My Project

Создано системой Doxygen 1.9.4

1 Алфавитный указатель классов	1
1.1 Классы	1
2 Список файлов	3
$2.1 \; \Phi$ айлы	3
3 Классы	5
3.1 Класс Data	5
3.1.1 Подробное описание	6
3.1.2 Методы	6
$3.1.2.1 \text{ get}$ _FileReader()	6
3.2 Класс Errors	6
3.2.1 Подробное описание	7
3.2.2 Методы	7
$3.2.2.1 \text{ error_recording}() \dots \dots \dots \dots \dots \dots \dots \dots \dots$	7
$3.2.2.2 \text{ get_File_Log()} \dots \dots \dots \dots \dots \dots \dots \dots \dots$	7
$3.2.2.3 \text{ set_File_Log()} \dots \dots \dots \dots \dots \dots \dots$	7
3.3 Класс interface	8
3.3.1 Подробное описание	8
3.3.2 Конструктор(ы)	8
3.3.2.1 interface()	8
3.3.3 Методы	9
$3.3.3.1 \text{ get_ip}() \dots \dots$	9
3.3.3.2 get_port()	9
$3.3.3.3 \text{ help}() \dots \dots$	9
3.4 Класс listener	10
3.4.1 Подробное описание	11
3.4.2 Методы	11
3.4.2.1 get_port()	11
3.4.2.2 interaction()	
3.4.2.3 sredn()	11
3.4.3 Данные класса	12
3.4.3.1 address	12
3.4.3.2 DB_clients	12
3.4.3.3 Err	
3.4.3.4 port	12
3.4.3.5 salt	12
3.4.3.6 vec	12
3.5 Класс User	13
3.5.1 Подробное описание	13
3.5.2 Методы	
3.5.2.1 CheckLogin()	
3.5.2.2 CheckPassword()	
3.5.2.3 get_hash()	
— "	

Предметный указатель

3.5.2.4 get_ID()	15
3.5.2.5 set_hash()	15
$3.5.2.6 \text{ set_ID}()$	15
4 Файлы	17
4.1 Файл Data.cpp	17
4.1.1 Подробное описание	17
4.2 Файл Data.h	17
4.2.1 Подробное описание	18
4.3 Data.h	19
4.4 Файл Errors.cpp	19
4.4.1 Подробное описание	19
4.5 Файл Errors.h	19
4.5.1 Подробное описание	20
4.6 Errors.h	21
4.7 Файл Interface.cpp	21
4.7.1 Подробное описание	21
4.8 Файл Interface.h	21
4.8.1 Подробное описание	22
4.9 Interface.h	22
4.10 Файл listener.cpp	23
4.10.1 Подробное описание	23
4.11 Файл listener.h	23
4.11.1 Подробное описание	24
4.12 listener.h	24
4.13 Файл sha224.cpp	25
4.13.1 Подробное описание	26
4.14 Файл sha224.h	26
4.14.1 Подробное описание	27
4.15 sha224.h	27
4.16 Файл User.cpp	27
4.16.1 Подробное описание	28
4.16.2 Функции	28
4.16.2.1 foo()	28
4.17 Файл User.h	28
4.17.1 Подробное описание	29
4.18 User.h	29

31

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Data		
	Класс Data для управления данными, связанными с файлами и клиентами	5
Errors		
	Класс для обработки и записи ошибок	6
interfac	e	
	Класс interface для обработки аргументов командной строки и хранения конфигу-	
	рационных данных	8
listener		
	Класс listener, реализующий серверную часть приложения. Этот класс отвечает за	
	установление соединения с клиентами, обработку их запросов и отправку ответов	10
User		
	Класс, представляющий пользователя системы	13

