Concord

v0.3

Gerado por Doxygen 1.9.7

Chapter 1

Concord

Projeto de ITP

Aluno: Erick Marques Oliveira Azevedo

Matrícula: 20210047901

1.1 Compilando a aplicação

1.1.1 Usando o Bash/Wsl:

```
# Na pasta do programa compile o programa usando o CMake:
$ cmake .
$ make
# Em seguida execute o arquivo gerado com o seguinte comando:
$ ./concord
```

1.2 Lista de comandos disponíveis

quit - Encerra o programa.

help - Exibe a tela de ajuda que contém todos os comandos do programa.

create-user <E-mail> <Senha_sem_espaços> <Nome_do_usuário> - Registra um novo usuário.

login <E-mail> <senha> - Loga o usuário no sistema.

disconnect - Desconecta o usuário atual.

create-server <id> <nome-do-servidor> - Cria um novo servidor.

set-server-desc <id> <nome-do-servidor> <descrição> - Altera a descrição de um servidor que você é o dono.

set-server-invite-code <id> <nome-do-servidor> <código-desejado> - Adiciona um código de convite a um servidor, tornando-o privado.

list-servers <id> - Lista todos os servidores criados.

remove-server <id> <nome-do-servidor> - Elimina um servidor.

enter-server <nome-do-servidor> <id> <nome-do-servidor> <código-de-convite> - Entra em um servidor, se ele for público, o código de convite não é necessário.

leave-server <id> <nome-do-servidor> - Quando estiver dentro do servidor, desconecte-se dele.

list-participants <id> - Lista todos os participantes de um servidor.

list-participants <id> - Lista todos os participantes de um servidor.

list-users <id> - Lista os usuários do sistema e exibe seus status.

create-channel <id> <nome> - Cria um novo canal dentro de um servidor.

list-channels <id> - Lista os canais do servidor.

enter-channel <id> - Entra em um canal existente.

leave-channel <id> - Sai do canal.

send-message <id> <mensagem> - Envia uma mensagem no canal atual.

list-messages <id> - Lista todas as mensagens do canal.

salvar - Salve os usuários e servidores registrados em arquivos.

Chapter 2

Índice das Estruturas de Dados

2.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

Chapter 3

Índice dos Arquivos

3.1 Lista de Arquivos

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

Chapter 4

Estruturas

4.1 Referência da Classe CanalTexto

`#include <canalt.h>`

Membros Públicos

- CanalTexto (string nome)
  Construtor.
- string get_nome ()
  Get do atributo nome.
- void add_mensagem (Mensagem mensagem)
  Adiciona uma mensagem a lista de mensagens.
- string listar_mensagens (vector< Usuario > &usuarios)
  Gera uma string que lista todas as mensagens enviadas.

Atributos Privados

- string nome
- vector< Mensagem > mensagens

4.1.1 Descrição detalhada

Definição na linha 10 do arquivo canalt.h.

4.1.2 Construtores e Destrutores

4.1.2.1 CanalTexto()

```
CanalTexto::CanalTexto (
            string nome )
```

Construtor.

Parâmetros

nome	O nome do canal de texto.

Definição na linha 11 do arquivo canalt.cpp.

```
00012 {
00013     this->nome = nome;
00014 }
```

4.1.3 Documentação das funções

4.1.3.1 add_mensagem()

```
void CanalTexto::add_mensagem (
            Mensagem mensagem )
```

Adiciona uma mensagem a lista de mensagens.

Parâmetros

mensagem	Mensagem que será adicionada.

Definição na linha 21 do arquivo canalt.cpp.

```
00022 {
00023     this->mensagens.push_back(mensagem);
00024 }
```

4.1.3.2 get_nome()

```
string CanalTexto::get_nome ( )
```

Get do atributo nome.

Retorna

Valor do atributo nome.

Definição na linha 16 do arquivo canalt.cpp.

```
00017 {
00018     return this->nome;
00019 }
```

4.1.3.3 listar_mensagens()

```
string CanalTexto::listar_mensagens (
            vector< Usuario > & usuarios )
```

Gera uma string que lista todas as mensagens enviadas.

Parâmetros

usuarios	Lista dos usuários disponíveis.

Retorna

Lista de mensagens formatada.

Definição na linha 26 do arquivo canalt.cpp.

```
00027 {
00028     stringstream ss;
00029     for (auto& m : this->mensagens) {
00030         for (auto& u : usuarios) {
00031             if (m.get_enviada_por() == u.get_id()) {
00032                 ss « u.get_nome() « " " « m.get_data_hora();
00033                 ss  « ": " « m.get_conteudo() « endl;
00034                 break;
00035             }
00036         }
00037     }
00038
00039     if (ss.str().empty())
00040         return "Nenhuma mensagem foi enviada nesse canal ainda, inicie uma conversa! :D";
00041     return ss.str();
00042 }
```

4.1.4 Campos

4.1.4.1 mensagens

```
vector<Mensagem> CanalTexto::mensagens  [private]
```

Definição na linha 13 do arquivo canalt.h.

4.1.4.2 nome

```
string CanalTexto::nome  [private]
```

Definição na linha 12 do arquivo canalt.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/canalt.h
- src/canalt.cpp

4.2 Referência da Classe Mensagem

```
#include <mensagem.h>
```

Membros Públicos

- Mensagem (int enviada_por, string conteudo)

  Construtor, data_hora é gerado automaticamente.
- int get_enviada_por ()

  Gets de data_hora, enviada_por e conteudo.
- string get_data_hora ()
- string get_conteudo ()

Atributos Privados

- int enviada_por
- string data_hora
- string conteudo

4.2.1 Descrição detalhada

Definição na linha 7 do arquivo mensagem.h.

4.2.2 Construtores e Destrutores

4.2.2.1 Mensagem()

```
Mensagem::Mensagem (
            int enviada_por,
            string conteudo )
```

Construtor, data_hora é gerado automaticamente.

Parâmetros

enviada_por	ID do usuário que enviou a mensagem.
conteudo	O conteúdo da mensagem.

Definição na linha 25 do arquivo mensagem.cpp.

```
00026 {
00027     this->enviada_por = enviada_por;
00028     this->conteudo = conteudo;
00029
00030     time_t t = time(0);
00031     tm* now = localtime(&t);
00032     stringstream ss;
00033     ss.fill('0');
00034     ss « "[" « setw(2) « now->tm_mday « "/" « setw(2) « now->tm_mon+1;
00035     ss « "/" « setw(2) « now->tm_year-100;
00036     ss « " às " « setw(2) « now->tm_hour « ":" « setw(2) « now->tm_min « "]";
00037     this->data_hora = ss.str();
00038 }
```

4.2.3 Documentação das funções

4.2.3.1 get_conteudo()

```
string Mensagem::get_conteudo ( )
```

Definição na linha 20 do arquivo mensagem.cpp.

```
00021 {
00022     return this->conteudo;
00023 }
```

4.2.3.2 get_data_hora()

```
string Mensagem::get_data_hora ( )
```

Definição na linha 15 do arquivo mensagem.cpp.

```
00016 {
00017     return this->data_hora;
00018 }
```

4.2.3.3 get_enviada_por()

```
int Mensagem::get_enviada_por ( )
```

Gets de data_hora, enviada_por e conteudo.

Retorna

O valor do atributo correspondente.

Definição na linha 10 do arquivo mensagem.cpp.

```
00011 {
00012     return this->enviada_por;
00013 }
```

4.2.4 Campos

4.2.4.1 conteudo

```
string Mensagem::conteudo  [private]
```

Definição na linha 11 do arquivo mensagem.h.

4.2.4.2 data_hora

```
string Mensagem::data_hora  [private]
```

Definição na linha 10 do arquivo mensagem.h.

4.2.4.3 enviada_por

```
int Mensagem::enviada_por  [private]
```

Definição na linha 9 do arquivo mensagem.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

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

4.3 Referência da Classe principal

`#include <principal.h>`

Diagrama de colaboração para principal:

Membros Públicos

- principal (Sistema &sistema)
- string processar_linha (string linha)
- void iniciar (istream &in, ostream &out)

  Método que recebe por o ¨cin¨ e o ¨cout¨ no main ele faz o lê 1 comando por linha e o processa, devolvendo ao output o resultado de cada um.

Atributos Privados

- Sistema * sistema
- stringstream ss
- bool sair = false

4.3.1 Descrição detalhada

Definição na linha 10 do arquivo principal.h.

4.3.2 Construtores e Destrutores

4.3.2.1 principal()

```
principal::principal (
              Sistema & sistema )
```

Definição na linha 22 do arquivo principal.cpp.

```
00023 {
00024     this->sair = false;
00025     this->sistema = &sistema;
00026 }
```

4.3.3 Documentação das funções

4.3.3.1 iniciar()

```
void principal::iniciar (
              istream & in,
              ostream & out )
```

Método que recebe por o ¨cin¨ e o ¨cout¨ no main ele faz o lê 1 comando por linha e o processa, devolvendo ao output o resultado de cada um.

Definição na linha 168 do arquivo principal.cpp.

```
00169 {
00170     string saida, linha;
00171     this->sair = false;
00172     while (!this->sair) {
00173         if (getline(inputStream, linha)) {
00174             saida = processar_linha(linha);
00175             outputStream « saida « endl;
00176         }
00177     }
00178 }
```

4.3.3.2 processar_linha()

```
string principal::processar_linha (
            string linha )
```

Definição na linha 29 do arquivo principal.cpp.

```
00030 {
00031     istringstream buf(linha);
00032     string Cchooser;
00033     buf » Cchooser;
00034
00035     if (Cchooser.empty()) {
00036         return "Digite um comando!";
00037     }
00038     if (Cchooser == "quit" ) {
00039         this->sair = true;
00040         return sistema->quit();
00041     }
00042     else if(Cchooser == "salvar"){
00043          sistema->salvar();
00044          return "Arquivos de servidores e usuários salvos!";
00045     }
00046     else if (Cchooser == "help") {
00047         return sistema->help();
00048     }
00049     else if (Cchooser == "create-user") {
00050         string email, senha, nome;
00051         buf » email;
00052         buf » senha;
00053         nome = resto_de(buf);
00054         if (email.empty() || senha.empty() || nome.empty())
00055             return "»>Uso: create-user EMAIL SENHA NOME";
00056         return sistema->create_user(email, senha, nome);
00057     }
00058     else if (Cchooser == "login") {
00059         string email, senha;
00060         buf » email;
00061         buf » senha;
00062         if (email.empty() || senha.empty())
00063             return "»>Uso: login EMAIL SENHA";
00064         return sistema->login(email, senha);
00065     }
00066     else if (Cchooser == "list-users") {
00067         return sistema->list_users();
00068     }
00069
00070     int id;
00071     if (!(buf » id)) {
00072         return "Este comando precisa ser precedido de um ID [" + Cchooser + "]...";
00073     }
00074
00075     if (Cchooser == "disconnect") {
00076         return sistema->disconnect(id);
00077     }
00078     else if (Cchooser == "create-server") {
00079         string nome;
00080         buf » nome;
00081         if (nome.empty())
00082             return "»>Uso: create-server ID NOME";
00083         return sistema->create_server(id, nome);
00084     }
00085     else if (Cchooser == "set-server-desc") {
00086         string nome, descricao;
00087         buf » nome;
00088         descricao = resto_de(buf);
00089         if (nome.empty())
00090             return "»>Uso: set-server-desc ID NOME [DESCRICAO...]";
00091         return sistema->set_server_desc(id, nome, descricao);
00092     }
00093     else if (Cchooser == "set-server-invite-code") {
00094         string nome, codigo;
00095         buf » nome;
00096         buf » codigo;
00097         if (nome.empty())
00098             return "»>Uso: set-server-invite-code ID NOME [CODIGO]";
00099         return sistema->set_server_invite_code(id, nome, codigo);
00100     }
00101     else if (Cchooser == "list-servers") {
00102         return sistema->list_servers(id);
00103     }
00104     else if (Cchooser == "remove-server") {
00105         string nome;
00106         buf » nome;
00107         if (nome.empty())
```

```
00108             return "»>Uso: remove-server ID NOME";
00109         return sistema->remove_server(id, nome);
00110     }
00111     else if (Cchooser == "enter-server") {
00112         string nome, codigo;
00113         buf » nome;
00114         buf » codigo;
00115         if (nome.empty())
00116             return "»>Uso: enter-server ID NOME [CODIGO]";
00117         return sistema->enter_server(id, nome, codigo);
00118     }
00119     else if (Cchooser == "leave-server") {
00120         string nome;
00121         buf » nome;
00122         if (nome.empty())
00123             return "»>Uso: leave-server ID NOME";
00124         return sistema->leave_server(id, nome);
00125     }
00126     else if (Cchooser == "list-participants") {
00127         return sistema->list_participants(id);
00128     }
00129     else if (Cchooser == "list-channels") {
00130         return sistema->list_channels(id);
00131     }
00132     else if (Cchooser == "create-channel") {
00133         string nome;
00134         buf » nome;
00135         if (nome.empty())
00136             return "»>Uso: create-channel ID NOME";
00137         return sistema->create_channel(id, nome);
00138     }
00139     else if (Cchooser == "enter-channel") {
00140         string nome;
00141         buf » nome;
00142         if (nome.empty())
00143             return "»>Uso: enter-channel ID NOME";
00144         return sistema->enter_channel(id, nome);
00145     }
00146     else if (Cchooser == "leave-channel") {
00147         return sistema->leave_channel(id);
00148     }
00149     else if (Cchooser == "send-message") {
00150         string mensagem;
00151         mensagem = resto_de(buf);
00152         if (mensagem.empty())
00153             return "»>Uso: send-message ID MESSAGE...";
00154         return sistema->send_message(id, mensagem);
00155     }
00156     else if (Cchooser == "list-messages") {
00157         return sistema->list_messages(id);
00158     }
00159     else {
00160         return "»>Comando não reconhecido [" + Cchooser + "]...";
00161     }
00162 }
```

