensagem() [1/2]

```
Mensagem::Mensagem ( )
```

Construtor padrão da classe Mensagem.

5.4.2.2 ~Mensagem()

```
Mensagem::~Mensagem ( )
```

Destrutor da classe Mensagem.

5.4.2.3 Mensagem() [2/2]

```
Mensagem::Mensagem (
            std::string dataHora,
            int enviadaPor,
            std::string conteudo )
```

Construtor da classe Mensagem com parâmetros.

Parameters

dataHora	A data e hora da mensagem.
enviadaPor	O ID do usuário que enviou a mensagem.
conteudo	O conteúdo da mensagem.

Construtor da classe Mensagem que recebe a data e hora, o remetente e o conteúdo da mensagem.

Parameters

dataHora	A data e hora da mensagem.
enviadaPor	O ID do remetente da mensagem.
conteudo	O conteúdo da mensagem.

5.4.3 Member Function Documentation

5.4.3.1 getConteudo()

```
std::string Mensagem::getConteudo ( ) const
```

Obtém o conteúdo da mensagem.

Returns

O conteúdo da mensagem.

5.4.3.2 getDataHora()

```
std::string Mensagem::getDataHora ( ) const
```

Obtém a data e hora da mensagem.

Returns

A data e hora da mensagem.

5.4.3.3 getEnviadoPor()

```
int Mensagem::getEnviadoPor ( ) const
```

Obtém o ID do usuário que enviou a mensagem.

Returns

O ID do usuário.

Obtém o ID do remetente da mensagem.

Returns

O ID do remetente.

5.4.3.4 operator==()

```
bool Mensagem::operator== (
            Mensagem & mensagem )
```

Sobrecarga do operador de igualdade. Verifica se duas mensagens são iguais.

Parameters

mensagem	A mensagem a ser comparada.

Returns

True se as mensagens são iguais, False caso contrário.

5.4.3.5 setConteudo()

```
void Mensagem::setConteudo (
            std::string conteudo )
```

Define o conteúdo da mensagem.

Parameters

conteudo	O conteúdo da mensagem.

5.4.3.6 setDataHora()

```
void Mensagem::setDataHora (
            std::string dataHora )
```

Define a data e hora da mensagem.

Parameters

dataHora	A data e hora da mensagem.

5.4.3.7 setEnviadoPor()

```
void Mensagem::setEnviadoPor (
            int id )
```

Define o ID do usuário que enviou a mensagem.

Parameters

id	O ID do usuário.

Define o remetente da mensagem.

Parameters

id	O ID do remetente.

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

- include/mensagem.h
- src/mensagem.cpp

5.5 Parser Class Reference

`#include <parser.h>`

Public Member Functions

- bool parse (std::string entrada)
- void print ()
- std::string getComando ()
- std::string getArg (int index)
- std::string getArgsEspace ()
- int qtdArgs (std::string comando)

5.5.1 Detailed Description

Classe responsável por realizar o parsing de comandos.

5.5.2 Member Function Documentation

5.5.2.1 getArg()

```
std::string Parser::getArg (
            int index )
```

Obtém o argumento na posição especificada.

Parameters

index	O índice do argumento desejado.

Returns

O argumento na posição especificada.

5.5.2.2 getArgsEspace()

```
std::string Parser::getArgsEspace ( )
```

Obtém os argumentos concatenados separados por espaço.

Returns

Os argumentos concatenados separados por espaço.

Obtém os argumentos concatenados por espaços.

Returns

Os argumentos concatenados por espaços.

5.5.2.3 getComando()

```
std::string Parser::getComando ( )
```

Obtém o comando extraído da entrada.

Returns

O comando extraído.

Obtém o comando extraído da análise da entrada.

Returns

O comando da entrada.

5.5.2.4 parse()

```
bool Parser::parse (
            std::string entrada )
```

Realiza o parsing da entrada fornecida.

Parameters

entrada	A entrada a ser parseada.

Returns

True se o parsing foi bem-sucedido, False caso contrário.

Faz o parsing da entrada fornecida e armazena as informações relevantes.

Parameters

entrada	A entrada a ser analisada.

