Concord

v0.2

Gerado por Doxygen 1.9.1

Chapter 1

Concord

Projeto de ITP

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.

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.

```
12 {
13     this->nome = nome;
14 }
```

4.1.3 Funções membros

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.

```
22 {
23     this->mensagens.push_back(mensagem);
24 }
```

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.

```
17 {
18     return this->nome;
19 }
```

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.

```
27 {
28     stringstream ss;
29     for (auto& m : this->mensagens) {
30         for (auto& u : usuarios) {
31             if (m.get_enviada_por() == u.get_id()) {
32                 ss « u.get_nome() « " " « m.get_data_hora();
33                 ss  « ": " « m.get_conteudo() « endl;
34                 break;
35             }
36         }
37     }
38
39     if (ss.str().empty())
40         return "Nenhuma mensagem foi enviada nesse canal ainda, inicie uma conversa! :D";
41     return ss.str();
42 }
```

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.

```
26 {
27     this->enviada_por = enviada_por;
28     this->conteudo = conteudo;
29
30     time_t t = time(0);
31     tm* now = localtime(&t);
32     stringstream ss;
33     ss.fill('0');
34     ss << "[" << setw(2) << now->tm_mday << "/" << setw(2) << now->tm_mon+1;
35     ss << "/" << setw(2) << now->tm_year-100;
36     ss << " às " << setw(2) << now->tm_hour << ":" << setw(2) << now->tm_min << "]";
37     this->data_hora = ss.str();
38 }
```

4.2.3 Funções membros

4.2.3.1 get_conteudo()

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

Definição na linha 20 do arquivo mensagem.cpp.

```
21 {
22     return this->conteudo;
23 }
```

4.2.3.2 get_data_hora()

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

Definição na linha 15 do arquivo mensagem.cpp.

```
16 {
17     return this->data_hora;
18 }
```

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.

```
11 {
12     return this->enviada_por;
13 }
```

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.

```
23 {
24     this->sair = false;
25     this->sistema = &sistema;
26 }
```

4.3.3 Funções membros

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 164 do arquivo principal.cpp.

```
165 {
166     string saida, linha;
167     this->sair = false;
168     while (!this->sair) {
169         if (getline(inputStream, linha)) {
170             saida = processar_linha(linha);
171             outputStream « saida « endl;
172         }
173     }
174 }
```

4.3.3.2 processar_linha()