4.3.4 Campos

4.3.4.1 sair

```
bool principal::sair = false  [private]
```

Definição na linha 14 do arquivo principal.h.

4.3.4.2 sistema

```
Sistema* principal::sistema  [private]
```

Definição na linha 12 do arquivo principal.h.

4.3.4.3 ss

```
stringstream principal::ss  [private]
```

Definição na linha 13 do arquivo principal.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/principal.h
- src/principal.cpp

4.4 Referência da Classe Servidor

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

Membros Públicos

- Servidor (int usuario_dono_id, string nome)

  canais_texto e participantes_id são iniciados vazios.
- int get_usuario_dono_id ()

  Gets dos atributos.
- string get_nome ()
- string get_descricao ()
- string get_convite ()
- void set_descricao (string descricao)

  Sets dos atributos.
- void set_convite (string codigo)
- void add_participante (int id)

  Adiciona um ID para participantes_id.
- void remove_participante (int id)

  Remove um ID de participantes_id.
- string imprimir_participantes (vector< Usuario > &usuarios)

  Cria uma string que contém todos os usuários em participantes_id.
- bool eh_participante (int id)

  Verifica se um ID existe em participantes_id.
- void add_canal_texto (CanalTexto &canal)

  Adiciona um canal de texto a lista de canais de texto.
- void remove_canal_texto (string canal)

  Remove um canal de texto da lista de canais de texto.
- bool eh_canal_texto (string canal)

  Verifica se um canal de texto existe na lista de canais.
- string imprimir_canais_texto ()

  Cria uma string com a lista de todos os canais registrados.
- void enviar_mensagem (Mensagem &mensagem, string nome_canal)

  Adiciona uma mensagem para a lista de mensagens.
- string imprimir_mensagens (string nome_canal, vector< Usuario > &usuarios)

  Imprime uma lista formatada de mensagens.

Atributos Privados

- int usuario_dono_id
- string nome
- string descricao
- string convite
- vector< int > participantes_id
- vector< CanalTexto > canais_texto

4.4.1 Descrição detalhada

Definição na linha 11 do arquivo servidor.h.

4.4.2 Construtores e Destrutores

4.4.2.1 Servidor()

```
Servidor::Servidor (
            int usuario_dono_id,
            string nome )
```

canais_texto e participantes_id são iniciados vazios.

Parâmetros

usuario_dono_id	é o ID do dono do servidor.
nome	é o nome do servidor.

Definição na linha 11 do arquivo servidor.cpp.

```
00012 {
00013     this->nome = nome;
00014     this->usuario_dono_id = dono_id;
00015     this->descricao = "";
00016     this->convite = "";
00017     this->participantes_id.push_back(dono_id);
00018 }
```

4.4.3 Documentação das funções

4.4.3.1 add_canal_texto()

```
void Servidor::add_canal_texto (
            CanalTexto & canal )
```

Adiciona um canal de texto a lista de canais de texto.

Parâmetros

canal	O novo canal de texto a ser adicionado.

Definição na linha 89 do arquivo servidor.cpp.

```
00090 {
00091     this->canais_texto.push_back(canal);
00092 }
```

4.4.3.2 add_participante()

```
void Servidor::add_participante (
            int id )
```

Adiciona um ID para participantes_id.

Parâmetros

id	o ID a ser adicionado.

Definição na linha 50 do arquivo servidor.cpp.

```
00051 {
00052     this->participantes_id.push_back(id);
00053 }
```

4.4.3.3 eh_canal_texto()

```
bool Servidor::eh_canal_texto (
            string canal )
```

Verifica se um canal de texto existe na lista de canais.

Parâmetros

nome	Nome do canal a ser verificado.

Retorna

Verdadeiro se o canal existir ou falso se o canal não existir.

Definição na linha 107 do arquivo servidor.cpp.

```
00108 {
00109     for (auto& textchannel : this->canais_texto) {
00110         if (textchannel.get_nome() == canal)
00111             return true;
00112     }
00113     return false;
00114 }
```

4.4.3.4 eh_participante()

```
bool Servidor::eh_participante (
            int id )
```

Verifica se um ID existe em participantes_id.

Parâmetros

id	ID que será verificado.

Retorna

Verdadeiro se o ID estiver em participantes_id; Falso caso contrário.

Definição na linha 80 do arquivo servidor.cpp.

```
00081 {
00082     for (auto& p_id : this->participantes_id) {
00083         if (p_id == id)
00084             return true;
00085     }
00086     return false;
00087 }
```

4.4.3.5 enviar_mensagem()

```
void Servidor::enviar_mensagem (
            Mensagem & mensagem,
            string nome_canal )
```

Adiciona uma mensagem para a lista de mensagens.

Parâmetros

mensagem	A mensagem que será adicionada.
nome_canal	O nome do canal no qual receberá a mensagem.

Definição na linha 128 do arquivo servidor.cpp.

```
00129 {
00130     for (auto& textchannel : this->canais_texto) {
00131         if (textchannel.get_nome() == nome_canal)
00132             textchannel.add_mensagem(mensagem);
00133     }
00134 }
```

4.4.3.6 get_convite()

```
string Servidor::get_convite ( )
```

Definição na linha 35 do arquivo servidor.cpp.

```
00036 {
00037     return this->convite;
00038 }
```

4.4.3.7 get_descricao()

```
string Servidor::get_descricao ( )
```

Definição na linha 30 do arquivo servidor.cpp.

```
00031 {
00032     return this->descricao;
00033 }
```

4.4.3.8 get_nome()

```
string Servidor::get_nome ( )
```

Definição na linha 20 do arquivo servidor.cpp.

```
00021 {
00022     return this->nome;
00023 }
```

4.4.3.9 get_usuario_dono_id()

```
int Servidor::get_usuario_dono_id ( )
```

Gets dos atributos.

Retorna

Valor do atributo correspondente.

Definição na linha 25 do arquivo servidor.cpp.

```
00026 {
00027     return this->usuario_dono_id;
00028 }
```

4.4.3.10 imprimir_canais_texto()

```
string Servidor::imprimir_canais_texto ( )
```

Cria uma string com a lista de todos os canais registrados.

Retorna

A lista de canais formatada.

Definição na linha 116 do arquivo servidor.cpp.

```
00117 {
00118     stringstream ss;
00119     for (auto& textchannel : this->canais_texto) {
00120         ss « "-> " « textchannel.get_nome() « endl;
00121     }
00122
00123     if (ss.str().empty())
00124         return "Não existem canais de texto neste servidor, caso seja o dono, experimente adicionar
      um!";
00125     return ss.str();
00126 }
```

4.4.3.11 imprimir_mensagens()

```
string Servidor::imprimir_mensagens (
            string nome_canal,
            vector< Usuario > & usuarios )
```

Imprime uma lista formatada de mensagens.

Parâmetros

nome_canal	O nome do canal no qual a lista de mensagens será requisitada.
usuarios	Lista de usuários disponíveis.

Retorna

A lista de mensagens formatada.

Definição na linha 136 do arquivo servidor.cpp.

```
00137 {
00138     for (auto& textchannel : this->canais_texto) {
00139         if (canal_texto == textchannel.get_nome())
00140             return textchannel.listar_mensagens(usuarios);
00141     }
00142     return "Este canal de texto não existe!";
00143 }
```

4.4.3.12 imprimir_participantes()

```
string Servidor::imprimir_participantes (
            vector< Usuario > & usuarios )
```

Cria uma string que contém todos os usuários em participantes_id.

Parâmetros

usuarios	Lista de todos os usuários participantes.

Retorna

Lista formatada de todos os usuários participantes.

Definição na linha 67 do arquivo servidor.cpp.

```
00068 {
00069     stringstream ss;
00070     for (auto& user : usuarios) {
00071         if (this->eh_participante(user.get_id()))
00072             ss « "-" « user.get_id() « " " « user.get_nome() « endl;
00073     }
00074
00075     if (ss.str().empty())
00076         return "Ninguém está nesse servidor ainda, entre e convide seus amigos!";
00077     return ss.str();
00078 }
```

4.4.3.13 remove_canal_texto()

```
void Servidor::remove_canal_texto (
            string canal )
```

Remove um canal de texto da lista de canais de texto.

Parâmetros

canal	O nome do canal a ser removido.

Definição na linha 94 do arquivo servidor.cpp.

```
00095 {
00096     auto contador = this->canais_texto.begin();
00097     while (contador != this->canais_texto.end()) {
00098         //Caso o nome seja encontrado na lista de canais, ele é deletado.
00099         if (contador->get_nome() == canal) {
00100             this->canais_texto.erase(contador);
00101             return;
00102         }
00103         ++contador;
00104     }
00105 }
```

4.4.3.14 remove_participante()

```
void Servidor::remove_participante (
            int id )
```

Remove um ID de participantes_id.

Parâmetros

id	ID a ser removido.

Definição na linha 55 do arquivo servidor.cpp.

```
00056 {
00057     auto i = this->participantes_id.begin();
00058     while (i != this->participantes_id.end()) {
00059         if (*i == id) {
00060             this->participantes_id.erase(i);
00061             return;
00062         }
00063         ++i;
00064     }
00065 }
```

4.4.3.15 set_convite()

```
void Servidor::set_convite (
            string codigo )
```

Definição na linha 45 do arquivo servidor.cpp.

```
00046 {
00047     this->convite = convite;
00048 }
```

4.4.3.16 set_descricao()

```
void Servidor::set_descricao (
            string descricao )
```

Sets dos atributos.

Parâmetros

Novo	valor a ser setado.

Definição na linha 40 do arquivo servidor.cpp.

```
00041 {
00042     this->descricao = descricao;
00043 }
```

4.4.4 Campos

4.4.4.1 canais_texto

```
vector<CanalTexto> Servidor::canais_texto  [private]
```

Definição na linha 18 do arquivo servidor.h.