Алфавитный	указатель	классов
TITOMORITIDIA	Y IXAOA I CAID	12/10/00/1

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

Data.cpp	
Класс Data для считывания данных клиентов из файла	17
Data.h	
Класс Data для управления данными, связанными с файлами и клиентами	17
Errors.cpp	
Класс Errors для обработки ошибок и их записи в файл	19
Errors.h	
Класс Errors для обработки и записи ошибок	19
Interface.cpp	
Класс Interface для обрабатки аргументов командной строки	21
Interface.h	
Класс Interface для обработки аргументов командной строки и хранения конфигурационных данных	21
listener.cpp	
Класс listener для вычисления среднего арифметического значений в векторе	23
listener.h	
Класс listener для вычисления среднего арифметического значений в векторе	23
${ m sha}224.{ m cpp}$	
Вычисляет SHA224 хэш для заданной строки	25
${ m sha}224.{ m h}$	
Вычисляет SHA224 хэш для заданной строки	26
User.cpp	
Класс User, представляющий пользователя системы	27
User.h	
Класс User, представляющий пользователя системы	28

4 Список файлов

Глава 3

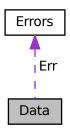
Классы

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.

6 Классы

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 \quad \text{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

Класс для обработки и записи ошибок.

```
\#include <Errors.h>
```

Открытые члены

```
• string get File Log ()
```

Возвращает имя файла лога ошибок.

• void set File Log (string file)

Устанавливает имя файла лога ошибок.

• void error_recording (string flag, string info)

Записывает информацию об ошибке в лог-файл.

Закрытые данные

string File_Log
 Имя файла лога ошибок.

3.2 Класс Errors 7

3.2.1 Подробное описание

Класс для обработки и записи ошибок.

Класс Errors предоставляет функциональность для записи информации об ошибках в лог-файл, а также для управления именем файла лога.

3.2.2 Методы

Записывает информацию об ошибке в лог-файл.

Аргументы

flag	Тип ошибки
info	Описание ошибки

```
3.2.2.2 \quad \text{get}\_File\_Log() string Errors::get_File_Log()
```

Возвращает имя файла лога ошибок.

Возвращает

Строка, содержащая имя файла лога ошибок.

```
3.2.2.3 set_File_Log()

void Errors::set_File_Log (
    string file )
```

Устанавливает имя файла лога ошибок.

Аргументы

file	Имя	файла	лога	ошибок.
------	-----	-------	------	---------

8 Классы

Объявления и описания членов классов находятся в файлах:

- Errors.h
- Errors.cpp

3.3 Класс interface

Класс interface для обработки аргументов командной строки и хранения конфигурационных данных.

```
#include <Interface.h>
```

Открытые члены

```
• interface (int argc, char **argv)
```

Конструктор, обрабатывающий аргументы командной строки.

• interface ()=delete

Запрещает создание объектов класса без аргументов.

• string get ip ()

Возвращает ІР-адрес.

• int get_port ()

Возвращает номер порта.

• int help ()

Возвращает справку и завершает работу.

Открытые атрибуты

• string ip

IP-адрес

int port

Порт

• string database

Название файла с базой данных

· string logfile

Название файла, куда записываются ошибки

3.3.1 Подробное описание

Класс interface для обработки аргументов командной строки и хранения конфигурационных данных.

Этот класс предназначен для обработки аргументов командной строки, полученных при запуске программы, и хранения конфигурационных данных, таких как IP-адрес, номер порта, имя файла базы данных и имя файла журнала ошибок.

3.3.2 Конструктор(ы)

Конструктор, обрабатывающий аргументы командной строки.

3.3 Класс interface 9

Аргументы

argc	Количество аргументов командной строки.
argv	Массив аргументов командной строки.

3.3.3 Методы

```
3.3.3.1 \text{ get\_ip}()
```

string interface::get_ip () [inline]

Возвращает ІР-адрес.

Возвращает

ІР-адрес.

int interface::get_port () [inline]

Возвращает номер порта.

Возвращает

int Номер порта.

3.3.3.3 help()

int interface::help ()

Возвращает справку и завершает работу.

Возвращает

int Возвращает справку.

