My Project

Создано системой Doxygen 1.9.4

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

Глава 3

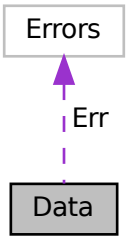
Классы

3.1 Класс Data

Класс Data для управления данными, связанными с файлами и клиентами.

#include <Data.h>

Граф связей класса Data:



Открытые члены

- string get_FileReader ()
  Возвращает имя файла, которое используется для чтения данных.
- void set_FileReader (string file)
- pair< vector< string >, vector< string > > getClient ()

Закрытые данные

- string FileReader
  Строка для хранения имени файла, которое используется для чтения данных.
- Errors Err
  Объект класса Errors для обработки ошибок.

3.1.1 Подробное описание

Класс Data для управления данными, связанными с файлами и клиентами.

Этот класс предназначен для хранения и обработки информации, связанной с чтением данных из файла, а также для получения информации о клиентах, хранимой в двух векторах.

3.1.2 Методы

3.1.2.1 get_FileReader()

string Data::get_FileReader ( )

Возвращает имя файла, которое используется для чтения данных.

Возвращает

std::string Имя файла, установленное с помощью set_FileReader.

Объявления и описания членов классов находятся в файлах:

- Data.h
- Data.cpp

3.2 Класс Errors

Открытые члены

- string get_File_Log ()
- void set_File_Log (string file)
- void error_recording (string flag, string info)

Закрытые данные

- string File_Log

Объявления и описания членов классов находятся в файлах:

- Errors.h
- Errors.cpp

3.3 Класс interface

Открытые члены

- interface (int argc, char **argv)
- string get_ip ()
- int get_port ()
- int help ()

Открытые атрибуты

- string ip
- int port
- string database
- string logfile

Объявления и описания членов классов находятся в файлах:

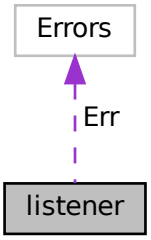
- Interface.h
- Interface.cpp

3.4 Класс listener

Класс listener, реализующий серверную часть приложения.

#include <listener.h>

Граф связей класса listener:



Открытые члены

- int interaction (string database, string logFile)

  Устанавливает взаимодействие с клиентами, обрабатывает их запросы и отправляет ответы.
- uint64_t sredn ()

  Вычисляет среднее арифметическое значений в векторе vec.
- string get_address ()
- void set_address (string address1)
- int get_port ()

  Устанавливает номер порта сервера.
- void set_port (int port1)
- vector< uint64_t > get_vec ()
- void set_vec (vector< uint64_t > v)
- string get_salt ()
- void set_salt ()
- pair< vector< string >, vector< string > > get_DB_clients ()
- void set_DB_clients (vector< string > login, vector< string > password)

Закрытые данные

- string address
- int port
- vector< uint64_t > vec
- string salt
- pair< vector< string >, vector< string > > DB_clients
- Errors Err

3.4.1 Подробное описание

Класс listener, реализующий серверную часть приложения.

Этот класс отвечает за установление соединения с клиентами, обработку их запросов и отправку ответов.

3.4.2 Методы

3.4.2.1 get_port()

int listener::get_port ( )

Устанавливает номер порта сервера.

Аргументы

port1	Номер порта, который нужно установить.

3.4.2.2 interaction()

```
int listener::interaction (
                string database,
                string logFile )
```

Устанавливает взаимодействие с клиентами, обрабатывает их запросы и отправляет ответы.

Аргументы

database	Путь к файлу базы данных.
logFile	Путь к файлу логов.

Возвращает

0, если взаимодействие успешно установлено; иначе -1.

DataReader

!!!!!!!!!

3.4.2.3 sredn()

```
uint64_t listener::sredn ( )
```

Вычисляет среднее арифметическое значений в векторе vec.

Возвращает

Среднее арифметическое значений в векторе.

3.4.3 Данные класса

3.4.3.1 address

```
string listener::address   [private]
```

IP-адрес сервера.

3.4.3.2 DB_clients

```
pair<vector<string>, vector<string> > listener::DB_clients   [private]
```

Пара векторов: логины и пароли пользователей.

3.4.3.3 Err

Errors listener::Err [private]

Объект для обработки ошибок.

3.4.3.4 port

int listener::port [private]

Номер порта сервера.

3.4.3.5 salt

string listener::salt [private]

Сгенерированная соль.

3.4.3.6 vec

vector<uint64_t> listener::vec [private]

Вектор значений.