4.4.4.2 convite

```
string Servidor::convite  [private]
```

Definição na linha 16 do arquivo servidor.h.

4.4.4.3 descricao

```
string Servidor::descricao  [private]
```

Definição na linha 15 do arquivo servidor.h.

4.4.4.4 nome

```
string Servidor::nome  [private]
```

Definição na linha 14 do arquivo servidor.h.

4.4.4.5 participantes_id

```
vector<int> Servidor::participantes_id  [private]
```

Definição na linha 17 do arquivo servidor.h.

4.4.4.6 usuario_dono_id

```
int Servidor::usuario_dono_id  [private]
```

Definição na linha 13 do arquivo servidor.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

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

4.5 Referência da Classe Sistema

`#include <sistema.h>`

Membros Públicos

- void salvar ()
- string quit ()

  Função que termina o programa.
- string help ()

  Função que exibe a tela de ajuda.
- string create_user (const string email, const string senha, const string nome)

  Função que cria um usuario e retorna uma string com uma mensagem de sucesso ao criar ou um erro.
- string login (const string email, const string senha)

  Função que realiza o login do usuário com email e senha, retorna uma string de erro ou uma mensagem login bem sucedido. Quando um usuário loga o sistema deve adicionar o usuário na tabela de usuários logados.
- string disconnect (int id)

  Função que desconecta um usuário específico do sistema, removendo a informação daquele usuário da tabela de usuários logados.
- string create_server (int id, const string nome)

  Função que gera um novo servidor e o adiciona a lista de servidores.
- string set_server_invite_code (int id, const string nome, const string codigo)

  Função que altera o código de convite de um servidor.
- string set_server_desc (int id, const string nome, const string descricao)

  Função que altera a descrição do servidor para a descrição que o usuário escolher.
- string list_servers (int id)

  Função que lista os servidores cadastrados no sistema.
- string remove_server (int id, const string nome)

  Função que remove um servidor da lista de servidores.
- string enter_server (int id, const string nome, const string codigo)

  Função que adiciona o usuário logado ao servidor usando o id e o código (caso necessário).
- string leave_server (int id, const string nome)

  Remove o usuário conectado do servidor que ele está logado.
- string list_participants (int id)

  Função que lista os participantes de um servidor que esteja sendo visualizado.
- string list_users ()

  Função que lista os usuários do sistema e se estão online ou offline.
- string list_channels (int id)

  Lista os canais do servidor que o usuário está visualizando.
- string create_channel (int id, const string nome)

  Cria um canal no servidor.
- string enter_channel (int id, const string nome)
- string leave_channel (int id)

  Remove o usuário do canal que ele está visualizando atualmente.
- string send_message (int id, const string mensagem)

  Envia uma mensagem para o canal que o usuário está visualizando.
- string list_messages (int id)

  Lista as mensagens de um canal.

Membros privados

- void salvarUsuarios ()
- void salvarServidores ()

Atributos Privados

- vector< Servidor > servers
- vector< Usuario > usuarios
- map< int, pair< string, string > > usuarios_logados

4.5.1 Descrição detalhada

Definição na linha 11 do arquivo sistema.h.

4.5.2 Documentação das funções

4.5.2.1 create_channel()

```
string Sistema::create_channel (
            int id,
            const string nome )
```

Cria um canal no servidor.

Parâmetros

id	um ID válido do usuário logado.
nome	O nome do novo canal.

Retorna

string Uma string que confirma a criação do canal, ou um erro caso não seja possível criar o canal.

Definição na linha 344 do arquivo sistema.cpp.

```
00345 {
00346     if (!this->usuarios_logados.count(id))
00347         return "Você não está logado!";
00348
00349     if (this->usuarios_logados[id].first.empty())
00350         return "Você não está visualizando nenhum servidor!";
00351
00352     for (auto& server : this->servers) {
00353         if (server.get_nome() == this->usuarios_logados[id].first) {
00354             if (server.eh_canal_texto(nome))
00355                 return "Já existe um canal com este nome!";
00356
00357             CanalTexto canal(nome);
00358             server.add_canal_texto(canal);
00359             return "Canal criado com sucesso!";
00360         }
00361     }
00362     return "Você não é o dono do servidor...";
00363 }
```

4.5.2.2 create_server()

```
string Sistema::create_server (
            int id,
            const string nome )
```

Função que gera um novo servidor e o adiciona a lista de servidores.

Parâmetros

id	O ID de um usuário logado no sistema.
nome	O nome do servidor recebido.

Retorna

string confirmação da criação do servidor, ou um erro caso não seja possível criá-lo.

Definição na linha 126 do arquivo sistema.cpp.

```
00127 {
00128     if (!this->usuarios_logados.count(id))
00129         return "Você não está logado!";
00130     for (auto& server : this->servers) {
00131         if (nome == server.get_nome())
00132             return "Já existe um servidor com este nome!";
00133     }
00134     Servidor servidor(id, nome);
00135     this->servers.push_back(servidor);
00136     return "Servidor criado com sucesso!";
00137 }
```

4.5.2.3 create_user()

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

Função que cria um usuario e retorna uma string com uma mensagem de sucesso ao criar ou um erro.

Parâmetros

email	o email que o usuário inseriu
senha	a senha que o usuário inseriu
nome	o nome que o usuário inseriu

Retorna

uma string com uma mensagem que diz se o usuario foi criado com sucesso ou nao e, caso tenha tido sucesso, o ID.

Definição na linha 83 do arquivo sistema.cpp.

```
00084 {
00085     if(!email_validate(email))
00086         return "Email inválido!";
00087     for (auto& user : this->usuarios) {
00088         if (email == user.get_email())
```

```
00089             return "Já existe uma conta registrada com esse e-mail! Tente logar!";
00090         if (nome == user.get_nome())
00091             return "Este nome de usuário já foi utilizado, tente outro!";
00092     }
00093
00094     int id = this->usuarios.size() + 1;
00095     Usuario usuario(id, nome, email, senha);
00096     this->usuarios.push_back(usuario);
00097     return cat("Novo usuário registrado! Seu ID: #", id);
00098 }
```

4.5.2.4 disconnect()

```
string Sistema::disconnect (
            int id )
```

Função que desconecta um usuário específico do sistema, removendo a informação daquele usuário da tabela de usuários logados.

Parâmetros

id	o ID válido de um usuário logado.

Retorna

Mensagem confirmando o logoff ou uma mensagem de erro caso falhe.

Definição na linha 117 do arquivo sistema.cpp.

```
00118 {
00119     if (!this->usuarios_logados.count(id))
00120         return "Este usuário já está desconectado...";
00121
00122     this->usuarios_logados.erase(id);
00123     return "Usuário offline!";
00124 }
```

4.5.2.5 enter_channel()

```
string Sistema::enter_channel (
            int id,
            const string nome )
```

Faz com que o usuário com id dado entre em um canal específico(com seu nome e tipo) ao entrar em um canal o sistema deve atualizar a tabela Sistema::usuariosLogados com a informação de que o usuário está vizualizando o canal em que entrou. Retorna uma mensagem de sucesso ou de erro em caso de falha.

Parâmetros

id	um id válido de algum usuário cadastrado e logado no sistema.
o	nome do canal que deseja entrar,

Retorna

¨Usuário entrou no canal <nome>!¨ ou uma mensagem de erro em caso de falha.

Definição na linha 365 do arquivo sistema.cpp.

```
00366 {
00367     if (!this->usuarios_logados.count(id))
00368         return "Você não está logado!";
00369 
00370     if (this->usuarios_logados[id].first.empty())
00371         return "Você não está visualizando nenhum servidor!";
00372 
00373     if (!this->usuarios_logados[id].second.empty())
00374         return "Você já está visualizando um canal de texto!";
00375 
00376     for (auto& server : this->servers) {
00377         if (server.get_nome() == this->usuarios_logados[id].first) {
00378             if (server.eh_canal_texto(nome)){
00379                 this->usuarios_logados[id].second = nome;
00380                 return cat("O usuário ", nome, " entrou no canal!");
00381             }
00382         }
00383     }
00384     return "Canal não encontrado!";
00385 }
```

4.5.2.6 enter_server()

```
string Sistema::enter_server (
              int id,
              const string nome,
              const string codigo )
```

Função que adiciona o usuário logado ao servidor usando o id e o código (caso necessário).

Parâmetros

id	o ID do usuário registrado e logado no sistema.
nome	o nome de um servidor registrado no sistema.
codigo	o código de convite para o servidor (caso necessário) ou uma string vazia.

Retorna

string com a confirmação de entrada no servidor ou erro caso não seja possível adicionar o usuário no servidor.

Definição na linha 212 do arquivo sistema.cpp.

```
00213 {
00214     if (!this->usuarios_logados.count(id))
00215         return "Você não está logado!";
00216 
00217     for (auto& server : this->servers) {
00218         if (nome == server.get_nome()) {
00219             if (this->usuarios_logados[id].first == nome)
00220                 return "Este usuário já está neste servidor!";
00221             if (id != server.get_usuario_dono_id() && codigo != server.get_convite())
00222                 return "Código de convite inválido, você o digitou corretamente?";
00223             this->usuarios_logados[id].first = nome;
00224             this->usuarios_logados[id].second = "";
00225             if (server.eh_participante(id))
00226                 return "Você entrou no servidor com sucesso!";
00227             server.add_participante(id);
00228             return "Você entrou no servidor com sucesso! Bem-vindo!";
00229         }
00230     }
00231     return "Este servidor não existe!";
00232 }
```

4.5.2.7 help()

```
string Sistema::help ( )
```

Função que exibe a tela de ajuda.

Retorna

uma string contendo todos os comandos do programa.

Definição na linha 56 do arquivo sistema.cpp.

```
00057 {
00058     string helpmi = "\n_____________________________Lista de comandos_____________________________\n\n --- "
00059                     "quit - Encerra o programa.\n --- "
00060                     "create-user <E-mail> <Senha_sem_espaços> <Nome do usuário> - Registra um novo usuário. \n --- "
00061                     "login <E-mail> <senha> - Loga o usuário no sistema.\n --- "
00062                     "disconnect - Desconecta o usuário atual.\n --- "
00063                     "create-server <id> <nome-do-servidor> - Cria um novo servidor.\n --- "
00064                     "set-server-desc <id> <nome-do-servidor> <descrição> - Altera a descrição de um servidor que você é o dono.\n --- "
00065                     "set-server-invite-code <id> <nome-do-servidor> <código-desejado> - Adiciona um código de convite a um servidor, tornando-o privado.\n --- "
00066                     "list-servers <id> - Lista todos os servidores criados.\n --- "
00067                     "remove-server <id> <nome-do-servidor> - Elimina um servidor.\n --- "
00068                     "enter-server <nome-do-servidor> <id> <nome-do-servidor> <código-de-convite> - Entra em um servidor, se ele for público, o código de convite não é necessário.\n --- "
00069                     "leave-server <id> <nome-do-servidor> - Quando estiver dentro do servidor, desconecte-se dele.\n --- "
00070                     "list-participants <id> - Lista todos os participantes de um servidor.\n --- "
00071                     "list-users <id> - Lista os usuários do sistema e exibe seus status.\n --- "
00072                     "create-channel <id> <nome> - Cria um novo canal dentro de um servidor.\n --- "
00073                     "list-channels <id> - Lista os canais do servidor.\n --- "
00074                     "enter-channel <id> - Entra em um canal existente.\n --- "
00075                     "leave-channel <id> - Sai do canal.\n --- "
00076                     "send-message <id> <mensagem> - Envia uma mensagem no canal atual.\n --- "
00077                     "list-messages <id> - Lista todas as mensagens do canal.\n"
00078                     "salvar - Salve os usuários e servidores registrados em arquivos.\n\n"
00079                     "_____________________________________________________\n";
00080     return helpmi;
00081 }
```

4.5.2.8 leave_channel()

```
string Sistema::leave_channel (
            int id )
```

Remove o usuário do canal que ele está visualizando atualmente.