Объявления и описания членов классов находятся в файлах:

- Interface.h
- Interface.cpp

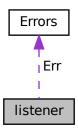
10 Классы

3.4 Kласc listener

Класс listener, реализующий серверную часть приложения. Этот класс отвечает за установление соединения с клиентами, обработку их запросов и отправку ответов.

#include tener.h>

Граф связей класса listener:



Открытые члены

• int interaction (string database, string logFile)

 ${
m Устанавливает}$ 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 < uint 64 + c > 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 > $\frac{\text{vec}}{\text{vec}}$
- string salt
- pair< vector< string >, vector< string > DB_clients
- Errors Err

3.4 Класс listener 11

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.

!!!!!!!!!

3.4.2.3 sredn()

uint64_t listener::sredn ()

Вычисляет среднее арифметическое значений в векторе vec.

Возвращает

Среднее арифметическое значений в векторе.

12 Классы

3.4.3 Данные класса

```
3.4.3.1 address
string listener::address [private]
ІР-адрес сервера.
3.4.3.2 DB_clients
pair < vector < string >, vector < string > > listener::DB\_clients \quad [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
{\tt vector}{<} {\tt uint64\_t}{>} \ {\tt listener}{::} {\tt vec} \quad [{\tt private}]
```

Объявления и описания членов классов находятся в файлах:

 \bullet listener.h

Вектор значений.

 \bullet listener.cpp

3.5 Класс User 13

3.5 Класс User

Класс, представляющий пользователя системы.

```
#include <User.h>
```

Открытые члены

• bool CheckLogin (vector< string > Db ID)

Проверяет, существует ли ID пользователя в базе данных.

• bool CheckPassword (vector< string > Db_hash, vector< string > Db_ID, string SALT)

Проверяет пароль пользователя.

• string get ID ()

Возвращает ID пользователя.

• void set ID (string ID1)

Устанавливает ID пользователя.

• string get_hash ()

Возвращает хэш пароля пользователя.

• void set hash (string hash1)

Устанавливает хэш пароля пользователя.

Закрытые данные

• string ID

Идентификатор пользователя.

string hash

Хэш пароля пользователя.

3.5.1 Подробное описание

Класс, представляющий пользователя системы.

Класс User хранит информацию о пользователе (ID и хэш пароля) и предоставляет методы для проверки подлинности пользователя. Обратите внимание, что хранение хэшей паролей напрямую в коде - небезопасная практика. В реальных приложениях необходимо использовать более безопасные методы, такие как bcrypt или Argon2, и хранить хэши в защищенной базе данных.

3.5.2 Методы

3.5.2.1 CheckLogin()

```
bool User::CheckLogin ( \label{eq:condition} vector < string > D\,b \quad ID \ )
```

Проверяет, существует ли ID пользователя в базе данных.

Функция проверяет наличие переданного ID пользователя в векторе ID из базы данных.

14 Классы

Аргументы

, содержащий ID пользователей из базы данных.

Возвращает

true, если ID пользователя найден в базе данных, false в противном случае.

3.5.2.2 CheckPassword()

```
\label{eq:bool_user::CheckPassword} $$ \operatorname{vector} < \operatorname{string} > \operatorname{Db\_password}, $$ \operatorname{vector} < \operatorname{string} > \operatorname{Db\_ID}, $$ \operatorname{string} SALT ) $$
```

Проверяет пароль пользователя.

Функция проверяет корректность пароля пользователя, сравнивая его хэш с хэшем из базы данных. Сначала по ID пользователя находится соответствующий пароль в базе данных. Затем вычисляется хэш переданного пароля с добавлением соли и сравнивается с хэшем из базы данных.

Аргументы

Db_password	Вектор строк, содержащий пароли пользователей из базы данных.
Db_ID	Вектор строк, содержащий ID пользователей из базы данных.
SALT	Строка, содержащая соль для хеширования.