Returns

true se a análise for bem-sucedida, false caso contrário.

5.5.2.5 print()

```
void Parser::print ( )
```

Imprime as informações do parser.

5.5.2.6 qtdArgs()

```
int Parser::qtdArgs (
            std::string comando )
```

Obtém a quantidade de argumentos para um determinado comando.

Parameters

comando	O comando desejado.

Returns

A quantidade de argumentos necessários para o comando.

Obtém a quantidade de argumentos necessários para um determinado comando.

Parameters

comando	O comando a ser verificado.

Returns

A quantidade de argumentos necessários para o comando.

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

- include/parser.h
- src/parser.cpp

5.6 Servidor Class Reference

```
#include <servidor.h>
```

Public Member Functions

- Servidor ()
- ~Servidor ()
- Servidor (int idDono, std::string nome, std::string descricao)
- void setNome (std::string nome)
- void setDescricao (std::string descricao)
- void setCodigo (std::string codigo)
- int getDono ()
- std::string getNome ()
- std::string getDesc ()
- std::string getCodigo ()
- std::vector< int > getParticipantesId ()

Public Attributes

- std::vector< int > **participantesID**
- std::vector< Canal * > **canais**

5.6.1 Detailed Description

Classe que representa um servidor.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Servidor() [1/2]

```
Servidor::Servidor ( )
```

Construtor padrão da classe Servidor.

5.6.2.2 ~Servidor()

```
Servidor::~Servidor ( )
```

Destrutor da classe Servidor.

Destrutor da classe Servidor. Libera a memória alocada para os objetos do tipo Canal.

5.6.2.3 Servidor() [2/2]

```
Servidor::Servidor (
            int idDono,
            std::string nome,
            std::string descricao )
```

Construtor da classe Servidor com parâmetros.

Parameters

idDono	O ID do dono do servidor.
nome	O nome do servidor.
descricao	A descrição do servidor.

Construtor da classe Servidor.

Parameters

idDono	O ID do dono do servidor.
nome	O nome do servidor.
descricao	A descrição do servidor.

5.6.3 Member Function Documentation

5.6.3.1 getCodigo()

```
std::string Servidor::getCodigo ( )
```

Obtém o código de convite do servidor.

Returns

O código de convite do servidor.

5.6.3.2 getDesc()

```
std::string Servidor::getDesc ( )
```

Obtém a descrição do servidor.

Returns

A descrição do servidor.

5.6.3.3 getDono()

```
int Servidor::getDono ( )
```

Obtém o ID do dono do servidor.

Returns

O ID do dono do servidor.

5.6.3.4 getNome()

```
std::string Servidor::getNome ( )
```

Obtém o nome do servidor.

Returns

O nome do servidor.

5.6.3.5 getParticipantesId()

```
std::vector< int > Servidor::getParticipantesId ( )
```

Obtém os IDs dos participantes do servidor.

Returns

Um vetor contendo os IDs dos participantes.

Obtém os IDs dos participantes do servidor.

Returns

Um vetor contendo os IDs dos participantes do servidor.

5.6.3.6 setCodigo()

```
void Servidor::setCodigo (
            std::string codigo )
```

Define o código de convite do servidor.

Parameters

codigo	O código de convite do servidor.

Define o código de convite do servidor.

Parameters

codigo	O novo código de convite do servidor.

5.6.3.7 setDescricao()

```
void Servidor::setDescricao (
            std::string descricao )
```

Define a descrição do servidor.

Parameters

descricao	A descrição do servidor.

Define a descrição do servidor.

Parameters

descricao	A nova descrição do servidor.

5.6.3.8 setNome()

```
void Servidor::setNome (
            std::string nome )
```

Define o nome do servidor.

Parameters

nome	O nome do servidor.

Define o nome do servidor.

Parameters

nome	O novo nome do servidor.