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

Definição na linha 29 do arquivo principal.cpp.

```
30 {
31     istringstream buf(linha);
32     string Cchooser;
33     buf » Cchooser;
34 
35     if (Cchooser.empty()) {
36         return "Digite um comando!";
37     }
38     if (Cchooser == "quit" ) {
39         this->sair = true;
40         return sistema->quit();
41     }
42     else if (Cchooser == "help") {
43         return sistema->help();
44     }
45     else if (Cchooser == "create-user") {
46         string email, senha, nome;
47         buf » email;
48         buf » senha;
49         nome = resto_de(buf);
```

```
50          if (email.empty() || senha.empty() || nome.empty())
51              return "»>Uso: create-user EMAIL SENHA NOME";
52          return sistema->create_user(email, senha, nome);
53      }
54      else if (Cchooser == "login") {
55          string email, senha;
56          buf » email;
57          buf » senha;
58          if (email.empty() || senha.empty())
59              return "»>Uso: login EMAIL SENHA";
60          return sistema->login(email, senha);
61      }
62      else if (Cchooser == "list-users") {
63          return sistema->list_users();
64      }
65
66      int id;
67      if (!(buf » id)) {
68          return "»>Comando precisa ser precedido de um ID [" + Cchooser + "]...";
69      }
70
71      if (Cchooser == "disconnect") {
72          return sistema->disconnect(id);
73      }
74      else if (Cchooser == "create-server") {
75          string nome;
76          buf » nome;
77          if (nome.empty())
78              return "»>Uso: create-server ID NOME";
79          return sistema->create_server(id, nome);
80      }
81      else if (Cchooser == "set-server-desc") {
82          string nome, descricao;
83          buf » nome;
84          descricao = resto_de(buf);
85          if (nome.empty())
86              return "»>Uso: set-server-desc ID NOME [DESCRICAO...]";
87          return sistema->set_server_desc(id, nome, descricao);
88      }
89      else if (Cchooser == "set-server-invite-code") {
90          string nome, codigo;
91          buf » nome;
92          buf » codigo;
93          if (nome.empty())
94              return "»>Uso: set-server-invite-code ID NOME [CODIGO]";
95          return sistema->set_server_invite_code(id, nome, codigo);
96      }
97      else if (Cchooser == "list-servers") {
98          return sistema->list_servers(id);
99      }
100      else if (Cchooser == "remove-server") {
101          string nome;
102          buf » nome;
103          if (nome.empty())
104              return "»>Uso: remove-server ID NOME";
105          return sistema->remove_server(id, nome);
106      }
107      else if (Cchooser == "enter-server") {
108          string nome, codigo;
109          buf » nome;
110          buf » codigo;
111          if (nome.empty())
112              return "»>Uso: enter-server ID NOME [CODIGO]";
113          return sistema->enter_server(id, nome, codigo);
114      }
115      else if (Cchooser == "leave-server") {
116          string nome;
117          buf » nome;
118          if (nome.empty())
119              return "»>Uso: leave-server ID NOME";
120          return sistema->leave_server(id, nome);
121      }
122      else if (Cchooser == "list-participants") {
123          return sistema->list_participants(id);
124      }
125      else if (Cchooser == "list-channels") {
126          return sistema->list_channels(id);
127      }
128      else if (Cchooser == "create-channel") {
129          string nome;
130          buf » nome;
131          if (nome.empty())
132              return "»>Uso: create-channel ID NOME";
133          return sistema->create_channel(id, nome);
134      }
135      else if (Cchooser == "enter-channel") {
136          string nome;
```

```
137         buf » nome;
138         if (nome.empty())
139             return "»>Uso: enter-channel ID NOME";
140         return sistema->enter_channel(id, nome);
141     }
142     else if (Cchooser == "leave-channel") {
143         return sistema->leave_channel(id);
144     }
145     else if (Cchooser == "send-message") {
146         string mensagem;
147         mensagem = resto_de(buf);
148         if (mensagem.empty())
149             return "»>Uso: send-message ID MESSAGE...";
150         return sistema->send_message(id, mensagem);
151     }
152     else if (Cchooser == "list-messages") {
153         return sistema->list_messages(id);
154     }
155     else {
156         return "»>Comando não reconhecido [" + Cchooser + "]...";
157     }
158 }
```

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.

```
12 {
13     this->nome = nome;
14     this->usuario_dono_id = dono_id;
15     this->descricao = "";
16     this->convite = "";
17     this->participantes_id.push_back(dono_id);
18 }
```

4.4.3 Funções membros

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.

```
90 {
91     this->canais_texto.push_back(canal);
92 }
```

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.

```
51 {
52     this->participantes_id.push_back(id);
53 }
```

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.

```
108 {
109     for (auto& textchannel : this->canais_texto) {
110         if (textchannel.get_nome() == canal)
111             return true;
112     }
113     return false;
114 }
```

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.

```
81 {
82     for (auto& p_id : this->participantes_id) {
83         if (p_id == id)
84             return true;
85     }
86     return false;
87 }
```

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.

```
129 {
130     for (auto& textchannel : this->canais_texto) {
131         if (textchannel.get_nome() == nome_canal)
132             textchannel.add_mensagem(mensagem);
133     }
134 }
```

4.4.3.6 get_convite()

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

Definição na linha 35 do arquivo servidor.cpp.

```
36 {
37     return this->convite;
38 }
```

4.4.3.7 get_descricao()

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

Definição na linha 30 do arquivo servidor.cpp.

```
31 {
32     return this->descricao;
33 }
```

4.4.3.8 get_nome()

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

Definição na linha 20 do arquivo servidor.cpp.

```
21 {
22     return this->nome;
23 }
```

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.

```
26 {
27     return this->usuario_dono_id;
28 }
```

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.

```
117 {
118     stringstream ss;
119     for (auto& textchannel : this->canais_texto) {
120         ss « "-> " « textchannel.get_nome() « endl;
121     }
122 
123     if (ss.str().empty())
124         return "Não existem canais de texto neste servidor, caso seja o dono, experimente adicionar
      um!";
125     return ss.str();
126 }
```

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.

```
137 {
138     for (auto& textchannel : this->canais_texto) {
139         if (canal_texto == textchannel.get_nome())
140             return textchannel.listar_mensagens(usuarios);
141     }
142     return "Este canal de texto não existe!";
143 }
```

