

Concordo

Generated by Doxygen 1.9.7

1 README	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Canal Class Reference	9
5.1.1 Constructor & Destructor Documentation	9
5.1.1.1 Canal() [1/2]	9
5.1.1.2 ~Canal()	9
5.1.1.3 Canal() [2/2]	9
5.1.2 Member Function Documentation	10
5.1.2.1 getNome()	10
5.1.2.2 operator==()	10
5.1.2.3 setNome()	10
5.2 Canal_Texto Class Reference	11
5.2.1 Constructor & Destructor Documentation	11
5.2.1.1 Canal_Texto()	11
5.2.1.2 ~Canal_Texto()	11
5.3 Canal_Voz Class Reference	12
5.3.1 Constructor & Destructor Documentation	12
5.3.1.1 Canal_Voz()	12
5.3.1.2 ~Canal_Voz()	12
5.3.2 Member Function Documentation	12
5.3.2.1 getUltima()	12
5.3.2.2 setUltima()	13
5.4 Mensagem Class Reference	13
5.4.1 Detailed Description	13
5.4.2 Constructor & Destructor Documentation	13
5.4.2.1 Mensagem() [1/2]	13
5.4.2.2 ~Mensagem()	14
5.4.2.3 Mensagem() [2/2]	14
5.4.3 Member Function Documentation	14
5.4.3.1 getConteudo()	14
5.4.3.2 getDataHora()	14
5.4.3.3 getEnviadoPor()	15
5.4.3.4 operator==()	15
5.4.3.5 setConteudo()	15

5.4.3.6 setDataHora()	15
5.4.3.7 setEnviadoPor()	17
5.5 Parser Class Reference	17
5.5.1 Detailed Description	17
5.5.2 Member Function Documentation	17
5.5.2.1 getArg()	17
5.5.2.2 getArgsEspace()	18
5.5.2.3 getComando()	18
5.5.2.4 parse()	18
5.5.2.5 print()	19
5.5.2.6 qtdArgs()	19
5.6 Servidor Class Reference	20
5.6.1 Detailed Description	20
5.6.2 Constructor & Destructor Documentation	20
5.6.2.1 Servidor() [1/2]	20
5.6.2.2 ~Servidor()	21
5.6.2.3 Servidor() [2/2]	21
5.6.3 Member Function Documentation	21
5.6.3.1 getCodigo()	21
5.6.3.2 getDesc()	21
5.6.3.3 getDono()	22
5.6.3.4 getNome()	22
5.6.3.5 getParticipantesId()	22
5.6.3.6 setCodigo()	22
5.6.3.7 setDescricao()	23
5.6.3.8 setNome()	23
5.7 Sistema Class Reference	24
5.7.1 Detailed Description	24
5.7.2 Constructor & Destructor Documentation	24
5.7.2.1 Sistema()	24
5.7.2.2 ~Sistema()	25
5.7.3 Member Function Documentation	25
5.7.3.1 create_channel()	25
5.7.3.2 create_server()	25
5.7.3.3 create_user()	26
5.7.3.4 disconnect()	26
5.7.3.5 download()	26
5.7.3.6 enter_channel()	26
5.7.3.7 enter_server()	27
5.7.3.8 iniciar()	27
5.7.3.9 leave_channel()	27
5.7.3.10 leave_server()	28

5.7.3.11 list_channels()	28
5.7.3.12 list_messages()	28
5.7.3.13 list_participants()	28
5.7.3.14 list_servers()	28
5.7.3.15 login()	28
5.7.3.16 procurarCanal()	29
5.7.3.17 procurarEmail()	29
5.7.3.18 procurarServidor()	30
5.7.3.19 quit()	30
5.7.3.20 remove_server()	31
5.7.3.21 send_message()	31
5.7.3.22 set_server_desc()	31
5.7.3.23 set_server_invite_code()	32
5.7.3.24 timeMessage()	32
5.7.3.25 upload()	32
5.8 Usuario Class Reference	33
5.8.1 Detailed Description	33
5.8.2 Constructor & Destructor Documentation	33
5.8.2.1 Usuario() [1/2]	33
5.8.2.2 ~Usuario()	33
5.8.2.3 Usuario() [2/2]	33
5.8.3 Member Function Documentation	34
5.8.3.1 getEmail()	34
5.8.3.2 getId()	34
5.8.3.3 getNome()	34
5.8.3.4 getSenha()	35
5.8.3.5 operator==(())	35
5.8.3.6 setEmail()	35
5.8.3.7 setId()	36
5.8.3.8 setNome()	36
5.8.3.9 setSenha()	36
6 File Documentation	39
6.1 canal.h	39
6.2 canal_texto.h	39
6.3 canal_voz.h	40
6.4 constantes.h	40
6.5 mensagem.h	41
6.6 parser.h	42
6.7 servidor.h	42
6.8 sistema.h	43
6.9 usuario.h	44