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

- include/servidor.h
- src/servidor.cpp

5.7 Sistema Class Reference

```
#include <sistema.h>
```

Public Member Functions

- Sistema ()=default
- ~Sistema ()
- std::vector< Usuario * > downloadUsuarios ()
- std::vector< Servidor * > downloadServidores ()
- std::string timeMessage ()
- bool procurarEmail (const std::string &email)

- bool procurarServidor (const std::string &nome)
- bool procurarCanal (const std::string &nome, const std::string &tipo)
- void iniciar ()
- bool quit ()
- bool create_user (const std::string email, const std::string senha, const std::string nome)
- void login (const std::string email, const std::string senha)
- void disconnect ()
- void create_server (const std::string nome)
- void set_server_desc (const std::string nome, const std::string desc)
- void set_server_invite_code (const std::string nome, const std::string convite)
- void list_servers ()
- void remove_server (const std::string nome)
- void enter_server (const std::string nome, const std::string convite)
- void leave_server ()
- void list_participants ()
- void create_channel (const std::string nome, const std::string tipo)
- void list_channels ()
- void enter_channel (const std::string nome)
- void leave_channel ()
- void send_message (const std::string mensagem)
- void list_messages ()

Static Public Attributes

- static std::string **getNomeServidorAtual**
- static int **getIdUsuarioAtual**
- static int **getIdCanalAtual**

5.7.1 Detailed Description

Classe que representa o sistema.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Sistema()

```
Sistema::Sistema ( )  [default]
```

Construtor padrão da classe Sistema.

5.7.2.2 ~Sistema()

```
Sistema::~Sistema ( )
```

Destrutor da classe Sistema.

Destrói o objeto Sistema, liberando a memória alocada para os servidores e usuários.

5.7.3 Member Function Documentation

5.7.3.1 create_channel()

```
void Sistema::create_channel (
            const std::string nome,
            const std::string tipo )
```

Cria um novo canal no servidor atual.

Parameters

nome	O nome do canal.
tipo	O tipo do canal.

Método para criar os canais no servidor.

Parameters

nome	O nome do canal para criar.
tipo	O tipo de canal para se criar.

5.7.3.2 create_server()

```
void Sistema::create_server (
            const std::string nome )
```

Cria um novo servidor.

Parameters

nome	O nome do servidor.

Cria um novo servidor com o nome fornecido.

Parameters

nome	o nome do servidor.

5.7.3.3 create_user()

```
bool Sistema::create_user (
            const std::string email,
            const std::string senha,
            const std::string nome )
```

Cria um novo usuário.

Parameters

email	O email do usuário.
senha	A senha do usuário.
nome	O nome do usuário.

Returns

True se o usuário foi criado com sucesso, False caso contrário.

Cria um novo usuário com o email, senha e nome fornecidos.

Parameters

email	o email do usuário.
senha	a senha do usuário.
nome	o nome do usuário.

Returns

true se o usuário foi criado com sucesso, false caso contrário.

5.7.3.4 disconnect()

```
void Sistema::disconnect ( )
```

Desconecta o usuário atual.

5.7.3.5 downloadServidores()

```
std::vector< Servidor * > Sistema::downloadServidores ( )
```

Faz o download dos servidores.

Returns

Um vetor de ponteiros para servidores.

Retorna um vetor com todos os servidores do sistema.

Returns

um vetor com todos os servidores do sistema.

5.7.3.6 downloadUsuarios()

```
std::vector< Usuario * > Sistema::downloadUsuarios ( )
```

Faz o download dos usuários.

Returns

Um vetor de ponteiros para usuários.

Retorna um vetor com todos os usuários do sistema.

Returns

um vetor com todos os usuários do sistema.

5.7.3.7 enter_channel()

```
void Sistema::enter_channel (
            const std::string nome )
```

Entra em um canal do servidor atual.

Parameters

nome	O nome do canal.

Método para entrar no canal disponiveis no servidor.

Parameters

nome	O nome do canal para entrar nele se existir.

5.7.3.8 enter_server()

```
void Sistema::enter_server (
            const std::string nome,
            const std::string convite )
```

Entra em um servidor.

Parameters

nome	O nome do servidor.
convite	O código de convite do servidor.

Método para entrar em um servidor.

Parameters

nome	O nome do servidor para entrar.
convite	O código de convite para o servidor (opcional).