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.

```
68 {
69     stringstream ss;
70     for (auto& user : usuarios) {
71         if (this->eh_participante(user.get_id()))
72             ss « "-" « user.get_id() « " " « user.get_nome() « endl;
73     }
74
75     if (ss.str().empty())
76         return "Ninguém está nesse servidor ainda, entre e convide seus amigos!";
77     return ss.str();
78 }
```

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.

```
95 {
96     auto contador = this->canais_texto.begin();
97     while (contador != this->canais_texto.end()) {
98         //Caso o nome seja encontrado na lista de canais, ele é deletado.
99         if (contador->get_nome() == canal) {
100             this->canais_texto.erase(contador);
101             return;
102         }
103         ++contador;
104     }
105 }
```

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.

```
56 {
57     auto i = this->participantes_id.begin();
58     while (i != this->participantes_id.end()) {
59         if (*i == id) {
60             this->participantes_id.erase(i);
61             return;
62         }
63         ++i;
64     }
65 }
```

4.4.3.15 set_convite()

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

Definição na linha 45 do arquivo servidor.cpp.

```
46 {
47     this->convite = convite;
48 }
```

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.

```
41 {
42     this->descricao = descricao;
43 }
```

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

- 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.

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 Funções membros

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 306 do arquivo sistema.cpp.

```
307 {
308     if (!this->usuarios_logados.count(id))
309         return "Você não está logado!";
310 
311     if (this->usuarios_logados[id].first.empty())
312         return "Você não está visualizando nenhum servidor!";
313 
314     for (auto& server : this->servers) {
315         if (server.get_nome() == this->usuarios_logados[id].first) {
316             if (server.eh_canal_texto(nome))
317                 return "Já existe um canal com este nome!";
318 
319             CanalTexto canal(nome);
320             server.add_canal_texto(canal);
321             return "Canal criado com sucesso!";
322         }
323     }
324     return "Você não é o dono do servidor...";
325 }
```

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 124 do arquivo sistema.cpp.

```
125 {
126     if (!this->usuarios_logados.count(id))
127         return "Você não está logado!";
128     for (auto& server : this->servers) {
129         if (nome == server.get_nome())
130             return "Já existe um servidor com este nome!";
131     }
132     Servidor servidor(id, nome);
133     this->servers.push_back(servidor);
134     return "Servidor criado com sucesso!";
135 }
```

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 81 do arquivo sistema.cpp.

```
82 {
83     if(!email_validate(email))
84         return "Email inválido!";
85     for (auto& user : this->usuarios) {
86         if (email == user.get_email())
87             return "Já existe uma conta registrada com esse e-mail! Tente logar!";
88         if (nome == user.get_nome())
89             return "Este nome de usuário já foi utilizado, tente outro!";
90     }
91
92     int id = this->usuarios.size() + 1;
93     Usuario usuario(id, nome, email, senha);
94     this->usuarios.push_back(usuario);
95     return cat("Novo usuário registrado! Seu ID: #", id);
96 }
```

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 115 do arquivo sistema.cpp.

```
116 {
117     if (!this->usuarios_logados.count(id))
118         return "Este usuário já está desconectado...";
119
120     this->usuarios_logados.erase(id);
121     return "Usuário offline!";
122 }
```

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 327 do arquivo sistema.cpp.

```
328 {
329     if (!this->usuarios_logados.count(id))
330         return "Você não está logado!";
331
332     if (this->usuarios_logados[id].first.empty())
333         return "Você não está visualizando nenhum servidor!";
334
335     if (!this->usuarios_logados[id].second.empty())
336         return "Você já está visualizando um canal de texto!";
337
338     for (auto& server : this->servers) {
339         if (server.get_nome() == this->usuarios_logados[id].first) {
340             if (server.eh_canal_texto(nome)){
341                 this->usuarios_logados[id].second = nome;
342                 return cat("O usuário ", nome, " entrou no canal!");
343             }
344         }
345     }
346     return "Canal não encontrado!";
347 }
```

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 210 do arquivo sistema.cpp.

```
211 {
212     if (!this->usuarios_logados.count(id))
213         return "Você não está logado!";
214 
215     for (auto& server : this->servers) {
216         if (nome == server.get_nome()) {
217             if (this->usuarios_logados[id].first == nome)
218                 return "Este usuário já está neste servidor!";
219             if (id != server.get_usuario_dono_id() && codigo != server.get_convite())
220                 return "Código de convite inválido, você o digitou corretamente?";
221             this->usuarios_logados[id].first = nome;
222             this->usuarios_logados[id].second = "";
223             if (server.eh_participante(id))
224                 return "Você entrou no servidor com sucesso!";
225             server.add_participante(id);
226             return "Você entrou no servidor com sucesso! Bem-vindo!";
227         }
228     }
229     return "Este servidor não existe!";
230 }
```