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 @endcode
```

Agora, para compilar o Concordo, siga os seguintes passos:

```
@icode{bash} mkdir build cd build cmake .. cmake --build . @endcode
```

Dessa forma, o Concordo está pronto para rodar: @icode{bash}

```
./Concordo @endcode Para rodar o Concordo com o script de teste  

atualizado para a parte 2, utilize o seguinte código: @icode{bash}
```

```
./Concordo < ../script-test.txt @endcode @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 usuáriuusuárioFazer

loginloginDeslogarDeslogarInterações com Servidores:ServidoresCriar
 servidorsservidorMudar a descrição do servidorsservidorSetar código de
 convite do servidorsservidorListar servidorsservidoresRemover
 servidorsservidorEntrar em um servidorsservidorSair de um

servidorsservidorListar participantes do servidor Funcionalidades da
 Parte 2: :Listar CanaisCanaisCriar CanaisCanaisEntrar em algum

CanalCanalSair de algum **Canal**CanalMandar alguma mensagem de
 textotextoListar as mensagens do **Servidor** @subsection autotoc_md5

Informações do Aluno Colaborador: **@edurs2602**

Luis Eduardo - 20220028973

Generated by Doxygen

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Canal	9
Canal_Texto	11
Canal_Voz	12
Mensagem	13
Parser	17
Servidor	20
Sistema	24
Usuario	33

Chapter 3

Class Index

3.1 Class List

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

Canal	9
Canal_Texto	11
Canal_Voz	12
Mensagem	13
Parser	17
Servidor	20
Sistema	24
Usuario	33

Chapter 4

File Index

4.1 File List

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

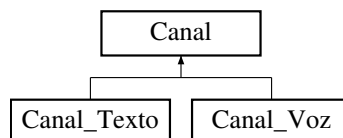
include/ canal.h	39
include/ canal_texto.h	39
include/ canal_voz.h	40
include/ constantes.h	40
include/ mensagem.h	41
include/ parser.h	42
include/ servidor.h	42
include/ sistema.h	43
include/ usuario.h	44

Chapter 5

Class Documentation

5.1 Canal Class Reference

Inheritance diagram for Canal:



Public Member Functions

- `Canal()`
- `virtual ~Canal()`
- `Canal(std::string nome)`
- `void setName(std::string nome)`
- `std::string getName()`
- `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

<i>nome</i>	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

<i>other</i>	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

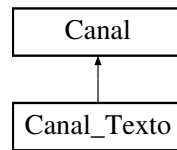
<i>nome</i>	O nome do canal.
-------------	------------------

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](#) ()
- **Canal_Texto** (std::string nome)
- void **adicionarMensagem** (const [Mensagem](#) &mensagem)