Parâmetros

id	O ID válido do usuário logado.

Retorna

string com a confirmação de que o usuário saiu do canal, ou uma mensagem de erro não seja possível removê-lo do canal.

Definição na linha 387 do arquivo sistema.cpp.

```
00388 {
00389     if (!this->usuarios_logados.count(id))
00390         return "Você não está logado!";
00391
00392     if (this->usuarios_logados[id].first.empty())
00393         return "Você não está visualizando nenhum servidor!";
00394
00395     if (this->usuarios_logados[id].second.empty())
00396         return "Você não está visualizando nenhum canal de texto!";
00397
00398     string c;
00399     c = this->usuarios_logados[id].second;
00400     this->usuarios_logados[id].second = "";
00401     return cat("Você saiu do canal ", c, "!");
00402
00403 }
```

4.5.2.9 leave_server()

```
string Sistema::leave_server (
            int id,
            const string nome )
```

Remove o usuário conectado do servidor que ele está logado.

Parâmetros

id	um ID válido de um usuário cadastrado e logado no sistema.
nome	um nome válido de um servidor cadastrado no sistema.

Retorna

string Confirmação que o usuário saiu do servidor, ou um erro, caso não seja possível remover o usuário.

Definição na linha 234 do arquivo sistema.cpp.

```
00235 {
00236     if (!this->usuarios_logados.count(id))
00237         return "Você não está logado!";
00238
00239     for (auto& server : this->servers) {
00240         if (nome == server.get_nome()) {
00241             if (id == server.get_usuario_dono_id())
00242                 return "»Você não pode remover o dono do servidor!";
00243
00244             if (!server.eh_participante(id))
00245                 return "Este usuário não participa deste servidor...";
00246
00247             if (this->usuarios_logados[id].first == nome) {
00248                 this->usuarios_logados[id].first = "";
00249                 this->usuarios_logados[id].second = "";
00250             }
00251             server.remove_participante(id);
00252             return "O usuário foi removido do servidor!";
00253         }
00254     }
00255     return "Este servidor não existe!";
00256 }
```

4.5.2.10 list_channels()

```
string Sistema::list_channels (
            int id )
```

Lista os canais do servidor que o usuário está visualizando.

Parâmetros

id	um ID válido do usuário logado.

Retorna

string A lista de canais, ou erro caso não seja possível obter a lista.

Definição na linha 329 do arquivo sistema.cpp.

```
00330 {
00331     if (!this->usuarios_logados.count(id))
```

```
00332         return "Você não está logado!";
00333
00334     if (this->usuarios_logados[id].first.empty())
00335         return "Você não está visualizando nenhum servidor!";
00336
00337     for (auto& server : this->servers) {
00338         if (server.get_nome() == this->usuarios_logados[id].first)
00339             return server.imprimir_canais_texto();
00340     }
00341     return "Este servidor não existe!";
00342 }
```

4.5.2.11 list_messages()

```
string Sistema::list_messages (
            int id )
```

Lista as mensagens de um canal.

Parâmetros

id	um ID válido do usuário logado.

Retorna

string a lista de mensagens formatada ou um erro caso não seja possível gerar a lista.

Definição na linha 426 do arquivo sistema.cpp.

```
00427 {
00428     if (!this->usuarios_logados.count(id))
00429         return "Você não está logado!";
00430
00431     if (this->usuarios_logados[id].first.empty())
00432         return "Você não está visualizando nenhum servidor!";
00433
00434     if (this->usuarios_logados[id].second.empty())
00435         return "Você não está visualizando nenhum canal de texto!";
00436
00437     for (auto& server : this->servers) {
00438         if (server.get_nome() == this->usuarios_logados[id].first) {
00439             return server.imprimir_mensagens(this->usuarios_logados[id].second, this->usuarios);
00440         }
00441     }
00442     return "Este servidor não existe!";
00443 }
```

4.5.2.12 list_participants()

```
string Sistema::list_participants (
            int id )
```

Função que lista os participantes de um servidor que esteja sendo visualizado.

Parâmetros

id	de um usuário cadastrado e logado no sistema.

Retorna

string com a lista de usuários no servidor ou erro caso não seja possível gerar a lista.

Definição na linha 258 do arquivo sistema.cpp.

```
00259 {
00260     if (!this->usuarios_logados.count(id))
00261         return "Você não está logado!";
00262 
00263     if (this->usuarios_logados[id].first.empty())
00264         return "O usuário não está visualizando nenhum servidor...";
00265 
00266     for (auto& server : this->servers) {
00267         if (server.get_nome() == this->usuarios_logados[id].first)
00268             return server.imprimir_participantes(this->usuarios);
00269     }
00270     return "Este servidor não existe!";
00271 }
```

4.5.2.13 list_servers()

```
string Sistema::list_servers (
            int id )
```

Função que lista os servidores cadastrados no sistema.

Parâmetros

id	o ID de um usuário cadastrado e logado no sistema.

Retorna

string com a lista de servidores no sistema ou uma mensagem de erro caso não seja possível gerar a lista.

Definição na linha 199 do arquivo sistema.cpp.

```
00200 {
00201     if (this->servers.empty())
00202         return "Nenhum servidor foi criado ainda...";
00203 
00204     stringstream ss;
00205     for (auto& server : this->servers) {
00206         ss « "-> " « server.get_nome() « " [\"" « server.get_descricao() « "\"] ";
00207         ss « "-> Dono do servidor: " « server.get_usuario_dono_id() « endl;
00208     }
00209     return ss.str();
00210 }
```

4.5.2.14 list_users()

```
string Sistema::list_users ( )
```

Função que lista os usuários do sistema e se estão online ou offline.

Retorna

string com a lista e o status dos usuários.

Definição na linha 273 do arquivo sistema.cpp.

```
00274 {
00275     if (this->usuarios.empty())
00276         return "Nenhum usuário encontrado!";
00277 
00278     stringstream ss;
00279     for (auto& user : this->usuarios) {
00280         ss « "#" « user.get_id() « " " « user.get_nome() « " [";
00281 
00282         if (this->usuarios_logados.count(user.get_id())) {
00283             ss « this->usuarios_logados[user.get_id()].first « " / ";
00284             ss « this->usuarios_logados[user.get_id()].second « "] ";
00285             ss « "Atualmente online" « endl;
00286         } else {
00287             ss « " / ] " « "Atualmente offline" « endl;
00288         }
00289     }
00290     return ss.str();
00291 }
```

4.5.2.15 login()

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

Função que realiza o login do usuário com email e senha, retorna uma string de erro ou uma mensagem login bem sucedido. Quando um usuário loga o sistema deve adicionar o usuário na tabela de usuários logados.

Parâmetros

email	o email do usuário, que o usuário insere no login.
senha	a senha inserida pelo usuário

Retorna

string que contém a confirmação do login, ou um erro.

Definição na linha 100 do arquivo sistema.cpp.

```
00101 {
00102     for (auto& user : this->usuarios) {
00103         if (email == user.get_email() && !user.same_pw(senha))
00104             return "Senha incorreta, verifique sua senha e tente novamente!";
00105 
00106         if (email == user.get_email() && user.same_pw(senha)) {
00107             if (this->usuarios_logados.count(user.get_id()))
00108                 return "Este usuário já está logado!";
00109 
00110             this->usuarios_logados[user.get_id()] = make_pair("","");
00111             return "Parabéns, você agora está Online!";
00112         }
00113     }
00114     return "Conta não encontrada! Verifique seu e-mail ou cadastre-se!";
00115 }
```

4.5.2.16 quit()

```
string Sistema::quit ( )
```

Função que termina o programa.

Retorna

uma string que anuncia o encerramento do programa.

Definição na linha 51 do arquivo sistema.cpp.

```
00052 {
00053     return "Encerrando o programa...";
00054 }
```

4.5.2.17 remove_server()

```
string Sistema::remove_server (
            int id,
            const string nome )
```

Função que remove um servidor da lista de servidores.

Parâmetros

id	o ID de um usuário tegistrado e logado no sistema.
nome	o nome de um servidor registrado no sistema.

Retorna

string com a confirmação da eliminação ou mensagem de erro em caso de falha.

Definição na linha 139 do arquivo sistema.cpp.

```
00140 {
00141     if (!this->usuarios_logados.count(id))
00142         return "Você não está logado!";
00143 
00144     auto i = this->servers.begin();
00145     while (i != this->servers.end()) {
00146         if (i->get_nome() == nome) {
00147             if (i->get_usuario_dono_id() != id)
00148                 return "Apenas o dono do servidor pode remover o servidor!";
00149 
00150             for (auto& user: this->usuarios) {
00151                 if (this->usuarios_logados[user.get_id()].first == nome) {
00152                     this->usuarios_logados[user.get_id()].first = "";
00153                     this->usuarios_logados[user.get_id()].second = "";
00154                 }
00155             }
00156             this->servers.erase(i);
00157             return "Servidor removido com sucesso!";
00158         }
00159         ++i;
00160     }
00161     return "Este servidor não existe!";
00162 }
```

4.5.2.18 salvar()

```
void Sistema::salvar ( )
```

Definição na linha 324 do arquivo sistema.cpp.

```
00324                     {
00325     salvarUsuarios();
00326     salvarServidores();
00327 }
```

4.5.2.19 salvarServidores()

```
void Sistema::salvarServidores ( )  [private]
```

Definição na linha 308 do arquivo sistema.cpp.

```
00309 {
00310     ofstream myfile;
00311     myfile.open("servidores.txt", std::ios_base::app);
00312 
00313     int count = 0;
00314     for (auto& server : this->servers){
00315         count++;
00316     }
00317     myfile « "Total de servidores registrados: " « count « endl;
00318 
00319     for (auto& server : this->servers) {
00320         myfile « server.get_usuario_dono_id() « endl « server.get_nome() « endl «
      server.get_descricao() « endl « server.get_convite() « endl;
00321     }
00322 }
```

4.5.2.20 salvarUsuarios()

```
void Sistema::salvarUsuarios ( )  [private]
```

Definição na linha 293 do arquivo sistema.cpp.

```
00294 {
00295     ofstream myfile;
00296     myfile.open("usuarios.txt", std::ios_base::app);
00297 
00298     int count = 0;
00299     for (auto& user : this->usuarios){
00300         count++;
00301     }
00302     myfile « "Total de usuários registrados: " « count « endl;
00303     for (auto& user : this->usuarios) {
00304         myfile « user.get_id() « endl « user.get_nome() « endl « user.get_email() « endl «
      user.get_senha() « endl;
00305     }
00306 }
```

4.5.2.21 send_message()

```
string Sistema::send_message (
            int id,
            const string mensagem )
```

Envia uma mensagem para o canal que o usuário está visualizando.

Parâmetros

id	um ID válido do usuário logado.
mensagem	A mensagem que será enviada.

Retorna

string uma mensagem de confirmação do envio, ou um erro caso não seja possível enviar a mensagem.

Definição na linha 405 do arquivo sistema.cpp.

```
00406 {
00407     if (!this->usuarios_logados.count(id))
00408         return "Você não está logado!";
00409 
00410     if (this->usuarios_logados[id].second.empty())
00411         return "Você não está visualizando nenhum canal de texto!";
00412 
00413     for (auto& server : this->servers) {
00414         if (server.get_nome() == this->usuarios_logados[id].first) {
00415             if (!server.eh_canal_texto(this->usuarios_logados[id].second))
00416                 return "Este canal não existe!";
00417 
00418             Mensagem mensagem(id, conteudo);
00419             server.enviar_mensagem(mensagem, this->usuarios_logados[id].second);
00420             return "Mensagem enviada!";
00421         }
00422     }
00423     return "Este servidor não existe!";
00424 }
```

4.5.2.22 set_server_desc()

```
string Sistema::set_server_desc (
            int id,
            const string nome,
            const string descricao )
```

Função que altera a descrição do servidor para a descrição que o usuário escolher.