5.7.3.9 iniciar()

```
void Sistema::iniciar ( )
```

Inicia o sistema.

Método para iniciar o Parser do sistema. e pegar os comandos e argumentos passados pelo usuario.

5.7.3.10 leave_channel()

```
void Sistema::leave_channel ( )
```

Sai do canal atual.

Método para sair do canal.

5.7.3.11 leave_server()

```
void Sistema::leave_server ( )
```

Sai do servidor atual.

Método para sair do servidor atual.

5.7.3.12 list_channels()

```
void Sistema::list_channels ( )
```

Lista os canais do servidor atual.

Método para listar todos os canais do servidor.

5.7.3.13 list_messages()

```
void Sistema::list_messages ( )
```

Lista as mensagens do canal atual.

Método para listar as mensagens do servidor.

5.7.3.14 list_participants()

```
void Sistema::list_participants ( )
```

Lista os participantes do servidor atual.

5.7.3.15 list_servers()

```
void Sistema::list_servers ( )
```

Lista os servidores disponíveis.

Lista os servidores disponíveis no sistema.

5.7.3.16 login()

```
void Sistema::login (
            const std::string email,
            const std::string senha )
```

Faz o login de um usuário.

Parameters

email	O email do usuário.
senha	A senha do usuário.

Realiza o login de um usuário com o email e senha fornecidos.

Parameters

email	o email do usuário.
senha	a senha do usuário.

5.7.3.17 procurarCanal()

```
bool Sistema::procurarCanal (
            const std::string & nome,
            const std::string & tipo )
```

Verifica se um canal já existe em um servidor.

Parameters

nome	O nome do canal a ser verificado.
tipo	O tipo do canal a ser verificado.

Returns

True se o canal já existe, False caso contrário.

Verifica se um determinado canal de um servidor está cadastrado no sistema.

Parameters

nome	o nome do canal a ser verificado.
tipo	o tipo do canal (¨texto¨ ou ¨voz¨).

Returns

true se o canal estiver cadastrado, false caso contrário.

5.7.3.18 procurarEmail()

```
bool Sistema::procurarEmail (
            const std::string & email )
```

Verifica se um email já está cadastrado.

Parameters

email	O email a ser verificado.

Returns

True se o email já está cadastrado, False caso contrário.

Verifica se um determinado email já está cadastrado no sistema.

Parameters

email	o email a ser verificado.

Returns

true se o email já estiver cadastrado, false caso contrário.

5.7.3.19 procurarServidor()

```
bool Sistema::procurarServidor (
            const std::string & nome )
```

Verifica se um servidor já existe.

Parameters

nome	O nome do servidor a ser verificado.

Returns

True se o servidor já existe, False caso contrário.

Verifica se um determinado servidor já está cadastrado no sistema.

Parameters

nome	o nome do servidor a ser verificado.

Returns

true se o servidor já estiver cadastrado, false caso contrário.

5.7.3.20 quit()

```
bool Sistema::quit ( )
```

Encerra o sistema.

Returns

True se o sistema deve ser encerrado, False caso contrário.

Finaliza o sistema Concordo.

Returns

true, indicando que o sistema deve ser finalizado.

5.7.3.21 remove_server()

```
void Sistema::remove_server (
            const std::string nome )
```

Remove um servidor.

Parameters

nome	O nome do servidor a ser removido.

Remove um servidor do sistema.

5.7.3.22 send_message()

```
void Sistema::send_message (
            const std::string mensagem )
```

Envia uma mensagem para o canal atual.

Parameters

mensagem	A mensagem a ser enviada.

5.7.3.23 set_server_desc()

```
void Sistema::set_server_desc (
            const std::string nome,
            const std::string desc )
```

Define a descrição de um servidor.

Parameters

nome	O nome do servidor.
desc	A descrição do servidor.

Define a descrição de um servidor existente.

Parameters

nome	o nome do servidor.
desc	a nova descrição do servidor.

5.7.3.24 set_server_invite_code()

```
void Sistema::set_server_invite_code (
            const std::string nome,
            const std::string codigo )
```

Define o código de convite de um servidor.