Объявления и описания членов классов находятся в файлах:

- listener.h
- listener.cpp

3.5 Класс User

Открытые члены

- bool CheckLogin (vector< string > Db_ID)
- bool CheckPassword (vector< string > Db_hash, vector< string > Db_ID, string SALT)
- string get_ID ()
- void set_ID (string ID1)
- string get_hash ()
- void set_hash (string hash1)

Закрытые данные

- string ID
- string hash

Объявления и описания членов классов находятся в файлах:

- User.h
- User.cpp

Глава 4

Файлы

4.1 Data.h

```
1
2 #include <string>
3 #include <vector>
4 #include <map>
5 #include <fstream>
6 #include <sstream>
7 #include <iostream>
8
9 #include "Errors.h"
10
11 using namespace std;
12
19 class Data{
20     public:
26         string get_FileReader();
27         void set_FileReader(string file);
28
29         pair<vector<string>, vector<string» getClient();
30     private:
34         string FileReader;
38         Errors Err;
39 };
```

4.2 Errors.h

```
1 #include <string>
2 #include <chrono>
3 #include <ctime>
4 #include <iostream>
5 #include <fstream>
6 #include <iomanip>
7
8 using namespace std;
9
10 /*
11 * @brief Класс для обработки и записи ошибок.
12 *
13 * Класс Errors предоставляет функциональность для записи информации об ошибках в лог-файл, а также
14 * для управления именем файла лога.
15 */
16 class Errors{
17     public:
18         string get_File_Log();
19         void set_File_Log(string file);
20
21         void error_recording(string flag, string info);
22
23     private:
24         string File_Log; /*< Имя файла лога ошибок. */};
25
```

4.3 Interface.h

```
1 #pragma once
2 #include <algorithm>
3 #include <iostream>
4 #include <string>
5 #include <string_view>
6 #include <array>
7 #include <unistd.h>
8
9 using namespace std;
10 /*
11 * @brief Класс interface для обработки аргументов командной строки и хранения конфигурационных данных.
12 *
13 *  Этот класс предназначен для обработки аргументов командной строки,
14 *  полученных при запуске программы, и хранения конфигурационных данных, таких как IP-адрес,
15 *  номер порта, имя файла базы данных и имя файла журнала ошибок.
16 */
17 class interface {
18 public:
19     string ip;   /*< IP-адрес. */
20     int port;    /*< Порт. */
21     string database;  /*< Название файла с базой данных. */
22     string logfile;     /*< Название файла, куда записываются ошибки. */
23     /*
24 * @brief Конструктор, обрабатывающий аргументы командной строки.
25 *
26 * @param argc Количество аргументов командной строки.
27 * @param argv Массив аргументов командной строки.
28 */
29     interface(int argc, char** argv);
30     /*
31 * @brief Запрещает создание объектов класса без аргументов.
32 */
33     interface() = delete;
34      /*
35 * @brief Возвращает IP-адрес.
36 * @return IP-адрес.
37 */
38     string get_ip(){
39         return ip;}
40
41     /*
42 * @brief Возвращает номер порта.
43 * @return Номер порта.
44 */
45     int get_port(){
46         return port;}
47
48     int help();
49
50 };
51
```

4.4 listener.h

```

2 #pragma one
3 #include <sys/types.h>
4 #include <iostream>
5 #include <cstdlib>
6 #include <cstring>
7 #include <unistd.h>
8 #include <sys/socket.h>
9 #include <netinet/in.h>
10 #include <arpa/inet.h>
11 #include <fstream>
12 #include <vector>
13 #include <random>
14
15 #include <string>
16 #include <locale>
17 #include <codecvt>
18
19 #include "Data.h"
20 #include <vector>
21 #include <numeric>
22 #include <limits>
23 #include <iostream>
24 #include "User.h"
25
26 using namespace std;
27
```

```
33 class listener{
34     public:
42         int interaction(string database, string logFile);
43 
48         uint64_t sredn();
49 
50         string get_address();
51         void set_address(string address1);
56         int get_port();
57         void set_port(int port1);
58 
59         vector<uint64_t> get_vec();
60         void set_vec(vector<uint64_t> v);
61 
62         string get_salt();
63         void set_salt();
64 
65         pair<vector<string>, vector<string» get_DB_clients();
66         void set_DB_clients(vector<string> login, vector<string> password);
67 
68     private:
69         string address;
70         int port;
71         vector<uint64_t> vec;
72         string salt;
73         pair<vector<string>, vector<string»DB_clients;
74         Errors Err;
75 };
```

4.5 sha224.h

```
1 #include <iostream>
2 #include <vector>
3 #include <cryptopp/hex.h>
4 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
5 #include <string>
6 #include <cryptopp/sha.h>
7 #include <cryptopp/hex.h>
8 #include <cryptopp/filters.h>
9 
10 std::string SHA224_hash(std::string msg);
```

4.6 User.h

```
1 #include <string>
2 #include  <vector>
3 #include <iostream>
4 
5 #include "sha224.h"
6 
7 using namespace std;
8 
9 /*
10 * @brief Класс, представляющий пользователя системы.
11 *
12 * Класс User хранит информацию о пользователе (ID и хэш пароля) и предоставляет методы для проверки
13 * подлинности пользователя. Обратите внимание, что хранение хэшей паролей напрямую в коде -
14 * небезопасная практика. В реальных приложениях необходимо использовать более безопасные методы,
15 * такие как bcrypt или Argon2, и хранить хэши в защищенной базе данных.
16 */
17 class User{
18     private:
19         string ID;  /*< Идентификатор пользователя. */
20         string hash;  /*< Хэш пароля пользователя. */
21     public:
22         bool CheckLogin(vector<string> Db_ID);
23         bool CheckPassword(vector<string> Db_hash, vector<string> Db_ID, string SALT);
24 
25         string get_ID();
26         void set_ID(string ID1);
27 
28         string get_hash();
29         void set_hash(string hash1);
30 
31 };
```

Предметный указатель