Parâmetros

id	o ID de um usuário logado no sistema.
nome	o nome do servidor que terá sua descrição alterada.
descricao	a descrição que será inserida.

Retorna

string que confirma a alteração ou erro caso não seja possível alterar a descrição.

Definição na linha 182 do arquivo sistema.cpp.

```
00183 {
00184     if (!this->usuarios_logados.count(id))
00185         return "Você não está logado!";
00186     for (auto& server : this->servers) {
00187         if (nome == server.get_nome()) {
00188             if (id != server.get_usuario_dono_id())
00189                 return "Você não é o dono do servidor...";
00190             server.set_descricao(descricao);
00191             if (descricao == "")
00192                 return cat(" Descrição do servidor \"", server.get_nome(), "\" removida!");
00193             return cat("Descrição do servidor \"", server.get_nome(),"\" adicionada!");
00194         }
00195     }
00196     return "Este servidor não existe!";
00197 }
```

4.5.2.23 set_server_invite_code()

```
string Sistema::set_server_invite_code (
            int id,
            const string nome,
            const string codigo )
```

Função que altera o código de convite de um servidor.

Parâmetros

id	o ID de um usuário logado no sistema.
nome	o nome de um servidor registrado no sistema
codigo	o código de convite recebido pelo comando.

Retorna

string confirmação da alteração do código de convite ou erro em caso de falha.

Definição na linha 164 do arquivo sistema.cpp.

```
00165 {
00166     if (!this->usuarios_logados.count(id))
00167         return "Você não está logado!";
00168
00169     for (auto& server : this->servers) {
00170         if (nome == server.get_nome()) {
00171             if (id != server.get_usuario_dono_id())
00172                 return "Você não é o dono do servidor...";
00173             server.set_convite(codigo);
00174             if (codigo == "")
00175                 return cat("Código de convite do servidor \"", server.get_nome(), "\" removido! O
      servidor agora é público!");
00176             return cat("Codigo de convite do servidor \"", server.get_nome(),"\" criado! Convide seus
      amigos! :D");
```

```
00177         }
00178     }
00179     return "Este servidor não existe!";
00180 }
```

4.5.3 Campos

4.5.3.1 servers

```
vector<Servidor> Sistema::servers  [private]
```

Definição na linha 13 do arquivo sistema.h.

4.5.3.2 usuarios

```
vector<Usuario> Sistema::usuarios  [private]
```

Definição na linha 14 do arquivo sistema.h.

4.5.3.3 usuarios_logados

```
map<int,pair<string,string> > Sistema::usuarios_logados  [private]
```

Definição na linha 15 do arquivo sistema.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

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

4.6 Referência da Classe Usuario

```
#include <usuario.h>
```

Membros Públicos

- Usuario (int id, string nome, string email, string senha)

  Constructor.
- int get_id ()
- string get_nome ()
- string get_email ()
- string get_senha ()
- bool same_pw (string senha)

Atributos Privados

- int id
- string nome
- string email
- string senha

4.6.1 Descrição detalhada

Definição na linha 7 do arquivo usuario.h.

4.6.2 Construtores e Destrutores

4.6.2.1 Usuario()

```
Usuario::Usuario (
            int id,
            string nome,
            string email,
            string senha )
```

Constructor.

Parâmetros

id	- O id do usuário.
nome	- o nome do usuário.
email	- o e-mail do usuário.
senha	- a senha do usuário.

Definição na linha 6 do arquivo usuario.cpp.

```
00007 {
00008     this->id = id;
00009     this->nome = nome;
00010     this->email = email;
00011     this->senha = senha;
00012 }
```

4.6.3 Documentação das funções

4.6.3.1 get_email()

```
string Usuario::get_email ( )
```

Definição na linha 24 do arquivo usuario.cpp.

```
00025 {
00026     return this->email;
00027 }
```

4.6.3.2 get_id()

```
int Usuario::get_id ( )
```

Retorna

Valor do atributo correspondente.

Definição na linha 14 do arquivo usuario.cpp.

```
00015 {
00016     return this->id;
00017 }
```

4.6.3.3 get_nome()

```
string Usuario::get_nome ( )
```

Definição na linha 19 do arquivo usuario.cpp.

```
00020 {
00021     return this->nome;
00022 }
```

4.6.3.4 get_senha()

```
string Usuario::get_senha ( )
```

Definição na linha 29 do arquivo usuario.cpp.

```
00030 {
00031     return this->senha;
00032 }
```

4.6.3.5 same_pw()

```
bool Usuario::same_pw (
            string senha )
```

Parâmetros

senha	que será verificada

Retorna

True se as senhas forem iguais, false se não.

Definição na linha 34 do arquivo usuario.cpp.

```
00035 {
00036     return this->senha == linha;
00037 }
```

4.6.4 Campos

4.6.4.1 email

```
string Usuario::email  [private]
```

Definição na linha 11 do arquivo usuario.h.

4.6.4.2 id

```
int Usuario::id  [private]
```

Definição na linha 9 do arquivo usuario.h.

4.6.4.3 nome

```
string Usuario::nome  [private]
```

Definição na linha 10 do arquivo usuario.h.

4.6.4.4 senha

```
string Usuario::senha  [private]
```

Definição na linha 12 do arquivo usuario.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

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

Chapter 5

Arquivos

5.1 Referência do Arquivo include/canalt.h

```
#include <string>
#include <vector>
#include ¨usuario.h¨
#include ¨mensagem.h¨
```

Gráfico de dependência de inclusões para canalt.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class CanalTexto

5.2 canalt.h

Ir para a documentação desse arquivo.

```
00001 #ifndef CANALT_H
00002 #define CANALT_H
00003 #include <string>
00004 #include <vector>
00005 #include "usuario.h"
00006 #include "mensagem.h"
00007
00008 using namespace std;
00009
00010 class CanalTexto {
00011 private:
00012     string nome;
00013     vector<Mensagem> mensagens;
00014
00015 public:
00020     CanalTexto(string nome);
00021
00026     string get_nome();
00027
00032     void add_mensagem(Mensagem mensagem);
00033
00039     string listar_mensagens(vector<Usuario>& usuarios);
00040 };
00041
00042 #endif
00043
```

5.3 Referência do Arquivo include/mensagem.h

`#include <string>`

Gráfico de dependência de inclusões para mensagem.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class Mensagem

5.4 mensagem.h

Ir para a documentação desse arquivo.

```
00001 #ifndef MENSAGEM_H
00002 #define MENSAGEM_H
00003 #include <string>
00004
00005 using namespace std;
00006
00007 class Mensagem {
00008 private:
00009     int enviada_por;
00010     string data_hora;
00011     string conteudo;
00012
00013 public:
00019     Mensagem(int enviada_por, string conteudo);
00020
00025     int get_enviada_por();
00026     string get_data_hora();
00027     string get_conteudo();
00028 };
00029
00030 #endif
00031
```

5.5 Referência do Arquivo include/principal.h

`#include <ostream>`
`#include <sstream>`
`#include <istream>`
`#include ¨sistema.h¨`

Gráfico de dependência de inclusões para principal.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class principal

5.6 principal.h

Ir para a documentação desse arquivo.

```
00001 #ifndef PRINCIPAL_H
00002 #define PRINCIPAL_H
00003 #include <ostream>
00004 #include <sstream>
00005 #include <istream>
00006 #include "sistema.h"
00007
00008 using namespace std;
00009
00010 class principal {
00011 private:
00012     Sistema *sistema;
00013     stringstream ss;
00014     bool sair = false;
00015
00016 public:
00017     principal(Sistema& sistema);
00018
00019     string processar_linha(string linha);
00020     void iniciar(istream& in, ostream& out);
00021 };
00022 #endif
```

5.7 Referência do Arquivo include/servidor.h

```
#include <vector>
#include <string>
#include ¨usuario.h¨
#include ¨mensagem.h¨
#include ¨canalt.h¨
```

Gráfico de dependência de inclusões para servidor.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class Servidor

5.8 servidor.h

Ir para a documentação desse arquivo.

```
00001 #ifndef SERVIDOR_H
00002 #define SERVIDOR_H
00003 #include <vector>
00004 #include <string>
00005 #include "usuario.h"
00006 #include "mensagem.h"
00007 #include "canalt.h"
00008
00009 using namespace std;
00010
00011 class Servidor {
00012 private:
00013     int usuario_dono_id;
00014     string nome;
00015     string descricao;
00016     string convite;
00017     vector<int> participantes_id;
00018     vector<CanalTexto> canais_texto;
00019
00020 public:
00026     Servidor(int usuario_dono_id, string nome);
00027
00032     int get_usuario_dono_id();
```

```
00033     string get_nome();
00034     string get_descricao();
00035     string get_convite();
00036 
00041     void set_descricao(string descricao);
00042     void set_convite(string codigo);
00043 
00048     void add_participante(int id);
00049 
00054     void remove_participante(int id);
00055 
00061     string imprimir_participantes(vector<Usuario>& usuarios);
00062 
00068     bool eh_participante(int id);
00069 
00074     void add_canal_texto(CanalTexto& canal);
00075 
00080     void remove_canal_texto(string canal);
00081 
00087     bool eh_canal_texto(string canal);
00088 
00093     string imprimir_canais_texto();
00094 
00100     void enviar_mensagem(Mensagem& mensagem, string nome_canal);
00101 
00108     string imprimir_mensagens(string nome_canal, vector<Usuario>& usuarios);
00109 };
00110 
00111 #endif
```

5.9 Referência do Arquivo include/sistema.h

```
#include <map>
#include <vector>
#include <string>
#include ¨usuario.h¨
#include ¨servidor.h¨
```

Gráfico de dependência de inclusões para sistema.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class Sistema

5.10 sistema.h

Ir para a documentação desse arquivo.

```
00001 #ifndef SISTEMA_H
00002 #define SISTEMA_H
00003 #include <map>
00004 #include <vector>
00005 #include <string>
00006 #include "usuario.h"
00007 #include "servidor.h"
00008 
00009 using namespace std;
00010 
00011 class Sistema {
00012 private:
00013     vector<Servidor> servers; //Vetor que armazena todos os servidores.
00014     vector<Usuario> usuarios; //Vetor que armazena os usuários cadastrados.
00015     map<int,pair<string,string» usuarios_logados; //Vetor que armazena os usuários que logaram no
      sistema.
00016     void salvarUsuarios();
00017     void salvarServidores();
00018 
00019 public:
00020 
```

```
00021     void salvar();
00026     string quit();
00027
00033     string help();
00034
00043     string create_user(const string email, const string senha, const string nome);
00044
00053     string login(const string email, const string senha);
00054
00061     string disconnect(int id);
00062
00069     string create_server(int id, const string nome);
00070
00079     string set_server_invite_code(int id, const string nome, const string codigo);
00080
00089     string set_server_desc(int id, const string nome, const string descricao);
00090
00097     string list_servers(int id);
00098
00106     string remove_server(int id, const string nome);
00107
00116     string enter_server(int id, const string nome, const string codigo);
00117
00125     string leave_server(int id, const string nome);
00126
00133     string list_participants(int id);
00134
00139     string list_users();
00140
00147     string list_channels(int id);
00148
00156     string create_channel(int id, const string nome);
00157
00166     string enter_channel(int id, const string nome);
00167
00174     string leave_channel(int id);
00175
00183     string send_message(int id, const string mensagem);
00184
00191     string list_messages(int id);
00192 };
00193
00194 #endif
```

5.11 Referência do Arquivo include/usuario.h

`#include <string>`

Gráfico de dependência de inclusões para usuario.h: Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:

Estruturas de Dados

- class Usuario

5.12 usuario.h

Ir para a documentação desse arquivo.

```
00001 #ifndef USUARIO_H
00002 #define USUARIO_H
00003 #include <string>
00004
00005 using namespace std;
00006
00007 class Usuario {
00008 private:
00009     int id;
00010     string nome;
00011     string email;
00012     string senha;
00013
```

```
00014 public:
00022     Usuario(int id, string nome, string email, string senha);
00023
00028     int get_id();
00029     string get_nome();
00030     string get_email();
00031     string get_senha();
00032
00037     bool same_pw(string senha);
00038 };
00039
00040 #endif
```

5.13 Referência do Arquivo README.md

5.14 Referência do Arquivo src/canalt.cpp

```
#include <string>
#include <ostream>
#include <vector>
#include <sstream>
#include ¨canalt.h¨
#include ¨mensagem.h¨
#include ¨usuario.h¨
```

Gráfico de dependência de inclusões para canalt.cpp:

5.15 canalt.cpp

Ir para a documentação desse arquivo.

```
00001 #include <string>
00002 #include <ostream>
00003 #include <vector>
00004 #include <sstream>
00005 #include "canalt.h"
00006 #include "mensagem.h"
00007 #include "usuario.h"
00008
00009 using namespace std;
00010
00011 CanalTexto::CanalTexto(string nome)
00012 {
00013     this->nome = nome;
00014 }
00015
00016 string CanalTexto::get_nome()
00017 {
00018     return this->nome;
00019 }
00020
00021 void CanalTexto::add_mensagem(Mensagem mensagem)
00022 {
00023     this->mensagens.push_back(mensagem);
00024 }
00025
00026 string CanalTexto::listar_mensagens(vector<Usuario>& usuarios)
00027 {
00028     stringstream ss;
00029     for (auto& m : this->mensagens) {
00030         for (auto& u : usuarios) {
00031             if (m.get_enviada_por() == u.get_id()) {
00032                 ss « u.get_nome() « " " « m.get_data_hora();
00033                 ss  « ": " « m.get_conteudo() « endl;
00034                 break;
00035             }
00036         }
00037     }
00038
00039     if (ss.str().empty())
00040         return "Nenhuma mensagem foi enviada nesse canal ainda, inicie uma conversa! :D";
00041     return ss.str();
00042 }
```

5.16 Referência do Arquivo src/concord_main.cpp

```
#include <iostream>
#include <list>
#include <vector>
#include <string>
#include ¨sistema.h¨
#include ¨principal.h¨
```

Gráfico de dependência de inclusões para concord_main.cpp:

Funções

- int main ()

5.16.1 Funções

5.16.1.1 main()

```
int main ( )
```

Definição na linha 10 do arquivo concord_main.cpp.

```
00011 {
00012     cout « "Bem vindo! Digite um comando para começar (ou 'help' para ajuda): ";
00013
00014     Sistema sistema; //Inicializando o Sistema
00015
00016     principal principal(sistema); //Inicializando o processo principal do sistema.
00017
00018     principal.iniciar(cin, cout); //Esse comando lê o cin e o cout, executando o método correto e
      imprimindo as mensagens em seguida.
00019
00020     return 0;
00021 }
```

5.17 concord_main.cpp

Ir para a documentação desse arquivo.

```
00001 #include <iostream>
00002 #include <list>
00003 #include <vector>
00004 #include <string>
00005 #include "sistema.h"
00006 #include "principal.h"
00007
00008 using namespace std;
00009
00010 int main()
00011 {
00012     cout « "Bem vindo! Digite um comando para começar (ou 'help' para ajuda): ";
00013
00014     Sistema sistema; //Inicializando o Sistema
00015
00016     principal principal(sistema); //Inicializando o processo principal do sistema.
00017
00018     principal.iniciar(cin, cout); //Esse comando lê o cin e o cout, executando o método correto e
      imprimindo as mensagens em seguida.
00019
00020     return 0;
00021 }
```

5.18 Referência do Arquivo src/mensagem.cpp

```
#include <iomanip>
#include <ctime>
#include <string>
#include <sstream>
#include ¨mensagem.h¨
```

Gráfico de dependência de inclusões para mensagem.cpp:

5.19 mensagem.cpp

Ir para a documentação desse arquivo.

```
00001 #include <iomanip>
00002 #include <ctime>
00003 #include <string>
00004 #include <sstream>
00005 #include "mensagem.h"
00006
00007 using namespace std;
00008
00009
00010 int Mensagem::get_enviada_por()
00011 {
00012     return this->enviada_por;
00013 }
00014
00015 string Mensagem::get_data_hora()
00016 {
00017     return this->data_hora;
00018 }
00019
00020 string Mensagem::get_conteudo()
00021 {
00022     return this->conteudo;
00023 }
00024
00025 Mensagem::Mensagem(int enviada_por, string conteudo)
00026 {
00027     this->enviada_por = enviada_por;
00028     this->conteudo = conteudo;
00029
00030     time_t t = time(0);
00031     tm* now = localtime(&t);
00032     stringstream ss;
00033     ss.fill('0');
00034     ss « "[" « setw(2) « now->tm_mday « "/" « setw(2) « now->tm_mon+1;
00035     ss « "/" « setw(2) « now->tm_year-100;
00036     ss « " às " « setw(2) « now->tm_hour « ":" « setw(2) « now->tm_min « "]";
00037     this->data_hora = ss.str();
00038 }
```

5.20 Referência do Arquivo src/principal.cpp

```
#include <istream>
#include <ostream>
#include <iostream>
#include <sstream>
#include <queue>
#include ¨principal.h¨
```

Gráfico de dependência de inclusões para principal.cpp:

Funções

- string resto_de (istringstream &ss)

5.20.1 Funções

5.20.1.1 resto_de()

```
string resto_de (
            istringstream & ss )
```

Definição na linha 11 do arquivo principal.cpp.

```
00012 {
00013     string resto;
00014     getline(ss, resto, '\0');
00015     if (resto != "" && (resto[0] == ' ' || resto[0] == '\t')) {
00016         resto = resto.substr(1); // Eliminar o primeiro caractere caso ele seja um espaço vazio.
00017     }
00018     return resto;
00019 }
```

5.21 principal.cpp

Ir para a documentação desse arquivo.

```
00001 #include <istream>
00002 #include <ostream>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <queue>
00006 #include "principal.h"
00007
00008 using namespace std;
00009
00010 // Função que recebe um istringstream e lê todo texto restante até o fim da linha
00011 string resto_de(istringstream &ss)
00012 {
00013     string resto;
00014     getline(ss, resto, '\0');
00015     if (resto != "" && (resto[0] == ' ' || resto[0] == '\t')) {
00016         resto = resto.substr(1); // Eliminar o primeiro caractere caso ele seja um espaço vazio.
00017     }
00018     return resto;
00019 }
00020
00021 // Recebe uma referência ao sistema que vai operar e guarda o seu endereço para executar as operações.
00022 principal::principal(Sistema &sistema)
00023 {
00024     this->sair = false;
00025     this->sistema = &sistema;
00026 }
00027
00028 // Método que recebe os comandos do usuário e os executa, usando o método equivalente no sistema.
00029 string principal::processar_linha(string linha)
00030 {
00031     istringstream buf(linha);
00032     string Cchooser;
00033     buf » Cchooser;
00034
00035     if (Cchooser.empty()) {
00036         return "Digite um comando!";
00037     }
00038     if (Cchooser == "quit" ) {
00039         this->sair = true;
00040         return sistema->quit();
00041     }
00042     else if(Cchooser == "salvar"){
00043          sistema->salvar();
00044          return "Arquivos de servidores e usuários salvos!";
00045     }
00046     else if (Cchooser == "help") {
00047         return sistema->help();
00048     }
00049     else if (Cchooser == "create-user") {
00050         string email, senha, nome;
00051         buf » email;
00052         buf » senha;
00053         nome = resto_de(buf);
00054         if (email.empty() || senha.empty() || nome.empty())
00055             return "»>Uso: create-user EMAIL SENHA NOME";
00056         return sistema->create_user(email, senha, nome);
```

```
00057     }
00058     else if (Cchooser == "login") {
00059         string email, senha;
00060         buf » email;
00061         buf » senha;
00062         if (email.empty() || senha.empty())
00063             return "»>Uso: login EMAIL SENHA";
00064         return sistema->login(email, senha);
00065     }
00066     else if (Cchooser == "list-users") {
00067         return sistema->list_users();
00068     }
00069 
00070     int id;
00071     if (!(buf » id)) {
00072         return "Este comando precisa ser precedido de um ID [" + Cchooser + "]...";
00073     }
00074 
00075     if (Cchooser == "disconnect") {
00076         return sistema->disconnect(id);
00077     }
00078     else if (Cchooser == "create-server") {
00079         string nome;
00080         buf » nome;
00081         if (nome.empty())
00082             return "»>Uso: create-server ID NOME";
00083         return sistema->create_server(id, nome);
00084     }
00085     else if (Cchooser == "set-server-desc") {
00086         string nome, descricao;
00087         buf » nome;
00088         descricao = resto_de(buf);
00089         if (nome.empty())
00090             return "»>Uso: set-server-desc ID NOME [DESCRICAO...]";
00091         return sistema->set_server_desc(id, nome, descricao);
00092     }
00093     else if (Cchooser == "set-server-invite-code") {
00094         string nome, codigo;
00095         buf » nome;
00096         buf » codigo;
00097         if (nome.empty())
00098             return "»>Uso: set-server-invite-code ID NOME [CODIGO]";
00099         return sistema->set_server_invite_code(id, nome, codigo);
00100     }
00101     else if (Cchooser == "list-servers") {
00102         return sistema->list_servers(id);
00103     }
00104     else if (Cchooser == "remove-server") {
00105         string nome;
00106         buf » nome;
00107         if (nome.empty())
00108             return "»>Uso: remove-server ID NOME";
00109         return sistema->remove_server(id, nome);
00110     }
00111     else if (Cchooser == "enter-server") {
00112         string nome, codigo;
00113         buf » nome;
00114         buf » codigo;
00115         if (nome.empty())
00116             return "»>Uso: enter-server ID NOME [CODIGO]";
00117         return sistema->enter_server(id, nome, codigo);
00118     }
00119     else if (Cchooser == "leave-server") {
00120         string nome;
00121         buf » nome;
00122         if (nome.empty())
00123             return "»>Uso: leave-server ID NOME";
00124         return sistema->leave_server(id, nome);
00125     }
00126     else if (Cchooser == "list-participants") {
00127         return sistema->list_participants(id);
00128     }
00129     else if (Cchooser == "list-channels") {
00130         return sistema->list_channels(id);
00131     }
00132     else if (Cchooser == "create-channel") {
00133         string nome;
00134         buf » nome;
00135         if (nome.empty())
00136             return "»>Uso: create-channel ID NOME";
00137         return sistema->create_channel(id, nome);
00138     }
00139     else if (Cchooser == "enter-channel") {
00140         string nome;
00141         buf » nome;
00142         if (nome.empty())
00143             return "»>Uso: enter-channel ID NOME";
```

```
00144         return sistema->enter_channel(id, nome);
00145     }
00146     else if (Cchooser == "leave-channel") {
00147         return sistema->leave_channel(id);
00148     }
00149     else if (Cchooser == "send-message") {
00150         string mensagem;
00151         mensagem = resto_de(buf);
00152         if (mensagem.empty())
00153             return "»>Uso: send-message ID MESSAGE...";
00154         return sistema->send_message(id, mensagem);
00155     }
00156     else if (Cchooser == "list-messages") {
00157         return sistema->list_messages(id);
00158     }
00159     else {
00160         return "»>Comando não reconhecido [" + Cchooser + "]...";
00161     }
00162 }
00163
00168 void principal::iniciar(istream &inputStream, ostream &outputStream)
00169 {
00170     string saida, linha;
00171     this->sair = false;
00172     while (!this->sair) {
00173         if (getline(inputStream, linha)) {
00174             saida = processar_linha(linha);
00175             outputStream « saida « endl;
00176         }
00177     }
00178 }
```