Возвращает

true, если пароль верный, false в противном случае.

```
3.5.2.3 \text{ get\_hash()}
```

```
string User::get_hash ( )
```

Возвращает хэш пароля пользователя.

Возвращает

Строка, содержащая хэш пароля пользователя.

3.5 Класс User 15

```
3.5.2.4 \text{ get\_ID}()
```

```
string User::get_ID ( )
```

Возвращает ID пользователя.

Возвращает

Строка, содержащая ID пользователя.

```
3.5.2.5 set_hash()

void User::set_hash(

string hash1)
```

Устанавливает хэш пароля пользователя.

Аргументы

hash1 Новый хэш пароля пользователя.

```
\begin{array}{ll} 3.5.2.6 & {\rm set\_ID()} \\ \\ {\rm void~User::set\_ID~(} \\ & {\rm string~ID1~)} \end{array}
```

Устанавливает ID пользователя.

Аргументы

ID1 Новый ID пользователя.

Объявления и описания членов классов находятся в файлах:

- User.h
- User.cpp

Классы 16

Глава 4

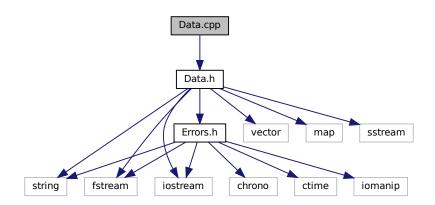
Файлы

4.1 Файл Data.cpp

Класс Data для считывания данных клиентов из файла.

#include "Data.h"

Граф включаемых заголовочных файлов для Data.cpp:



4.1.1 Подробное описание

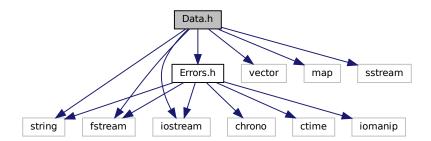
Класс Data для считывания данных клиентов из файла.

4.2 Файл Data.h

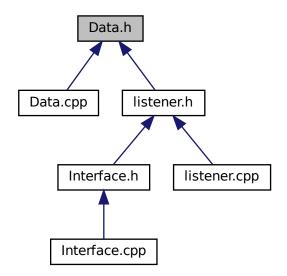
Класс Data для управления данными, связанными с файлами и клиентами.

#include <string> #include <vector> 18 Файлы

```
#include <map>
#include <fstream>
#include <sstream>
#include <iostream>
#include "Errors.h"
Граф включаемых заголовочных файлов для Data.h:
```



Граф файлов, в которые включается этот файл:



Классы

• class Data

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

4.2.1 Подробное описание

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

4.3 Data.h

4.3 Data.h

См. документацию.

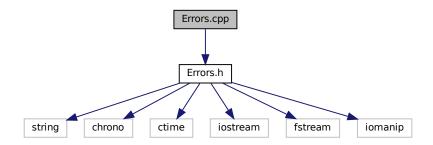
```
5 #include <string>
6 #include <vector>
7 #include <map>
8 #include <fstream>
9 #include <sstream>
10 #include <iostream>
12 #include "Errors.h"
13
14 using namespace std;
15
22 class Data{
^{23}
            string get_FileReader();
void set_FileReader(string file);
^{29}
30
31
32
            {\tt pair}{<}{\tt vector}{<}{\tt string}{>},\, {\tt vector}{<}{\tt string}{\,{>}}\,\, {\tt getClient}();
33
            string FileReader;
41
            Errors Err;
42 };
```

4.4 Файл Errors.cpp

Класс Errors для обработки ошибок и их записи в файл.

#include "Errors.h"

Граф включаемых заголовочных файлов для Errors.cpp:



4.4.1 Подробное описание

Класс Errors для обработки ошибок и их записи в файл.

4.5 Файл Errors.h

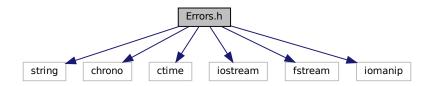
Класс Errors для обработки и записи ошибок.

```
#include <string>
#include <chrono>
```

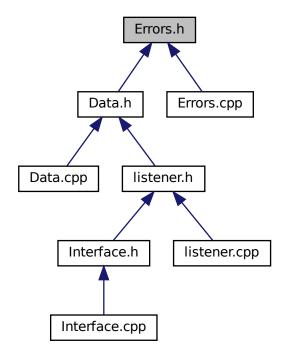
20 Файлы

```
#include <ctime>
#include <iostream>
#include <fstream>
#include <iomanip>
```

Граф включаемых заголовочных файлов для Errors.h:



Граф файлов, в которые включается этот файл:



Классы

• class Errors

Класс для обработки и записи ошибок.

4.5.1 Подробное описание

Класс Errors для обработки и записи ошибок.

4.6 Errors.h 21

4.6 Errors.h

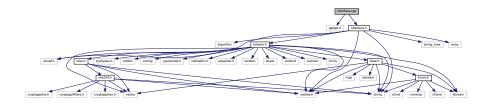
См. документацию.

```
5 #include <string>
6 #include <chrono>
7 #include <ctime>
8 #include <iostream>
9 #include <fstream>
10 #include <iomanip>
12 using namespace std;
13
20 class Errors {
21 public:
     string get File Log();
^{28}
     void set_File_Log(string file);
35
     void error_recording(string flag, string info);
42
43
44 private:
    string File_Log;
47 };
48
```

4.7 Файл Interface.cpp

Класс Interface для обрабатки аргументов командной строки.

```
#include <getopt.h>
#include "Interface.h"
Граф включаемых заголовочных файлов для Interface.cpp:
```



4.7.1 Подробное описание

Класс Interface для обрабатки аргументов командной строки.

4.8 Файл Interface.h

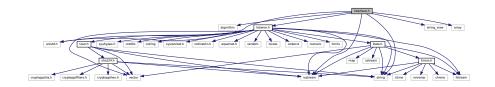
Класс Interface для обработки аргументов командной строки и хранения конфигурационных данных.

```
#include <algorithm>
#include <iostream>
#include <string>
#include <string_view>
#include <array>
#include <unistd.h>
```

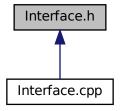
22 Файлы

#include "listener.h"

Граф включаемых заголовочных файлов для Interface.h:



Граф файлов, в которые включается этот файл:



Классы

· class interface

Класс interface для обработки аргументов командной строки и хранения конфигурационных данных.

4.8.1 Подробное описание

Класс Interface для обработки аргументов командной строки и хранения конфигурационных данных.

4.9 Interface.h

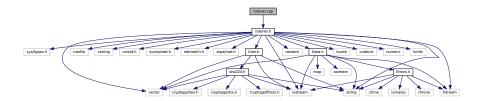

```
27
       string ip;
^{29}
       int port;
       string database;
31
33
       string logfile;
34
       interface(int argc, char** argv);
41
^{46}
       interface() = delete;
^{47}
       {\tt string \ get\_ip}() \ \{
53
54
           return ip;
       }
55
56
62
       int get port() {
63
\frac{64}{69}
       int help();
70 };
```

4.10 Файл listener.cpp

Класс listener для вычисления среднего арифметического значений в векторе.

```
#include "listener.h"
```

Граф включаемых заголовочных файлов для listener.cpp:



4.10.1 Подробное описание

Класс listener для вычисления среднего арифметического значений в векторе.

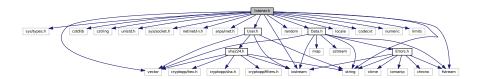
4.11 Файл listener.h

Класс listener для вычисления среднего арифметического значений в векторе.

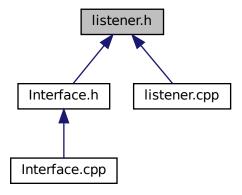
```
#include <sys/types.h>
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fstream>
#include <vector>
#include <random>
#include <string>
#include <locale>
```

24

```
#include <codecvt>
#include <numeric>
#include <limits>
#include "User.h"
#include "Data.h"
Граф включаемых заголовочных файлов для listener.h:
```



Граф файлов, в которые включается этот файл:



Классы

· class listener

Класс listener, реализующий серверную часть приложения. Этот класс отвечает за установление соединения с клиентами, обработку их запросов и отправку ответов.

4.11.1 Подробное описание

Класс listener для вычисления среднего арифметического значений в векторе.

4.12 listener.h

```
{f C}_{M}. Документацию. 1 5 #pragma one 6 #include <sys/types.h>
```

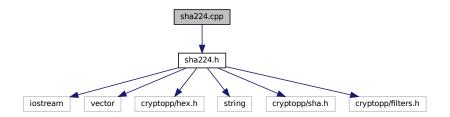
```
7 #include <iostream>
8 #include <cstdlib>
9 #include <cstring>
10 #include <unistd.h>
11 #include <sys/socket.h>
12 #include <netinet/in.h>
13 #include <arpa/inet.h>
14 \ \# include < fstream >
15 #include <vector>
16 \ \# include < random >
17 #include <string>
18 #include <locale>
19 #include <codecvt>
20 #include <numeric>
21 #include inits>
22
23 #include "User.h"
24 #include "Data.h"
26 using namespace std;
32 \text{ class listener} \{
33
      public:
          int interaction(string database, string logFile);
40
41
46
          uint64_t sredn();
^{47}
48
          string get_address();
          void set_address(string address1);
49
          int get_port();
void set_port(int port1);
54
55
56
          vector<uint64_t> get_vec();
58
          void set \_vec(vector < uint64\_t > v);
59
          string get_salt();
void set_salt();
60
61
          pair<vector<string>, vector<string> get_DB_clients();
void set_DB_clients(vector<string> login, vector<string> password);
64
65
      private:
66
          string address;
67
68
          int port;
69
          vector < uint64_t > vec;
70
          string salt;
71
          pair < vector < string >, \ vector < string * DB\_clients;
72
73 };
          Errors Err;
```

4.13 Файл sha224.cpp

Вычисляет SHA224 хэш для заданной строки.

#include "sha224.h"

Граф включаемых заголовочных файлов для sha224.cpp:



Функции

• std::string SHA224_hash (std::string msg)

26 Файлы

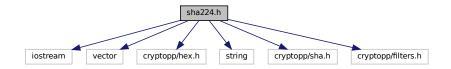
4.13.1 Подробное описание

Вычисляет SHA224 хэш для заданной строки.

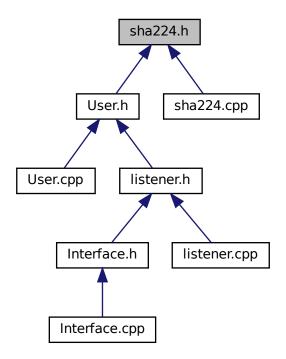
4.14 Файл sha224.h

Вычисляет SHA224 хэш для заданной строки.

```
#include <iostream>
#include <vector>
#include <cryptopp/hex.h>
#include <string>
#include <cryptopp/sha.h>
#include <cryptopp/filters.h>
Граф включаемых заголовочных файлов для sha224.h:
```



Граф файлов, в которые включается этот файл:



4.15 sha 224.h 27

Макросы

- #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1

Функции

• std::string SHA224 hash (std::string msg)

4.14.1 Подробное описание

Вычисляет SHA224 хэш для заданной строки.

4.15 sha224.h

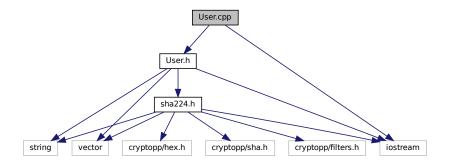
См. документацию.

```
1
5 #include <iostream>
6 #include <vector>
7 #include <cryptopp/hex.h>
8 #define CRYPTOPP_ENABLE_NAMESPACE_WEAK 1
9 #include <string>
10 #include <cryptopp/sha.h>
11 #include <cryptopp/hex.h>
12 #include <cryptopp/filters.h>
13
14 std::string SHA224_hash(std::string msg);
```

4.16 Файл User.cpp

Класс User, представляющий пользователя системы.

```
#include "User.h"
#include <iostream>
Граф включаемых заголовочных файлов для User.cpp:
```



Функции

• void foo ()

Пустая глобальная функция, для Doxygen.

28

4.16.1 Подробное описание

Класс User, представляющий пользователя системы.

4.16.2 Функции

```
4.16.2.1 foo()
```

void foo ()

Пустая глобальная функция, для Doxygen.

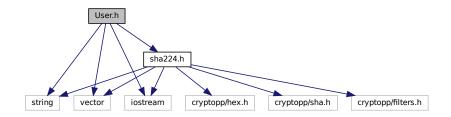
.

4.17 Файл User.h

Класс User, представляющий пользователя системы.

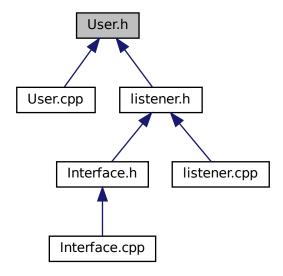
```
#include <string>
#include <vector>
#include <iostream>
#include "sha224.h"
```

Граф включаемых заголовочных файлов для User.h:



4.18 User.h

Граф файлов, в которые включается этот файл:



Классы

• class User

Класс, представляющий пользователя системы.

4.17.1 Подробное описание

Класс User, представляющий пользователя системы.

4.18 User.h

См. документацию.

```
5 #include <string>
6 #include <vector>
7 #include <iostream>
9 #include "sha224.h"
11 using namespace std;
12
21 class User {
22 private:
        string ID;
^{26}
        string hash;
27
28 public:
        bool CheckLogin(vector<string> Db_ID);
bool CheckPassword(vector<string> Db_hash, vector<string> Db_ID, string SALT);
29
30
        string get_ID();
void set_ID(string ID1);
32
33
34
        string get_hash();
void set_hash(string hash1);
35
36
37 };
```

30 Файлы

Предметный указатель

address	interface, 8
listener, 12	Interface.cpp, 21
CIL ID I	Interface.h, 21
CheckPassword	1: 4 10
User, 14	listener, 10 address, 12
Data, 5	DB clients, 12
get FileReader, 6	Err, 12
Data.cpp, 17	get_port, 11
Data.h, 17	interaction, 11
DB clients	port, 12
listener, 12	salt, 12
	sredn , 11
Err	vec, 12
listener, 12	listener.cpp, 23
error_recording	listener.h, 23
Errors, 7	instellerini, 20
Errors, 6	port
error_recording, 7	listener, 12
$get_File_Log, 7$	
$set_File_Log, 7$	salt
Errors.cpp, 19	listener, 12
Errors.h, 19	$\operatorname{set}_{\operatorname{Log}}$
	Errors, 7
foo	set_hash
User.cpp, 28	User, 15
got File Log	set_ID
get_File_Log	User, 15
Errors, 7	sha224.cpp, 25
get_FileReader	sha224.h, 26
Data, 6	sredn
get_hash User, 14	listener, 11
	II 10
get_ID	User, 13
User, 14	CheckPassword, 14
get_ip interface, 9	get_hash, 14
get_port	get_ID, 14
interface, 9	set_hash, 15
listener, 11	set_ID, 15
insteller, 11	CheckLogin, 13
help	User.cpp, 27
interface, 9	foo, 28
	User.h, 28
interaction	vec
listener, 11	listener, 12
interface, 8	110001101, 12
$\operatorname{get}_{-}\operatorname{ip}, 9$	CheckLogin
$get_port, 9$	User, 13
help, 9	