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 55 do arquivo sistema.cpp.

```
56 {
57     string helpmi = "\n_____________________________Lista de comandos_____________________________\n\n --- "
58                     "quit - Encerra o programa.\n --- "
59                     "create-user <E-mail> <Senha_sem_espaços> <Nome do usuário> - Registra um novo
       usuário. \n --- "
60                     "login <E-mail> <senha> - Loga o usuário no sistema.\n --- "
61                     "disconnect - Desconecta o usuário atual.\n --- "
62                     "create-server <id> <nome-do-servidor> - Cria um novo servidor.\n --- "
63                     "set-server-desc <id> <nome-do-servidor> <descrição> - Altera a descrição de um
       servidor que você é o dono.\n --- "
64                     "set-server-invite-code <id> <nome-do-servidor> <código-desejado> - Adiciona um
       código de convite a um servidor, tornando-o privado.\n --- "
65                     "list-servers <id> - Lista todos os servidores criados.\n --- "
66                     "remove-server <id> <nome-do-servidor> - Elimina um servidor.\n --- "
67                     "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 --- "
68                     "leave-server <id> <nome-do-servidor> - Quando estiver dentro do servidor,
       desconecte-se dele.\n --- "
69                     "list-participants <id> - Lista todos os participantes de um servidor.\n --- "
70                     "list-users <id> - Lista os usuários do sistema e exibe seus status.\n --- "
71                     "create-channel <id> <nome> - Cria um novo canal dentro de um servidor.\n --- "
72                     "list-channels <id> - Lista os canais do servidor.\n --- "
73                     "enter-channel <id> - Entra em um canal existente.\n --- "
74                     "leave-channel <id> - Sai do canal.\n --- "
75                     "send-message <id> <mensagem> - Envia uma mensagem no canal atual.\n --- "
76                     "list-messages <id> - Lista todas as mensagens do canal.\n\n"
77                     "_____________________________________________________\n";
78     return helpmi;
79 }
```

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 349 do arquivo sistema.cpp.

```
350 {
351     if (!this->usuarios_logados.count(id))
352         return "Você não está logado!";
353 
354     if (this->usuarios_logados[id].first.empty())
355         return "Você não está visualizando nenhum servidor!";
356 
357     if (this->usuarios_logados[id].second.empty())
358         return "Você não está visualizando nenhum canal de texto!";
359 
360     string c;
361     c = this->usuarios_logados[id].second;
362     this->usuarios_logados[id].second = "";
363     return cat("Você saiu do canal ", c, "!");
364 
365 }
```

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 232 do arquivo sistema.cpp.

```
233 {
234     if (!this->usuarios_logados.count(id))
235         return "Você não está logado!";
236 
237     for (auto& server : this->servers) {
238         if (nome == server.get_nome()) {
239             if (id == server.get_usuario_dono_id())
240                 return "»Você não pode remover o dono do servidor!";
241 
242             if (!server.eh_participante(id))
243                 return "Este usuário não participa deste servidor...";
244 
245             if (this->usuarios_logados[id].first == nome) {
246                 this->usuarios_logados[id].first = "";
247                 this->usuarios_logados[id].second = "";
```

```
248             }
249             server.remove_participante(id);
250             return "O usuário foi removido do servidor!";
251         }
252     }
253     return "Este servidor não existe!";
254 }
```

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 291 do arquivo sistema.cpp.

```
292 {
293     if (!this->usuarios_logados.count(id))
294         return "Você não está logado!";
295
296     if (this->usuarios_logados[id].first.empty())
297         return "Você não está visualizando nenhum servidor!";
298
299     for (auto& server : this->servers) {
300         if (server.get_nome() == this->usuarios_logados[id].first)
301             return server.imprimir_canais_texto();
302     }
303     return "Este servidor não existe!";
304 }
```