5.22 Referência do Arquivo src/servidor.cpp

```
#include <string>
#include <ostream>
#include <sstream>
#include <algorithm>
#include ¨servidor.h¨
#include ¨mensagem.h¨
#include ¨canalt.h¨
```

Gráfico de dependência de inclusões para servidor.cpp:

5.23 servidor.cpp

Ir para a documentação desse arquivo.

```
00001 #include <string>
00002 #include <ostream>
00003 #include <sstream>
00004 #include <algorithm>
00005 #include "servidor.h"
00006 #include "mensagem.h"
00007 #include "canalt.h"
00008
00009 using namespace std;
00010
00011 Servidor::Servidor(int dono_id, string nome)
00012 {
00013     this->nome = nome;
00014     this->usuario_dono_id = dono_id;
00015     this->descricao = "";
00016     this->convite = "";
00017     this->participantes_id.push_back(dono_id);
00018 }
00019
00020 string Servidor::get_nome()
00021 {
00022     return this->nome;
00023 }
00024
00025 int Servidor::get_usuario_dono_id()
00026 {
```

```
00027     return this->usuario_dono_id;
00028 }
00029
00030 string Servidor::get_descricao()
00031 {
00032     return this->descricao;
00033 }
00034
00035 string Servidor::get_convite()
00036 {
00037     return this->convite;
00038 }
00039
00040 void Servidor::set_descricao(string descricao)
00041 {
00042     this->descricao = descricao;
00043 }
00044
00045 void Servidor::set_convite(string convite)
00046 {
00047     this->convite = convite;
00048 }
00049
00050 void Servidor::add_participante(int id)
00051 {
00052     this->participantes_id.push_back(id);
00053 }
00054
00055 void Servidor::remove_participante(int id)
00056 {
00057     auto i = this->participantes_id.begin();
00058     while (i != this->participantes_id.end()) {
00059         if (*i == id) {
00060             this->participantes_id.erase(i);
00061             return;
00062         }
00063         ++i;
00064     }
00065 }
00066
00067 string Servidor::imprimir_participantes(vector<Usuario>& usuarios)
00068 {
00069     stringstream ss;
00070     for (auto& user : usuarios) {
00071         if (this->eh_participante(user.get_id()))
00072             ss « "-" « user.get_id() « " " « user.get_nome() « endl;
00073     }
00074
00075     if (ss.str().empty())
00076         return "Ninguém está nesse servidor ainda, entre e convide seus amigos!";
00077     return ss.str();
00078 }
00079
00080 bool Servidor::eh_participante(int id)
00081 {
00082     for (auto& p_id : this->participantes_id) {
00083         if (p_id == id)
00084             return true;
00085     }
00086     return false;
00087 }
00088
00089 void Servidor::add_canal_texto(CanalTexto& canal)
00090 {
00091     this->canais_texto.push_back(canal);
00092 }
00093
00094 void Servidor::remove_canal_texto(string canal)
00095 {
00096     auto contador = this->canais_texto.begin();
00097     while (contador != this->canais_texto.end()) {
00098         //Caso o nome seja encontrado na lista de canais, ele é deletado.
00099         if (contador->get_nome() == canal) {
00100             this->canais_texto.erase(contador);
00101             return;
00102         }
00103         ++contador;
00104     }
00105 }
00106
00107 bool Servidor::eh_canal_texto(string canal)
00108 {
00109     for (auto& textchannel : this->canais_texto) {
00110         if (textchannel.get_nome() == canal)
00111             return true;
00112     }
00113     return false;
```

```
00114 }
00115
00116 string Servidor::imprimir_canais_texto()
00117 {
00118     stringstream ss;
00119     for (auto& textchannel : this->canais_texto) {
00120         ss « "-> " « textchannel.get_nome() « endl;
00121     }
00122
00123     if (ss.str().empty())
00124         return "Não existem canais de texto neste servidor, caso seja o dono, experimente adicionar
      um!";
00125     return ss.str();
00126 }
00127
00128 void Servidor::enviar_mensagem(Mensagem& mensagem, string nome_canal)
00129 {
00130     for (auto& textchannel : this->canais_texto) {
00131         if (textchannel.get_nome() == nome_canal)
00132             textchannel.add_mensagem(mensagem);
00133     }
00134 }
00135
00136 string Servidor::imprimir_mensagens(string canal_texto, vector<Usuario>& usuarios)
00137 {
00138     for (auto& textchannel : this->canais_texto) {
00139         if (canal_texto == textchannel.get_nome())
00140             return textchannel.listar_mensagens(usuarios);
00141     }
00142     return "Este canal de texto não existe!";
00143 }
00144
00145
00146
```

5.24 Referência do Arquivo src/sistema.cpp

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <regex>
#include <fstream>
#include ¨sistema.h¨
#include ¨servidor.h¨
#include ¨usuario.h¨
```

Gráfico de dependência de inclusões para sistema.cpp:

Funções

- template<typename StreamableT >
  string cat (StreamableT s)
- template<typename StreamableHead , typename... StreamableTail>
  string cat (StreamableHead h, StreamableTail... t)
- bool email_validate (string email)

  Valida o e-mail inserido.

5.24.1 Funções

5.24.1.1 cat() [1/2]

```
template<typename StreamableHead , typename...  StreamableTail>
string cat (
```

```
            StreamableHead h,
            StreamableTail...  t )
```

Concatena diversas variaveis ¨Streamable¨ recursivamente.

Definição na linha 28 do arquivo sistema.cpp.

```
00029 {
00030     stringstream ss;
00031     ss « h;
00032     return ss.str() + cat(t...);
00033 }
```

5.24.1.2 cat() [2/2]

```
template<typename StreamableT >
string cat (
            StreamableT s )
```

Concatena duas strings ¨Streamable¨

Definição na linha 17 do arquivo sistema.cpp.

```
00018 {
00019     stringstream ss;
00020     ss « s;
00021     return ss.str();
00022 }
```

5.24.1.3 email_validate()

```
bool email_validate (
            string email )
```

Valida o e-mail inserido.

Parâmetros

email	recebe o e-mail digitado pelo usuario

Retorna

true caso o e-mail seja valido

false caso o e-mail seja invalido

Definição na linha 42 do arquivo sistema.cpp.

```
00043 {
00044     const regex email_regex ("\\w+@\\w+\\.[\\w\\.]+"); //cria um formato para o e-mail
00045     return regex_match(email, email_regex);
00046 }
```

5.25 sistema.cpp

Ir para a documentação desse arquivo.

```
00001 #include <iostream>
00002 #include <sstream>
00003 #include <algorithm>
```

```
00004 #include <vector>
00005 #include <regex>
00006 #include <fstream>
00007 #include "sistema.h"
00008 #include "servidor.h"
00009 #include "usuario.h"
00010
00011 using namespace std;
00012
00016 template <typename StreamableT>
00017 string cat(StreamableT s)
00018 {
00019     stringstream ss;
00020     ss « s;
00021     return ss.str();
00022 }
00023
00027 template <typename StreamableHead, typename... StreamableTail>
00028 string cat(StreamableHead h, StreamableTail... t)
00029 {
00030     stringstream ss;
00031     ss « h;
00032     return ss.str() + cat(t...);
00033 }
00034
00042 bool email_validate(string email)
00043 {
00044     const regex email_regex ("\\w+@\\w+\\.[\\w\\.]+"); //cria um formato para o e-mail
00045     return regex_match(email, email_regex);
00046 }
00047
00048
00049 // Comandos do programa.
00050
00051 string Sistema::quit()
00052 {
00053     return "Encerrando o programa...";
00054 }
00055
00056 string Sistema::help()
00057 {
00058     string helpmi = "\n_____________________________Lista de comandos_____________________________\n\n ---
      "
00059                     "quit - Encerra o programa.\n --- "
00060                     "create-user <E-mail> <Senha_sem_espaços> <Nome do usuário> - Registra um novo
      usuário. \n --- "
00061                     "login <E-mail> <senha> - Loga o usuário no sistema.\n --- "
00062                     "disconnect - Desconecta o usuário atual.\n --- "
00063                     "create-server <id> <nome-do-servidor> - Cria um novo servidor.\n --- "
00064                     "set-server-desc <id> <nome-do-servidor> <descrição> - Altera a descrição de um
      servidor que você é o dono.\n --- "
00065                     "set-server-invite-code <id> <nome-do-servidor> <código-desejado> - Adiciona um
      código de convite a um servidor, tornando-o privado.\n --- "
00066                     "list-servers <id> - Lista todos os servidores criados.\n --- "
00067                     "remove-server <id> <nome-do-servidor> - Elimina um servidor.\n --- "
00068                     "enter-server <nome-do-servidor> <id> <nome-do-servidor> <código-de-convite> -
      Entra em um servidor, se ele for público, o código de convite não é necessário.\n --- "
00069                     "leave-server <id> <nome-do-servidor> - Quando estiver dentro do servidor,
      desconecte-se dele.\n --- "
00070                     "list-participants <id> - Lista todos os participantes de um servidor.\n --- "
00071                     "list-users <id> - Lista os usuários do sistema e exibe seus status.\n --- "
00072                     "create-channel <id> <nome> - Cria um novo canal dentro de um servidor.\n --- "
00073                     "list-channels <id> - Lista os canais do servidor.\n --- "
00074                     "enter-channel <id> - Entra em um canal existente.\n --- "
00075                     "leave-channel <id> - Sai do canal.\n --- "
00076                     "send-message <id> <mensagem> - Envia uma mensagem no canal atual.\n --- "
00077                     "list-messages <id> - Lista todas as mensagens do canal.\n"
00078                     "salvar - Salve os usuários e servidores registrados em arquivos.\n\n"
00079                     "_____________________________________________________\n";
00080     return helpmi;
00081 }
00082
00083 string Sistema::create_user(const string email, const string senha, const string nome)
00084 {
00085     if(!email_validate(email))
00086         return "Email inválido!";
00087     for (auto& user : this->usuarios) {
00088         if (email == user.get_email())
00089             return "Já existe uma conta registrada com esse e-mail! Tente logar!";
00090         if (nome == user.get_nome())
00091             return "Este nome de usuário já foi utilizado, tente outro!";
00092     }
00093
00094     int id = this->usuarios.size() + 1;
00095     Usuario usuario(id, nome, email, senha);
00096     this->usuarios.push_back(usuario);
00097     return cat("Novo usuário registrado! Seu ID: #", id);
```

```
00098 }
00099
00100 string Sistema::login(const string email, const string senha)
00101 {
00102     for (auto& user : this->usuarios) {
00103         if (email == user.get_email() && !user.same_pw(senha))
00104             return "Senha incorreta, verifique sua senha e tente novamente!";
00105
00106         if (email == user.get_email() && user.same_pw(senha)) {
00107             if (this->usuarios_logados.count(user.get_id()))
00108                 return "Este usuário já está logado!";
00109
00110             this->usuarios_logados[user.get_id()] = make_pair("","");
00111             return "Parabéns, você agora está Online!";
00112         }
00113     }
00114     return "Conta não encontrada! Verifique seu e-mail ou cadastre-se!";
00115 }
00116
00117 string Sistema::disconnect(int id)
00118 {
00119     if (!this->usuarios_logados.count(id))
00120         return "Este usuário já está desconectado...";
00121
00122     this->usuarios_logados.erase(id);
00123     return "Usuário offline!";
00124 }
00125
00126 string Sistema::create_server(int id, const string nome)
00127 {
00128     if (!this->usuarios_logados.count(id))
00129         return "Você não está logado!";
00130     for (auto& server : this->servers) {
00131         if (nome == server.get_nome())
00132             return "Já existe um servidor com este nome!";
00133     }
00134     Servidor servidor(id, nome);
00135     this->servers.push_back(servidor);
00136     return "Servidor criado com sucesso!";
00137 }
00138
00139 string Sistema::remove_server(int id, const string nome)
00140 {
00141     if (!this->usuarios_logados.count(id))
00142         return "Você não está logado!";
00143
00144     auto i = this->servers.begin();
00145     while (i != this->servers.end()) {
00146         if (i->get_nome() == nome) {
00147             if (i->get_usuario_dono_id() != id)
00148                 return "Apenas o dono do servidor pode remover o servidor!";
00149
00150             for (auto& user: this->usuarios) {
00151                 if (this->usuarios_logados[user.get_id()].first == nome) {
00152                     this->usuarios_logados[user.get_id()].first = "";
00153                     this->usuarios_logados[user.get_id()].second = "";
00154                 }
00155             }
00156             this->servers.erase(i);
00157             return "Servidor removido com sucesso!";
00158         }
00159         ++i;
00160     }
00161     return "Este servidor não existe!";
00162 }
00163
00164 string Sistema::set_server_invite_code(int id, const string nome, const string codigo)
00165 {
00166     if (!this->usuarios_logados.count(id))
00167         return "Você não está logado!";
00168
00169     for (auto& server : this->servers) {
00170         if (nome == server.get_nome()) {
00171             if (id != server.get_usuario_dono_id())
00172                 return "Você não é o dono do servidor...";
00173             server.set_convite(codigo);
00174             if (codigo == "")
00175                 return cat("Código de convite do servidor \"", server.get_nome(), "\" removido! O
      servidor agora é público!");
00176             return cat("Codigo de convite do servidor \"", server.get_nome(),"\" criado! Convide seus
      amigos! :D");
00177         }
00178     }
00179     return "Este servidor não existe!";
00180 }
00181
00182 string Sistema::set_server_desc(int id, const string nome, const string descricao)
```

```
00183 {
00184     if (!this->usuarios_logados.count(id))
00185         return "Você não está logado!";
00186     for (auto& server : this->servers) {
00187         if (nome == server.get_nome()) {
00188             if (id != server.get_usuario_dono_id())
00189                 return "Você não é o dono do servidor...";
00190             server.set_descricao(descricao);
00191             if (descricao == "")
00192                 return cat(" Descrição do servidor \"", server.get_nome(), "\" removida!");
00193             return cat("Descrição do servidor \"", server.get_nome(),"\" adicionada!");
00194         }
00195     }
00196     return "Este servidor não existe!";
00197 }
00198
00199 string Sistema::list_servers(int id)
00200 {
00201     if (this->servers.empty())
00202         return "Nenhum servidor foi criado ainda...";
00203
00204     stringstream ss;
00205     for (auto& server : this->servers) {
00206         ss « "-> " « server.get_nome() « " [\"" « server.get_descricao() « "\"] ";
00207         ss « "-> Dono do servidor: " « server.get_usuario_dono_id() « endl;
00208     }
00209     return ss.str();
00210 }
00211
00212 string Sistema::enter_server(int id, const string nome, const string codigo)
00213 {
00214     if (!this->usuarios_logados.count(id))
00215         return "Você não está logado!";
00216
00217     for (auto& server : this->servers) {
00218         if (nome == server.get_nome()) {
00219             if (this->usuarios_logados[id].first == nome)
00220                 return "Este usuário já está neste servidor!";
00221             if (id != server.get_usuario_dono_id() && codigo != server.get_convite())
00222                 return "Código de convite inválido, você o digitou corretamente?";
00223             this->usuarios_logados[id].first = nome;
00224             this->usuarios_logados[id].second = "";
00225             if (server.eh_participante(id))
00226                 return "Você entrou no servidor com sucesso!";
00227             server.add_participante(id);
00228             return "Você entrou no servidor com sucesso! Bem-vindo!";
00229         }
00230     }
00231     return "Este servidor não existe!";
00232 }
00233
00234 string Sistema::leave_server(int id, const string nome)
00235 {
00236     if (!this->usuarios_logados.count(id))
00237         return "Você não está logado!";
00238
00239     for (auto& server : this->servers) {
00240         if (nome == server.get_nome()) {
00241             if (id == server.get_usuario_dono_id())
00242                 return "»Você não pode remover o dono do servidor!";
00243
00244             if (!server.eh_participante(id))
00245                 return "Este usuário não participa deste servidor...";
00246
00247             if (this->usuarios_logados[id].first == nome) {
00248                 this->usuarios_logados[id].first = "";
00249                 this->usuarios_logados[id].second = "";
00250             }
00251             server.remove_participante(id);
00252             return "O usuário foi removido do servidor!";
00253         }
00254     }
00255     return "Este servidor não existe!";
00256 }
00257
00258 string Sistema::list_participants(int id)
00259 {
00260     if (!this->usuarios_logados.count(id))
00261         return "Você não está logado!";
00262
00263     if (this->usuarios_logados[id].first.empty())
00264         return "O usuário não está visualizando nenhum servidor...";
00265
00266     for (auto& server : this->servers) {
00267         if (server.get_nome() == this->usuarios_logados[id].first)
00268             return server.imprimir_participantes(this->usuarios);
00269     }
```

```
00270     return "Este servidor não existe!";
00271 }
00272
00273 string Sistema::list_users()
00274 {
00275     if (this->usuarios.empty())
00276         return "Nenhum usuário encontrado!";
00277
00278     stringstream ss;
00279     for (auto& user : this->usuarios) {
00280         ss « "#" « user.get_id() « " " « user.get_nome() « " [";
00281
00282         if (this->usuarios_logados.count(user.get_id())) {
00283             ss « this->usuarios_logados[user.get_id()].first « " / ";
00284             ss « this->usuarios_logados[user.get_id()].second « "] ";
00285             ss « "Atualmente online" « endl;
00286         } else {
00287             ss « " / ] " « "Atualmente offline" « endl;
00288         }
00289     }
00290     return ss.str();
00291 }
00292
00293 void Sistema::salvarUsuarios()
00294 {
00295     ofstream myfile;
00296     myfile.open("usuarios.txt", std::ios_base::app);
00297
00298     int count = 0;
00299     for (auto& user : this->usuarios){
00300         count++;
00301     }
00302     myfile « "Total de usuários registrados: " « count « endl;
00303     for (auto& user : this->usuarios) {
00304         myfile « user.get_id() « endl « user.get_nome() « endl « user.get_email() « endl «
      user.get_senha() « endl;
00305     }
00306 }
00307
00308 void Sistema::salvarServidores()
00309 {
00310     ofstream myfile;
00311     myfile.open("servidores.txt", std::ios_base::app);
00312
00313     int count = 0;
00314     for (auto& server : this->servers){
00315         count++;
00316     }
00317     myfile « "Total de servidores registrados: " « count « endl;
00318
00319     for (auto& server : this->servers) {
00320         myfile « server.get_usuario_dono_id() « endl « server.get_nome() « endl «
      server.get_descricao() « endl « server.get_convite() « endl;
00321     }
00322 }
00323
00324 void Sistema::salvar(){
00325     salvarUsuarios();
00326     salvarServidores();
00327 }
00328
00329 string Sistema::list_channels(int id)
00330 {
00331     if (!this->usuarios_logados.count(id))
00332         return "Você não está logado!";
00333
00334     if (this->usuarios_logados[id].first.empty())
00335         return "Você não está visualizando nenhum servidor!";
00336
00337     for (auto& server : this->servers) {
00338         if (server.get_nome() == this->usuarios_logados[id].first)
00339             return server.imprimir_canais_texto();
00340     }
00341     return "Este servidor não existe!";
00342 }
00343
00344 string Sistema::create_channel(int id, const string nome)
00345 {
00346     if (!this->usuarios_logados.count(id))
00347         return "Você não está logado!";
00348
00349     if (this->usuarios_logados[id].first.empty())
00350         return "Você não está visualizando nenhum servidor!";
00351
00352     for (auto& server : this->servers) {
00353         if (server.get_nome() == this->usuarios_logados[id].first) {
00354             if (server.eh_canal_texto(nome))
```

```
00355                 return "Já existe um canal com este nome!";
00356
00357             CanalTexto canal(nome);
00358             server.add_canal_texto(canal);
00359             return "Canal criado com sucesso!";
00360         }
00361     }
00362     return "Você não é o dono do servidor...";
00363 }
00364
00365 string Sistema::enter_channel(int id, const string nome)
00366 {
00367     if (!this->usuarios_logados.count(id))
00368         return "Você não está logado!";
00369
00370     if (this->usuarios_logados[id].first.empty())
00371         return "Você não está visualizando nenhum servidor!";
00372
00373     if (!this->usuarios_logados[id].second.empty())
00374         return "Você já está visualizando um canal de texto!";
00375
00376     for (auto& server : this->servers) {
00377         if (server.get_nome() == this->usuarios_logados[id].first) {
00378             if (server.eh_canal_texto(nome)){
00379                 this->usuarios_logados[id].second = nome;
00380                 return cat("O usuário ", nome, " entrou no canal!");
00381             }
00382         }
00383     }
00384     return "Canal não encontrado!";
00385 }
00386
00387 string Sistema::leave_channel(int id)
00388 {
00389     if (!this->usuarios_logados.count(id))
00390         return "Você não está logado!";
00391
00392     if (this->usuarios_logados[id].first.empty())
00393         return "Você não está visualizando nenhum servidor!";
00394
00395     if (this->usuarios_logados[id].second.empty())
00396         return "Você não está visualizando nenhum canal de texto!";
00397
00398     string c;
00399     c = this->usuarios_logados[id].second;
00400     this->usuarios_logados[id].second = "";
00401     return cat("Você saiu do canal ", c, "!");
00402
00403 }
00404
00405 string Sistema::send_message(int id, const string conteudo)
00406 {
00407     if (!this->usuarios_logados.count(id))
00408         return "Você não está logado!";
00409
00410     if (this->usuarios_logados[id].second.empty())
00411         return "Você não está visualizando nenhum canal de texto!";
00412
00413     for (auto& server : this->servers) {
00414         if (server.get_nome() == this->usuarios_logados[id].first) {
00415             if (!server.eh_canal_texto(this->usuarios_logados[id].second))
00416                 return "Este canal não existe!";
00417
00418             Mensagem mensagem(id, conteudo);
00419             server.enviar_mensagem(mensagem, this->usuarios_logados[id].second);
00420             return "Mensagem enviada!";
00421         }
00422     }
00423     return "Este servidor não existe!";
00424 }
00425
00426 string Sistema::list_messages(int id)
00427 {
00428     if (!this->usuarios_logados.count(id))
00429         return "Você não está logado!";
00430
00431     if (this->usuarios_logados[id].first.empty())
00432         return "Você não está visualizando nenhum servidor!";
00433
00434     if (this->usuarios_logados[id].second.empty())
00435         return "Você não está visualizando nenhum canal de texto!";
00436
00437     for (auto& server : this->servers) {
00438         if (server.get_nome() == this->usuarios_logados[id].first) {
00439             return server.imprimir_mensagens(this->usuarios_logados[id].second, this->usuarios);
00440         }
00441     }
```

```
00442     return "Este servidor não existe!";
00443 }
```

5.26 Referência do Arquivo src/usuario.cpp

```
#include <string>
#include ¨usuario.h¨
```

Gráfico de dependência de inclusões para usuario.cpp:

5.27 usuario.cpp

Ir para a documentação desse arquivo.

```
00001 #include <string>
00002 #include "usuario.h"
00003
00004 using namespace std;
00005
00006 Usuario::Usuario(int id, string nome, string email, string senha)
00007 {
00008     this->id = id;
00009     this->nome = nome;
00010     this->email = email;
00011     this->senha = senha;
00012 }
00013
00014 int Usuario::get_id()
00015 {
00016     return this->id;
00017 }
00018
00019 string Usuario::get_nome()
00020 {
00021     return this->nome;
00022 }
00023
00024 string Usuario::get_email()
00025 {
00026     return this->email;
00027 }
00028
00029 string Usuario::get_senha()
00030 {
00031     return this->senha;
00032 }
00033
00034 bool Usuario::same_pw(string linha)
00035 {
00036     return this->senha == linha;
00037 }
```