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 ( )
```

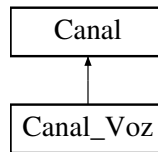
Destrutor da classe [Canal_Texto](#).

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](#) ()
- **Canal_Voz** (std::string nome)
- 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

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

Construtor da classe [Mensagem](#) que recebe a data e hora, o remetente e o conteúdo da mensagem.

Parameters

<i>dataHora</i>	A data e hora da mensagem.
<i>enviadaPor</i>	O ID do remetente da mensagem.
<i>conteudo</i>	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

<i>mensagem</i>	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

<i>conteudo</i>	O conteúdo da mensagem.
-----------------	-------------------------

5.4.3.6 setDataHora()

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

Define a data e hora da mensagem.

Parameters

<i>dataHora</i>	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

<i>id</i>	O ID do usuário.
-----------	------------------

Define o remetente da mensagem.

Parameters

<i>id</i>	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

<i>index</i>	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

<i>entrada</i>	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

<i>entrada</i>	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

<i>comando</i>	O comando desejado.
----------------	---------------------

Returns

A quantidade de argumentos necessários para o comando.

Obtém a quantidade de argumentos necessários para um determinado comando.

Parameters

<i>comando</i>	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)
- void [adicionarParticipante](#) (int id)
- void [adicionarCanal](#) ([Canal](#) *canal)
- int [getDono](#) ()
- std::string [getNome](#) ()
- std::string [getDesc](#) ()
- std::string [getCodigo](#) ()
- std::vector< int > [getParticipantesId](#) ()
- std::vector< [Canal](#) * > [getCanais](#) ()

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

<i>idDono</i>	O ID do dono do servidor.
<i>nome</i>	O nome do servidor.
<i>descricao</i>	A descrição do servidor.

Construtor da classe [Servidor](#).

Parameters

<i>idDono</i>	O ID do dono do servidor.
<i>nome</i>	O nome do servidor.
<i>descricao</i>	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

<i>codigo</i>	O código de convite do servidor.
---------------	----------------------------------

Define o código de convite do servidor.

Parameters

<i>codigo</i>	O novo código de convite do servidor.
---------------	---------------------------------------

5.6.3.7 setDescricao()

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

Define a descrição do servidor.

Parameters

<i>descricao</i>	A descrição do servidor.
------------------	--------------------------

Define a descrição do servidor.

Parameters

<i>descricao</i>	A nova descrição do servidor.
------------------	-------------------------------

5.6.3.8 setNome()

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

Define o nome do servidor.

Parameters

<i>nome</i>	O nome do servidor.
-------------	---------------------

Define o nome do servidor.

Parameters

<i>nome</i>	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](#) ()
- void [download](#) ()
- void [upload](#) ()
- 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](#).

Destroi 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

<i>nome</i>	O nome do canal.
<i>tipo</i>	O tipo do canal.

Método para criar os canais no servidor.

Parameters

<i>nome</i>	O nome do canal para criar.
<i>tipo</i>	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

<i>nome</i>	O nome do servidor.
-------------	---------------------

Cria um novo servidor com o nome fornecido.

Parameters

<i>nome</i>	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

<i>email</i>	O email do usuário.
<i>senha</i>	A senha do usuário.
<i>nome</i>	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

<i>email</i>	o email do usuário.
<i>senha</i>	a senha do usuário.
<i>nome</i>	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 download()

```
void Sistema::download ( )
```

Chama as duas funções: downloadUsuarios e downloadServidores.

Método para realizar o download de usuários e servidores do sistema.

5.7.3.6 enter_channel()

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

Entra em um canal do servidor atual.

Parameters

<i>nome</i>	O nome do canal.
-------------	------------------

Método para entrar no canal disponíveis no servidor.

Parameters

<i>nome</i>	O nome do canal para entrar nele se existir.
-------------	--

5.7.3.7 enter_server()

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

Entra em um servidor.

Parameters

<i>nome</i>	O nome do servidor.
<i>convite</i>	O código de convite do servidor.

Método para entrar em um servidor.

Parameters

<i>nome</i>	O nome do servidor para entrar.
<i>convite</i>	O código de convite para o servidor (opcional).

5.7.3.8 iniciar()

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

Inicia o sistema.

Método para iniciar o [Parser](#) do sistema. e pegar os comandos e argumentos passados pelo usuário.

5.7.3.9 leave_channel()

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

Sai do canal atual.

Método para sair do canal.

5.7.3.10 leave_server()

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

Sai do servidor atual.

Método para sair do servidor atual.

5.7.3.11 list_channels()

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

Lista os canais do servidor atual.

Método para listar todos os canais do servidor.

5.7.3.12 list_messages()

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

Lista as mensagens do canal atual.

Método para listar as mensagens do servidor.

5.7.3.13 list_participants()

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

Lista os participantes do servidor atual.

5.7.3.14 list_servers()

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

Lista os servidores disponíveis.

Lista os servidores disponíveis no sistema.

5.7.3.15 login()

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

Faz o login de um usuário.

Parameters

<i>email</i>	O email do usuário.
<i>senha</i>	A senha do usuário.

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

Parameters

<i>email</i>	o email do usuário.
<i>senha</i>	a senha do usuário.

5.7.3.16 procurarCanal()

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

Verifica se um canal já existe em um servidor.

Parameters

<i>nome</i>	O nome do canal a ser verificado.
<i>tipo</i>	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

<i>nome</i>	o nome do canal a ser verificado.
<i>tipo</i>	o tipo do canal ("texto" ou "voz").

Returns

true se o canal estiver cadastrado, false caso contrário.

5.7.3.17 procurarEmail()

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

Verifica se um email já está cadastrado.

Parameters

<i>email</i>	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

<i>email</i>	o email a ser verificado.
--------------	---------------------------

Returns

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

5.7.3.18 procurarServidor()

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

Verifica se um servidor já existe.

Parameters

<i>nome</i>	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

<i>nome</i>	o nome do servidor a ser verificado.
-------------	--------------------------------------

Returns

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

5.7.3.19 quit()

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

Encerra o sistema.

Returns

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

Finaliza o sistema Concorde.

Returns

true, indicando que o sistema deve ser finalizado.

5.7.3.20 remove_server()

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

Remove um servidor.

Parameters

<i>nome</i>	O nome do servidor a ser removido.
-------------	------------------------------------

Remove um servidor do sistema.

5.7.3.21 send_message()

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

Envia uma mensagem para o canal atual.

Parameters

<i>mensagem</i>	A mensagem a ser enviada.
-----------------	---------------------------

5.7.3.22 set_server_desc()

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

Define a descrição de um servidor.

Parameters

<i>nome</i>	O nome do servidor.
<i>desc</i>	A descrição do servidor.

Define a descrição de um servidor existente.

Parameters

<i>nome</i>	o nome do servidor.
<i>desc</i>	a nova descrição do servidor.

5.7.3.23 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

<i>nome</i>	O nome do servidor.
<i>convite</i>	O código de convite do servidor.

Define o código de convite de um servidor existente.

Parameters

<i>nome</i>	o nome do servidor.
<i>codigo</i>	o novo código de convite do servidor.

5.7.3.24 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.

5.7.3.25 upload()

```
void Sistema::upload ( )
```

Chama as duas funções: uploadUsuarios e uploadServidores.

Carrega os usuários e servidores do disco. Executa os métodos uploadUsuarios() e uploadServidores() para carregar os dados.

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

<i>email</i>	O email do usuário.
<i>senha</i>	A senha do usuário.
<i>nome</i>	O nome do usuário.

Construtor da classe [Usuario](#).

Parameters

<i>email</i>	O email do usuário.
<i>senha</i>	A senha do usuário.
<i>nome</i>	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

<i>usuario</i>	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

<i>usuario</i>	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

<i>email</i>	O email do usuário.
--------------	---------------------

Define o email do usuário.

Parameters

<i>email</i>	O novo email do usuário.
--------------	--------------------------

5.8.3.7 setId()

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

Define o ID do usuário.

Parameters

<i>id</i>	O ID do usuário.
-----------	------------------

Define o ID do usuário.

Parameters

<i>id</i>	O novo ID do usuário.
-----------	-----------------------

5.8.3.8 setNome()

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

Define o nome do usuário.

Parameters

<i>nome</i>	O nome do usuário.
-------------	--------------------

Define o nome do usuário.

Parameters

<i>nome</i>	O novo nome do usuário.
-------------	-------------------------

5.8.3.9 setSenha()

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

Define a senha do usuário.

Parameters

<i>senha</i>	A senha do usuário.
--------------	---------------------

Define a senha do usuário.

Parameters

<i>senha</i>	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 #include <vector>
00006
00007 /*
00008  * Classe Canal
00009  * com um atributo privado chamado de nome do tipo string
00010  * com funções construtora e destrutora
00011  * função que seta o nome do Canal
00012  * outra função que retorna o nome do canal
00013  * e uma sobrecarga de operador
00014  */
00015
00016 class Canal {
00017 private:
00018     std::string nome;
00019
00020 public:
00021     Canal();
00022     virtual ~Canal();
00023     Canal(std::string nome);
00024
00025     void setNome(std::string nome);
00026
00027     std::string getNome();
00028
00029     bool operator==(const Canal &other) const;
00030 };
00031
00032 #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 destrutora
00014  */
00015 class Canal_Texto : public Canal {
00016 private:
00017     std::vector<Mensagem> mensagens;
00018
00019 public:
```

```

00020     std::vector<Mensagem> texto;
00021
00022     Canal_Texto();
00023     ~Canal_Texto();
00024     Canal_Texto(std::string nome);
00025
00026     void adicionarMensagem(const Mensagem &mensagem);
00027 };
00028
00029 #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     Canal_Voz(std::string nome);
00023
00024     void setUltima(Mensagem x);
00025
00026     std::string getUltima();
00027 };
00028
00029 #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
00062     void adicionarParticipante(int id);
00063
00064     void adicionarCanal(Canal *canal);
00065
00070     int getDono();
00071
00076     std::string getNome();
00077
00082     std::string getDesc();
00083
00088     std::string getCodigo();
00089
00094     std::vector<int> getParticipantesId();
00095
00096     std::vector<Canal *> getCanais();
00097 };
00098
00099 #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 nomeServidorAtual; // Nome do servidor atual.
00021     void downloadServidores(); // Metodo para salvar servidores em um arquivo txt.
00022     void downloadUsuarios(); // Metodo para salvar usuarios em um arquivo txt.
00023     void uploadServidores(); // Metodo para carregar servidores de um arquivo txt.
00024     void uploadUsuarios(); // Metodo para carregar usuarios de um arquivo txt.
00025
00026 public:
00030     Sistema() = default;
00031
00035     ~Sistema();
00036
00040     void download();
00041
00045     void upload();
00046
00047     static std::string
00048         getNomeServidorAtual; // Getter para o nome do servidor atual.
00049     static int getIdUsuarioAtual; // Getter para o ID do usuário atual.
00050     static int getIdCanalAtual; // Getter para o ID do canal atual.
00051
00056     std::string timeMessage();
00057
00063     bool procurarEmail(const std::string &email);
00064
00070     bool procurarServidor(const std::string &nome);
00071
00078     bool procurarCanal(const std::string &nome, const std::string &tipo);
00079
00080     // funcionalidades pt 1
00081
00085     void iniciar();
00086
00091     bool quit();
00092
00100     bool create_user(const std::string email, const std::string senha,
00101                     const std::string nome);
00102
00108     void login(const std::string email, const std::string senha);
00109
00113     void disconnect();
00114
00119     void create_server(const std::string nome);
00120
00126     void set_server_desc(const std::string nome, const std::string desc);
00127
00133     void set_server_invite_code(const std::string nome,
00134                                 const std::string convite);
00135
00139     void list_servers();
00140
00145     void remove_server(const std::string nome);
00146
00152     void enter_server(const std::string nome, const std::string convite);
00153
00157     void leave_server();
00158
00162     void list_participants();
00163
00164     // funcionalidades pt 2
00165
00171     void create_channel(const std::string nome, const std::string tipo);
00172
00176     void list_channels();
00177
00182     void enter_channel(const std::string nome);
00183
00187     void leave_channel();
00188
00193     void send_message(const std::string mensagem);
00194

```

```
00198     void list_messages();
00199 };
00200
00201 #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

- ~Canal
 - Canal, [9](#)
- ~Canal_Texto
 - Canal_Texto, [11](#)
- ~Canal_Voz
 - Canal_Voz, [12](#)
- ~Mensagem
 - Mensagem, [13](#)
- ~Servidor
 - Servidor, [20](#)
- ~Sistema
 - Sistema, [24](#)
- ~Usuario
 - Usuario, [33](#)
- Canal, [9](#)
 - ~Canal, [9](#)
 - Canal, [9](#)
 - getNome, [10](#)
 - operator==, [10](#)
 - setNome, [10](#)
- Canal_Texto, [11](#)
 - ~Canal_Texto, [11](#)
 - Canal_Texto, [11](#)
- Canal_Voz, [12](#)
 - ~Canal_Voz, [12](#)
 - Canal_Voz, [12](#)
 - getUltima, [12](#)
 - setUltima, [12](#)
- create_channel
 - Sistema, [25](#)
- create_server
 - Sistema, [25](#)
- create_user
 - Sistema, [25](#)
- disconnect
 - Sistema, [26](#)
- download
 - Sistema, [26](#)
- enter_channel
 - Sistema, [26](#)
- enter_server
 - Sistema, [27](#)
- getArg
 - Parser, [17](#)
- getArgsEspace
 - Parser, [18](#)

- getCodigo
 - Servidor, [21](#)
- getComando
 - Parser, [18](#)
- getConteudo
 - Mensagem, [14](#)
- getDataHora
 - Mensagem, [14](#)
- getDesc
 - Servidor, [21](#)
- getDono
 - Servidor, [21](#)
- getEmail
 - Usuario, [34](#)
- getEnviadoPor
 - Mensagem, [14](#)
- getId
 - Usuario, [34](#)
- getNome
 - Canal, [10](#)
 - Servidor, [22](#)
 - Usuario, [34](#)
- getParticipantesId
 - Servidor, [22](#)
- getSenha
 - Usuario, [34](#)
- getUltima
 - Canal_Voz, [12](#)
- include/canal.h, [39](#)
- include/canal_texto.h, [39](#)
- include/canal_voz.h, [40](#)
- include/constantes.h, [40](#)
- include/mensagem.h, [41](#)
- include/parser.h, [42](#)
- include/servidor.h, [42](#)
- include/sistema.h, [43](#)
- include/usuario.h, [44](#)
- iniciar
 - Sistema, [27](#)
- leave_channel
 - Sistema, [27](#)
- leave_server
 - Sistema, [27](#)
- list_channels
 - Sistema, [28](#)
- list_messages
 - Sistema, [28](#)
- list_participants

- Sistema, 28
- list_servers
 - Sistema, 28
- login
 - Sistema, 28
- Mensagem, 13
 - ~Mensagem, 13
 - getConteudo, 14
 - getDataHora, 14
 - getEnviadoPor, 14
 - Mensagem, 13, 14
 - operator==, 15
 - setConteudo, 15
 - setDataHora, 15
 - setEnviadoPor, 17
- operator==
 - Canal, 10
 - Mensagem, 15
 - Usuario, 35
- parse
 - Parser, 18
- Parser, 17
 - getArg, 17
 - getArgsEspace, 18
 - getComando, 18
 - parse, 18
 - print, 19
 - qtdArgs, 19
- print
 - Parser, 19
- procurarCanal
 - Sistema, 29
- procurarEmail
 - Sistema, 29
- procurarServidor
 - Sistema, 30
- qtdArgs
 - Parser, 19
- quit
 - Sistema, 30
- README, 2
- remove_server
 - Sistema, 31
- send_message
 - Sistema, 31
- Servidor, 20
 - ~Servidor, 20
 - getCodigo, 21
 - getDesc, 21
 - getDono, 21
 - getNome, 22
 - getParticipantesId, 22
 - Servidor, 20, 21
 - setCodigo, 22
 - setDescricao, 23
 - setNome, 23
- set_server_desc
 - Sistema, 31
- set_server_invite_code
 - Sistema, 32
- setCodigo
 - Servidor, 22
- setConteudo
 - Mensagem, 15
- setDataHora
 - Mensagem, 15
- setDescricao
 - Servidor, 23
- setEmail
 - Usuario, 35
- setEnviadoPor
 - Mensagem, 17
- setId
 - Usuario, 36
- setNome
 - Canal, 10
 - Servidor, 23
 - Usuario, 36
- setSenha
 - Usuario, 36
- setUltima
 - Canal_Voz, 12
- Sistema, 24
 - ~Sistema, 24
 - create_channel, 25
 - create_server, 25
 - create_user, 25
 - disconnect, 26
 - download, 26
 - enter_channel, 26
 - enter_server, 27
 - iniciar, 27
 - leave_channel, 27
 - leave_server, 27
 - list_channels, 28
 - list_messages, 28
 - list_participants, 28
 - list_servers, 28
 - login, 28
 - procurarCanal, 29
 - procurarEmail, 29
 - procurarServidor, 30
 - quit, 30
 - remove_server, 31
 - send_message, 31
 - set_server_desc, 31
 - set_server_invite_code, 32
 - Sistema, 24
 - timeMessage, 32
 - upload, 32
- timeMessage
 - Sistema, 32

upload

 Sistema, [32](#)

Usuario, [33](#)

 ~Usuario, [33](#)

 getEmail, [34](#)

 getId, [34](#)

 getNome, [34](#)

 getSenha, [34](#)

 operator==, [35](#)

 setEmail, [35](#)

 setId, [36](#)

 setNome, [36](#)

 setSenha, [36](#)

 Usuario, [33](#)