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 388 do arquivo sistema.cpp.

```
389 {
390     if (!this->usuarios_logados.count(id))
391         return "Você não está logado!";
392 
393     if (this->usuarios_logados[id].first.empty())
394         return "Você não está visualizando nenhum servidor!";
395 
396     if (this->usuarios_logados[id].second.empty())
397         return "Você não está visualizando nenhum canal de texto!";
398 
399     for (auto& server : this->servers) {
400         if (server.get_nome() == this->usuarios_logados[id].first) {
401             return server.imprimir_mensagens(this->usuarios_logados[id].second, this->usuarios);
402         }
403     }
404     return "Este servidor não existe!";
405 }
```

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 256 do arquivo sistema.cpp.

```
257 {
258     if (!this->usuarios_logados.count(id))
259         return "Você não está logado!";
260 
261     if (this->usuarios_logados[id].first.empty())
262         return "O usuário não está visualizando nenhum servidor...";
263 
264     for (auto& server : this->servers) {
265         if (server.get_nome() == this->usuarios_logados[id].first)
266             return server.imprimir_participantes(this->usuarios);
267     }
268     return "Este servidor não existe!";
269 }
```

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 197 do arquivo sistema.cpp.

```
198 {
199     if (this->servers.empty())
200         return "Nenhum servidor foi criado ainda...";
201 
202     stringstream ss;
203     for (auto& server : this->servers) {
204         ss « "-> " « server.get_nome() « " [\"" « server.get_descricao() « "\"] ";
205         ss « "-> Dono do servidor: " « server.get_usuario_dono_id() « endl;
206     }
207     return ss.str();
208 }
```

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 271 do arquivo sistema.cpp.

```
272 {
273     if (this->usuarios.empty())
274         return "Nenhum usuário encontrado!";
275 
276     stringstream ss;
277     for (auto& user : this->usuarios) {
278         ss « "#" « user.get_id() « " " « user.get_nome() « " [";
279 
280         if (this->usuarios_logados.count(user.get_id())) {
281             ss « this->usuarios_logados[user.get_id()].first « " / ";
282             ss « this->usuarios_logados[user.get_id()].second « "] ";
283             ss « "Atualmente online" « endl;
284         } else {
285             ss « " / ] " « "Atualmente offline" « endl;
286         }
287     }
288     return ss.str();
289 }
```

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 98 do arquivo sistema.cpp.

```
99 {
100     for (auto& user : this->usuarios) {
101         if (email == user.get_email() && !user.same_pw(senha))
102             return "Senha incorreta, verifique sua senha e tente novamente!";
103 
104         if (email == user.get_email() && user.same_pw(senha)) {
105             if (this->usuarios_logados.count(user.get_id()))
106                 return "Este usuário já está logado!";
107 
108             this->usuarios_logados[user.get_id()] = make_pair("","");
109             return "Parabéns, você agora está Online!";
110         }
111     }
112     return "Conta não encontrada! Verifique seu e-mail ou cadastre-se!";
113 }
```

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 50 do arquivo sistema.cpp.

```
51 {
52     return "Encerrando o programa...";
53 }
```

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 137 do arquivo sistema.cpp.

```
138 {
139     if (!this->usuarios_logados.count(id))
140         return "Você não está logado!";
141 
142     auto i = this->servers.begin();
143     while (i != this->servers.end()) {
144         if (i->get_nome() == nome) {
145             if (i->get_usuario_dono_id() != id)
146                 return "Apenas o dono do servidor pode remover o servidor!";
147 
148             for (auto& user: this->usuarios) {
149                 if (this->usuarios_logados[user.get_id()].first == nome) {
150                     this->usuarios_logados[user.get_id()].first = "";
151                     this->usuarios_logados[user.get_id()].second = "";
152                 }
153             }
154             this->servers.erase(i);
155             return "Servidor removido com sucesso!";
156         }
157         ++i;
158     }
159     return "Este servidor não existe!";
160 }
```

4.5.2.18 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 367 do arquivo sistema.cpp.

```
368 {
369     if (!this->usuarios_logados.count(id))
370         return "Você não está logado!";
371 
372     if (this->usuarios_logados[id].second.empty())
373         return "Você não está visualizando nenhum canal de texto!";
374 
375     for (auto& server : this->servers) {
376         if (server.get_nome() == this->usuarios_logados[id].first) {
377             if (!server.eh_canal_texto(this->usuarios_logados[id].second))
378                 return "Este canal não existe!";
379 
380             Mensagem mensagem(id, conteudo);
381             server.enviar_mensagem(mensagem, this->usuarios_logados[id].second);
382             return "Mensagem enviada!";
383         }
384     }
385     return "Este servidor não existe!";
386 }
```