Parameters

nome	O nome do servidor.
convite	O código de convite do servidor.

Define o código de convite de um servidor existente.

Parameters

nome	o nome do servidor.
codigo	o novo código de convite do servidor.

5.7.3.25 timeMessage()

```
std::string Sistema::timeMessage ( )
```

Obtém a hora atual para uso nas mensagens.

Returns

Uma string contendo a hora atual.

Retorna uma string com a data e hora atuais no formato ¨<dd/mm/yyyy - hh:mm:ss>¨.

Returns

uma string com a data e hora atuais.

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

- include/sistema.h
- src/sistema.cpp

5.8 Usuario Class Reference

`#include <usuario.h>`

Public Member Functions

- Usuario ()
- ~Usuario ()
- Usuario (std::string email, std::string senha, std::string nome)
- void setId (int id)
- void setNome (std::string nome)
- void setEmail (std::string email)
- void setSenha (std::string senha)
- int getId () const
- std::string getNome () const
- std::string getEmail () const
- std::string getSenha () const
- bool operator== (const Usuario &usuario) const

5.8.1 Detailed Description

Classe que representa um usuário.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Usuario() [1/2]

```
Usuario::Usuario ( )
```

Construtor padrão da classe Usuario.

5.8.2.2 ~Usuario()

```
Usuario::~Usuario ( )
```

Destrutor da classe Usuario.

5.8.2.3 Usuario() [2/2]

```
Usuario::Usuario (
            std::string email,
            std::string senha,
            std::string nome )
```

Construtor da classe Usuario que recebe o email, senha e nome.

Parameters

email	O email do usuário.
senha	A senha do usuário.
nome	O nome do usuário.

Construtor da classe Usuario.

Parameters

email	O email do usuário.
senha	A senha do usuário.
nome	O nome do usuário.

5.8.3 Member Function Documentation

5.8.3.1 getEmail()

```
std::string Usuario::getEmail ( ) const
```

Obtém o email do usuário.

Returns

O email do usuário.

5.8.3.2 getId()

```
int Usuario::getId ( ) const
```

Obtém o ID do usuário.

Returns

O ID do usuário.

5.8.3.3 getNome()

```
std::string Usuario::getNome ( ) const
```

Obtém o nome do usuário.

Returns

O nome do usuário.

5.8.3.4 getSenha()

```
std::string Usuario::getSenha ( ) const
```

Obtém a senha do usuário.

Returns

A senha do usuário.

5.8.3.5 operator==()

```
bool Usuario::operator== (
            const Usuario & usuario ) const
```

Sobrecarga do operador de igualdade para comparar dois objetos Usuario.

Parameters

usuario	O objeto Usuario a ser comparado.

Returns

True se os objetos são iguais, False caso contrário.

Sobrecarga do operador de igualdade para comparar dois usuários.

Parameters

usuario	O usuário a ser comparado.

Returns

true se os usuários possuem o mesmo ID, false caso contrário.

5.8.3.6 setEmail()

```
void Usuario::setEmail (
            std::string email )
```

Define o email do usuário.

Parameters

email	O email do usuário.

Define o email do usuário.

Parameters

email	O novo email do usuário.

5.8.3.7 setId()

```
void Usuario::setId (
            int id )
```

Define o ID do usuário.

Parameters

id	O ID do usuário.

Define o ID do usuário.

Parameters

id	O novo ID do usuário.

5.8.3.8 setNome()

```
void Usuario::setNome (
            std::string nome )
```

Define o nome do usuário.

Parameters

nome	O nome do usuário.

Define o nome do usuário.

Parameters

nome	O novo nome do usuário.

5.8.3.9 setSenha()

```
void Usuario::setSenha (
            std::string senha )
```

Define a senha do usuário.

Parameters

senha	A senha do usuário.

Define a senha do usuário.

Parameters

senha	A nova senha do usuário.

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

- include/usuario.h
- src/usuario.cpp

Chapter 6

File Documentation

6.1 canal.h

```
00001 #ifndef CANAL_H
00002 #define CANAL_H
00003
00004 #include <iostream>
00005
00006 /*
00007  * Classe Canal
00008  * com um atributo privado chamado de nome do tipo string
00009  * com funções construtura e destrutora
00010  * função que seta o nome do Canal
00011  * outra função que retorna o nome do canal
00012  * e uma sobrecarga de operador
00013  */
00014
00015 class Canal {
00016 private:
00017   std::string nome;
00018
00019 public:
00020   Canal();
00021   virtual ~Canal();
00022   Canal(std::string nome);
00023
00024   void setNome(std::string nome);
00025
00026   std::string getNome();
00027
00028   bool operator==(const Canal &other) const;
00029 };
00030
00031 #endif // !CANAL_H
```

6.2 canal_texto.h

```
00001 #ifndef CANAL_TEXTO_H
00002 #define CANAL_TEXTO_H
00003
00004 #include <iostream>
00005 #include <vector>
00006
00007 #include "canal.h"
00008 #include "mensagem.h"
00009
00010 /*
00011  * Classe Canal Texto que herda de Canal
00012  * com um vetor de Mensagem chamado texto
00013  * e uma função construtora e destrutura
00014  */
00015 class Canal_Texto : public Canal {
00016 public:
00017   std::vector<Mensagem> texto;
00018
00019   Canal_Texto();
00020   ~Canal_Texto();
00021 };
00022
00023 #endif
```

6.3 canal_voz.h

```
00001 #ifndef CANAL_VOZ_H
00002 #define CANAL_VOZ_H
00003
00004 #include "canal.h"
00005 #include "mensagem.h"
00006
00007 /*
00008  * Classe Canal Voz que herda de Canal
00009  * com um atributo privado de Mensagem chamado ultima
00010  * uma função construtora e destrutura
00011  * uma função que seta a ultima mensagem
00012  * e outra que retorna a ultima mensagem
00013  */
00014
00015 class Canal_Voz : public Canal {
00016 private:
00017   Mensagem ultima;
00018
00019 public:
00020   Canal_Voz();
00021   ~Canal_Voz();
00022
00023   void setUltima(Mensagem x);
00024
00025   std::string getUltima();
00026 };
00027
00028 #endif
```

6.4 constantes.h

```
00001 #ifndef CONSTANTES_H
00002 #define CONSTANTES_H
00003
00004 #include <iostream>
00005 #include <vector>
00006
00007 /*
00008  *  Arquivo de constantes
00009  */
00010
00011 namespace cte {
00012
00013 const std::string SAIR = "quit"; // Comando para sair do programa.
00014 const std::string CRIAR_USUARIO =
00015     "create-user";                 // Comando para criar um novo usuário.
00016 const std::string LOGIN = "login"; // Comando para fazer login.
00017 const std::string DESCONECTAR =
00018     "disconnect"; // Comando para desconectar do servidor.
00019 const std::string CRIAR_SERVIDOR =
00020     "create-server"; // Comando para criar um novo servidor.
00021 const std::string MUDAR_DESCRICAO_SERVIDOR =
00022     "set-server-desc"; // Comando para mudar a descrição de um servidor.
00023 const std::string MUDAR_CONVITE_SERVIDOR =
00024     "set-server-invite-code"; // Comando para mudar o código de convite de um
00025                               // servidor.
00026 const std::string LISTAR_SERVIDORES =
00027     "list-servers"; // Comando para listar os servidores disponíveis.
00028 const std::string REMOVER_SERVIDOR =
00029     "remove-server"; // Comando para remover um servidor.
00030 const std::string ENTRAR_SERVIDOR =
00031     "enter-server"; // Comando para entrar em um servidor.
00032 const std::string SAIR_SERVIDOR =
00033     "leave-server"; // Comando para sair de um servidor.
00034 const std::string LISTAR_PARTICIPANTES =
00035     "list-participants"; // Comando para listar os participantes de um servidor.
00036 const std::string LISTAR_CANAIS =
00037     "list-channels"; // Comando para listar os canais de um servidor.
00038 const std::string CRIAR_CANAL =
00039     "create-channel"; // Comando para criar um novo canal em um servidor.
00040 const std::string ENTRAR_CANAL =
00041     "enter-channel"; // Comando para entrar em um canal.
00042 const std::string SAIR_CANAL =
00043     "leave-channel"; // Comando para sair de um canal.
00044 const std::string ENVIAR_MENSAGEM =
00045     "send-message"; // Comando para enviar uma mensagem.
00046 const std::string LISTAR_MENSAGENS =
00047     "list-messages"; // Comando para listar as mensagens de um canal.
00048
00049 const std::vector<std::pair<std::string, int» commands_simple_args = {
00050
```

```
00051     {SAIR, 0},
00052     {CRIAR_USUARIO, 2},
00053     {LOGIN, 2},
00054     {DESCONECTAR, 0},
00055     {CRIAR_SERVIDOR, 1},
00056     {MUDAR_DESCRICAO_SERVIDOR, 1},
00057     {MUDAR_CONVITE_SERVIDOR, 1},
00058     {LISTAR_SERVIDORES, 0},
00059     {REMOVER_SERVIDOR, 1},
00060     {ENTRAR_SERVIDOR, 1},
00061     {SAIR_SERVIDOR, 0},
00062     {LISTAR_PARTICIPANTES, 0},
00063     {LISTAR_CANAIS, 0},
00064     {CRIAR_CANAL, 2},
00065     {ENTRAR_CANAL, 1},
00066     {SAIR_CANAL, 0},
00067     {ENVIAR_MENSAGEM, 1},
00068     {LISTAR_MENSAGENS, 0}
00069
00070 };
00071
00072 } // namespace cte
00073
00074 #endif // !
```

6.5 mensagem.h

```
00001 #ifndef MENSAGEM_H
00002 #define MENSAGEM_H
00003
00004 #include "usuario.h"
00005 #include <iostream>
00006
00011 class Mensagem {
00012 private:
00013   int enviadaPor;       // ID do usuário que enviou a mensagem.
00014   std::string dataHora; // Data e hora em que a mensagem foi enviada.
00015   std::string conteudo; // Conteúdo da mensagem.
00016
00017 public:
00021   Mensagem();
00022
00026   ~Mensagem();
00027
00034   Mensagem(std::string dataHora, int enviadaPor, std::string conteudo);
00035
00040   void setEnviadoPor(int id);
00041
00046   void setDataHora(std::string dataHora);
00047
00052   void setConteudo(std::string conteudo);
00053
00058   int getEnviadoPor() const;
00059
00064   std::string getDataHora() const;
00065
00070   std::string getConteudo() const;
00071
00078   bool operator==(Mensagem &mensagem);
00079 };
00080
00081 #endif // !MENSAGEM_H
```

6.6 parser.h

```
00001 #ifndef PARSER_H
00002 #define PARSER_H
00003
00004 #include <iostream>
00005 #include <vector>
00006
00011 class Parser {
00012 private:
00013   std::string comando;           // O comando extraído da entrada.
00014   std::vector<std::string> args; // Os argumentos extraídos da entrada.
00015   std::string argsEspace; // Os argumentos concatenados separados por espaço.
00016
00017 public:
00023   bool parse(std::string entrada);
```

```
00024
00028   void print();
00029
00034   std::string getComando();
00035
00041   std::string getArg(int index);
00042
00047   std::string getArgsEspace();
00048
00054   int qtdArgs(std::string comando);
00055 };
00056
00057 #endif // !PARSER_H
```

6.7 servidor.h

```
00001 #include "canal.h"
00002 #include <iostream>
00003 #include <memory>
00004 #include <string>
00005 #include <vector>
00006
00007 #ifndef SERVIDOR_H
00008 #define SERVIDOR_H
00009
00014 class Servidor {
00015 private:
00016   int idDono;                // ID do dono do servidor.
00017   std::string nome;          // Nome do servidor.
00018   std::string descricao;     // Descrição do servidor.
00019   std::string codigoConvite; // Código de convite do servidor.
00020
00021 public:
00025   Servidor();
00026
00030   ~Servidor();
00031
00038   Servidor(int idDono, std::string nome, std::string descricao);
00039
00040   std::vector<int>
00041       participantesID;         // Vetor de IDs dos participantes do servidor.
00042   std::vector<Canal *> canais; // Vetor de ponteiros para os canais do servidor.
00043
00048   void setNome(std::string nome);
00049
00054   void setDescricao(std::string descricao);
00055
00060   void setCodigo(std::string codigo);
00061
00066   int getDono();
00067
00072   std::string getNome();
00073
00078   std::string getDesc();
00079
00084   std::string getCodigo();
00085
00090   std::vector<int> getParticipantesId();
00091 };
00092
00093 #endif // !SERVIDOR_H
```

6.8 sistema.h

```
00001 #ifndef SISTEMA_H
00002 #define SISTEMA_H
00003
00004 #include <iostream>
00005 #include <vector>
00006
00007 #include "servidor.h"
00008 #include "usuario.h"
00009
00014 class Sistema {
00015 private:
00016   std::vector<Usuario *> usuarios;    // Vetor de ponteiros para usuários.
00017   std::vector<Servidor *> servidores; // Vetor de ponteiros para servidores.
00018   int idUsuarioAtual;                 // ID do usuário atual.
00019   std::string nomeCanalAtual;         // Nome do canal atual.
```

```
00020   std::string nomerServidorAtual;     // Nome do servidor atual.
00021
00022 public:
00026   Sistema() = default;
00027
00031   ~Sistema();
00032
00037   std::vector<Usuario *> downloadUsuarios();
00038
00043   std::vector<Servidor *> downloadServidores();
00044
00045   static std::string
00046       getNomeServidorAtual;     // Getter para o nome do servidor atual.
00047   static int getIdUsuarioAtual; // Getter para o ID do usuário atual.
00048   static int getIdCanalAtual;   // Getter para o ID do canal atual.
00049
00054   std::string timeMessage();
00055
00061   bool procurarEmail(const std::string &email);
00062
00068   bool procurarServidor(const std::string &nome);
00069
00076   bool procurarCanal(const std::string &nome, const std::string &tipo);
00077
00078   // funcionalidades pt 1
00079
00083   void iniciar();
00084
00089   bool quit();
00090
00098   bool create_user(const std::string email, const std::string senha,
00099                    const std::string nome);
00100
00106   void login(const std::string email, const std::string senha);
00107
00111   void disconnect();
00112
00117   void create_server(const std::string nome);
00118
00124   void set_server_desc(const std::string nome, const std::string desc);
00125
00131   void set_server_invite_code(const std::string nome,
00132                               const std::string convite);
00133
00137   void list_servers();
00138
00143   void remove_server(const std::string nome);
00144
00150   void enter_server(const std::string nome, const std::string convite);
00151
00155   void leave_server();
00156
00160   void list_participants();
00161
00162   // funcionalidades pt 2
00163
00169   void create_channel(const std::string nome, const std::string tipo);
00170
00174   void list_channels();
00175
00180   void enter_channel(const std::string nome);
00181
00185   void leave_channel();
00186
00191   void send_message(const std::string mensagem);
00192
00196   void list_messages();
00197 };
00198
00199 #endif // !DEBUG
```

6.9 usuario.h

```
00001 #ifndef USUARIO_H
00002 #define USUARIO_H
00003
00004 #include <iostream>
00005
00010 class Usuario {
00011 private:
00012   int id;            // ID do usuário.
00013   std::string nome;  // Nome do usuário.
00014   std::string email; // Email do usuário.
```

```
00015   std::string senha; // Senha do usuário.
00016
00017 public:
00021   Usuario();
00022
00026   ~Usuario();
00027
00034   Usuario(std::string email, std::string senha, std::string nome);
00035
00040   void setId(int id);
00041
00046   void setNome(std::string nome);
00047
00052   void setEmail(std::string email);
00053
00058   void setSenha(std::string senha);
00059
00064   int getId() const;
00065
00070   std::string getNome() const;
00071
00076   std::string getEmail() const;
00077
00082   std::string getSenha() const;
00083
00089   bool operator==(const Usuario &usuario) const;
00090 };
00091
00092 #endif // !USUARIO_H
```

Index