4.5.2.19 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 180 do arquivo sistema.cpp.

```
181 {
182     if (!this->usuarios_logados.count(id))
183         return "Você não está logado!";
184     for (auto& server : this->servers) {
185         if (nome == server.get_nome()) {
186             if (id != server.get_usuario_dono_id())
187                 return "Você não é o dono do servidor...";
188             server.set_descricao(descricao);
189             if (descricao == "")
190                 return cat(" Descrição do servidor \"", server.get_nome(), "\" removida!");
191             return cat("Descrição do servidor \"", server.get_nome(),"\" adicionada!");
192         }
193     }
194     return "Este servidor não existe!";
195 }
```

4.5.2.20 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 162 do arquivo sistema.cpp.

```
163 {
164     if (!this->usuarios_logados.count(id))
165         return "Você não está logado!";
166 
167     for (auto& server : this->servers) {
168         if (nome == server.get_nome()) {
169             if (id != server.get_usuario_dono_id())
170                 return "Você não é o dono do servidor...";
171             server.set_convite(codigo);
172             if (codigo == "")
173                 return cat("Código de convite do servidor \"", server.get_nome(), "\" removido! O
    servidor agora é público!");
174             return cat("Codigo de convite do servidor \"", server.get_nome(),"\" criado! Convide seus
    amigos! :D");
175         }
176     }
177     return "Este servidor não existe!";
178 }
```

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 ()
- 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.

```
7 {
8     this->id = id;
9     this->nome = nome;
10     this->email = email;
11     this->senha = senha;
12 }
```

4.6.3 Funções membros

4.6.3.1 get_email()

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

Definição na linha 24 do arquivo usuario.cpp.

```
25 {
26     return this->email;
27 }
```

4.6.3.2 get_id()

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

Retorna

Valor do atributo correspondente.

Definição na linha 14 do arquivo usuario.cpp.

```
15 {
16     return this->id;
17 }
```

4.6.3.3 get_nome()

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

Definição na linha 19 do arquivo usuario.cpp.

```
20 {
21     return this->nome;
22 }
```

4.6.3.4 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 29 do arquivo usuario.cpp.

```
30 {
31     return this->senha == linha;
32 }
```

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 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

5.3 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

5.4 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

5.5 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

5.6 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

5.7 Referência do Arquivo README.md

5.8 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:

5.9 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 ()

5.9.1 Funções

5.9.1.1 main()

```
int main ( )
```

Definição na linha 10 do arquivo concord_main.cpp.

```
11 {
12     cout « "Bem vindo! Digite um comando para começar (ou 'help' para ajuda): ";
13
14     Sistema sistema; //Inicializando o Sistema
15
16     principal principal(sistema); //Inicializando o processo principal do sistema.
17
18     principal.iniciar(cin, cout); //Esse comando lê o cin e o cout, executando o método correto e
       imprimindo as mensagens em seguida.
19
20     return 0;
21 }
```

5.10 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:

5.11 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)

5.11.1 Funções

5.11.1.1 resto_de()

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

Definição na linha 11 do arquivo principal.cpp.

```
12 {
13     string resto;
14     getline(ss, resto, '\0');
15     if (resto != "" && (resto[0] == ' ' || resto[0] == '\t')) {
16         resto = resto.substr(1); // Eliminar o primeiro caractere caso ele seja um espaço vazio.
17     }
18     return resto;
19 }
```

5.12 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:

5.13 Referência do Arquivo src/sistema.cpp

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <regex>
#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.

5.13.1 Funções

5.13.1.1 cat() [1/2]

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

Concatena diversas variaveis "Streamable" recursivamente.

Definição na linha 27 do arquivo sistema.cpp.

```
28 {
29     stringstream ss;
30     ss « h;
31     return ss.str() + cat(t...);
32 }
```

5.13.1.2 cat() [2/2]

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

Concatena duas strings "Streamable"

Definição na linha 16 do arquivo sistema.cpp.

```
17 {
18     stringstream ss;
19     ss « s;
20     return ss.str();
21 }
```

5.13.1.3 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 41 do arquivo sistema.cpp.

```
42 {
43     const regex email_regex ("\\w+@\\w+\\.[\\w\\.]+"); //cria um formato para o e-mail
44     return regex_match(email, email_regex);
45 }
```

5.14 Referência do Arquivo src/usuario.cpp

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

Gráfico de dependência de inclusões para usuario.cpp: