

Разработка сетевого сервера

Создано системой Doxygen 1.9.1

1	Алфавитный указатель классов	1
1.1	Классы	1
2	Список файлов	3
2.1	Файлы	3
3	Классы	5
3.1	Класс Connection	5
3.1.1	Подробное описание	5
3.1.2	Методы	5
3.1.2.1	connection()	5
3.2	Структура Params	6
3.2.1	Подробное описание	7
3.3	Класс UserInterface	7
3.3.1	Подробное описание	7
3.3.2	Конструктор(ы)	7
3.3.2.1	UserInterface()	7
3.3.3	Методы	8
3.3.3.1	getDescription()	8
3.3.3.2	getParams()	8
3.3.3.3	Parser()	8
4	Файлы	11
4.1	Файл connection.cpp	11
4.1.1	Подробное описание	12
4.1.2	Функции	12
4.1.2.1	datawrite()	12
4.1.2.2	findUserInFile()	13
4.2	Файл connection.h	13
4.2.1	Подробное описание	14
4.3	Файл crypto.cpp	14
4.3.1	Подробное описание	15
4.3.2	Функции	15
4.3.2.1	auth()	15
4.4	Файл crypto.h	16
4.4.1	Подробное описание	17
4.4.2	Функции	18
4.4.2.1	auth()	18
4.5	Файл interface.cpp	19
4.5.1	Подробное описание	19
4.6	Файл interface.h	20
4.6.1	Подробное описание	21
4.7	Файл log.cpp	21
4.7.1	Подробное описание	22

4.7.2 Функции	22
4.7.2.1 getCurrentTime()	22
4.7.2.2 logError()	22
4.8 Файл log.h	23
4.8.1 Подробное описание	24
4.8.2 Функции	24
4.8.2.1 getCurrentTime()	24
4.8.2.2 logError()	24
4.9 Файл main.cpp	25
4.9.1 Подробное описание	25
4.9.2 Функции	26
4.9.2.1 main()	26
Предметный указатель	27

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Connection	Класс для управления сетевыми соединениями	5
Params	Структура для хранения параметров командной строки	6
UserInterface	Класс для обработки параметров командной строки	7

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

connection.cpp	Реализация функций сетевого соединения и аутентификации	11
connection.h	Заголовочный файл класса сетевого соединения	13
crypto.cpp	Реализация криптографических функций	14
crypto.h	Заголовочный файл для криптографических функций	16
interface.cpp	Реализация пользовательского интерфейса	19
interface.h	Заголовочный файл пользовательского интерфейса	20
log.cpp	Реализация функций логирования	21
log.h	Заголовочный файл для функций логирования	23
main.cpp	Главный файл серверного приложения	25

Глава 3

Классы

3.1 Класс Connection

Класс для управления сетевыми соединениями

```
#include <connection.h>
```

Открытые статические члены

- static int `connection` (const `Params` *p)
Устанавливает соединение и обрабатывает клиентов

3.1.1 Подробное описание

Класс для управления сетевыми соединениями

3.1.2 Методы

3.1.2.1 `connection()`

```
int Connection::connection (  
    const Params * p ) [static]
```

Устанавливает соединение и обрабатывает клиентов

Основная функция установки соединения и обработки клиентов

Аргументы

in	p	Параметры соединения
----	---	----------------------

Возвращает

0 при успехе, 1 при ошибке

Исключения

system_error	при сетевых ошибках
--------------	---------------------

Аргументы

p	Указатель на параметры соединения
---	-----------------------------------

Возвращает

0 при успехе, 1 при ошибке аутентификации

Исключения

std::system_error	при сетевых ошибках
-------------------	---------------------

Объявления и описания членов классов находятся в файлах:

- [connection.h](#)
- [connection.cpp](#)

3.2 Структура Params

Структура для хранения параметров командной строки

```
#include <interface.h>
```

Открытые атрибуты

- string [inFileName](#)
Имя файла с базой пользователей
- string [inFileJournal](#)
Имя файла журнала
- string [inFileData](#)
Имя файла с данными (резервный параметр)
- string [logFile](#)
Имя файла для логирования ошибок
- int [Port](#)
Порт сервера для прослушивания
- string [Address](#)
IP-адрес сервера

3.2.1 Подробное описание

Структура для хранения параметров командной строки

Содержит все необходимые параметры для настройки сервера и работы с файлами

Объявления и описания членов структуры находятся в файле:

- [interface.h](#)

3.3 Класс `UserInterface`

Класс для обработки параметров командной строки

```
#include <interface.h>
```

Открытые члены

- [UserInterface](#) ()
Конструктор класса [UserInterface](#).
- `bool` [Parser](#) (int argc, const char **argv)
Парсинг аргументов командной строки
- `string` [getDescription](#) ()
Получение описания параметров
- [Params](#) [getParams](#) ()
Получение параметров

3.3.1 Подробное описание

Класс для обработки параметров командной строки

Обеспечивает парсинг, валидацию и хранение параметров командной строки

3.3.2 Конструктор(ы)

3.3.2.1 `UserInterface()`

```
UserInterface::UserInterface ( )
```

Конструктор класса [UserInterface](#).

Инициализирует опции командной строки с обязательными параметрами

Инициализирует опции командной строки с обязательными параметрами и значениями по умолчанию < Опция для вывода справки

< Файл логирования (по умолчанию journal.txt)

< Обязательный параметр: файл базы пользователей

< Обязательный параметр: файл журнала

< Обязательный параметр: порт сервера

< Адрес сервера (по умолчанию 127.0.0.1)

3.3.3 Методы

3.3.3.1 getDescription()

```
std::string UserInterface::getDescription ( )
```

Получение описания параметров

Возвращает

Строка с описанием поддерживаемых опций

3.3.3.2 getParams()

```
Params UserInterface::getParams ( ) [inline]
```

Получение параметров

Возвращает

Структура [Params](#) с распарсенными значениями

3.3.3.3 Parser()

```
bool UserInterface::Parser (
    int argc,
    const char ** argv )
```

Парсинг аргументов командной строки

Аргументы

in	argc	Количество аргументов
in	argv	Массив аргументов

Возвращает

true если парсинг успешен, false если требуется показать справку

Аргументы

in	argc	Количество аргументов
in	argv	Массив аргументов

Возвращает

true если парсинг успешен, false если требуется показать справку

Исключения

exception	при ошибках парсинга или отсутствии обязательных параметров
-----------	---

Объявления и описания членов классов находятся в файлах:

- [interface.h](#)
- [interface.cpp](#)

Глава 4

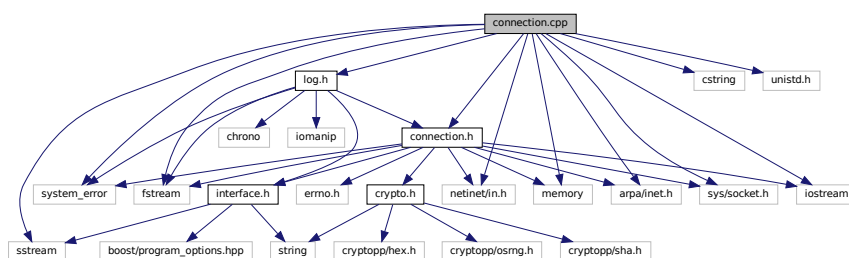
Файлы

4.1 Файл connection.cpp

Реализация функций сетевого соединения и аутентификации

```
#include "connection.h"
#include "log.h"
#include <fstream>
#include <sstream>
#include <iostream>
#include <cstring>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <memory>
#include <system_error>
```

Граф включаемых заголовочных файлов для connection.cpp:



Функции

- bool [findUserInFile](#) (const std::string &filename, const std::string &username, std::string &password)
Поиск пользователя в файле по логину
- int [datawrite](#) (int s, int client_socket, const [Params](#) *p)
Обработка передачи данных после успешной аутентификации

4.1.1 Подробное описание

Реализация функций сетевого соединения и аутентификации

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

Серверная часть приложения, обрабатывающая подключения клиентов, аутентификацию и вычисления. Реализует полный цикл работы сервера: создание сокета, привязка к порту, прослушивание соединений, обработка клиентских запросов, аутентификация пользователей и выполнение вычислительных операций.

4.1.2 Функции

4.1.2.1 datawrite()

```
int datawrite (
    int s,
    int client_socket,
    const Params * p )
```

Обработка передачи данных после успешной аутентификации

Аргументы

s	Дескриптор основного сокета сервера
client_socket	Дескриптор сокета клиента
p	Указатель на структуру параметров соединения

Возвращает

0 при успешном выполнении

Исключения

std::system_error	при ошибках сетевого взаимодействия
-------------------	-------------------------------------

4.1.2.2 findUserInFile()

```
bool findUserInFile (
    const std::string & filename,
    const std::string & username,
    std::string & password )
```

Поиск пользователя в файле по логину

Аргументы

	filename	Путь к файлу с данными пользователей
	username	Логин пользователя для поиска
out	password	Найденный пароль пользователя

Возвращает

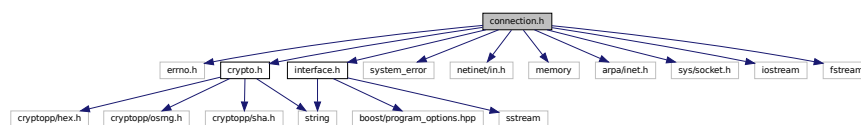
true если пользователь найден, иначе false

4.2 Файл connection.h

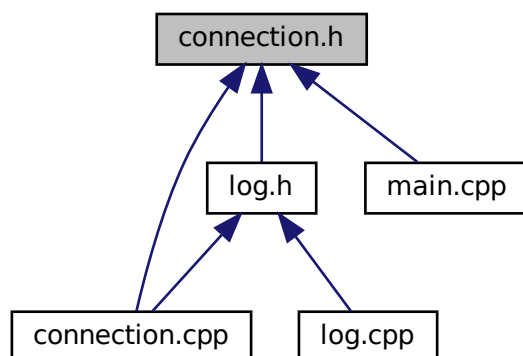
Заголовочный файл класса сетевого соединения

```
#include "errno.h"
#include "crypto.h"
#include "interface.h"
#include <system_error>
#include <netinet/in.h>
#include <memory>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <iostream>
#include <fstream>
```

Граф включаемых заголовочных файлов для connection.h:



Граф файлов, в которые включается этот файл:



Классы

- class [Connection](#)
Класс для управления сетевыми соединениями

Макросы

- `#define BUFFER_SIZE 1024`
Размер буфера для сетевого обмена

4.2.1 Подробное описание

Заголовочный файл класса сетевого соединения

Автор

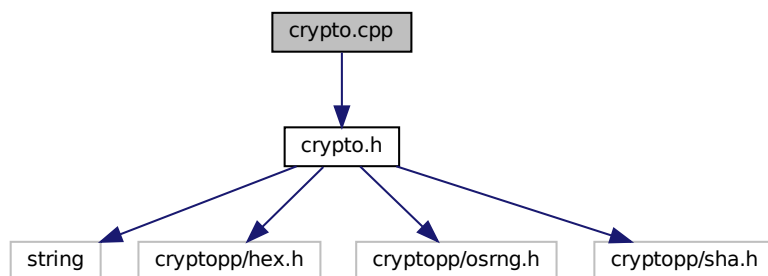
Веселов А.Н.

4.3 Файл crypto.cpp

Реализация криптографических функций

```
#include "crypto.h"
```

Граф включаемых заголовочных файлов для crypto.cpp:



Функции

- string [auth](#) (string salt, string pass)

Функция аутентификации с использованием SHA224 хеширования

4.3.1 Подробное описание

Реализация криптографических функций

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

Реализует функцию хеширования с использованием алгоритма SHA224

4.3.2 Функции

4.3.2.1 auth()

```
string auth (  
    string salt,  
    string pass )
```

Функция аутентификации с использованием SHA224 хеширования

Аргументы

in	salt	Соль для хеширования
in	pass	Пароль пользователя

Возвращает

Хеш-строка в hex-формате

Вычисляет SHA224 хеш от конкатенации соли и пароля, результат возвращает в hex-формате < Объект для вычисления SHA224 хеша

< Результирующий хеш

< Конкатенация соли и пароля

< Использование SHA224 для хеширования

< Кодирование результата в hex

< Возврат хеша в hex-формате

4.4 Файл crypto.h

Заголовочный файл для криптографических функций

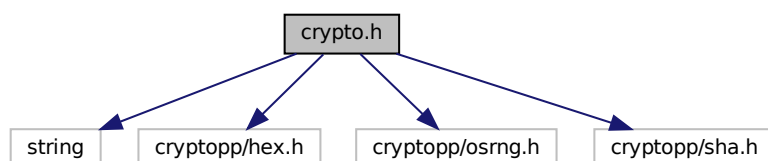
```
#include <string>
```

```
#include <cryptopp/hex.h>
```

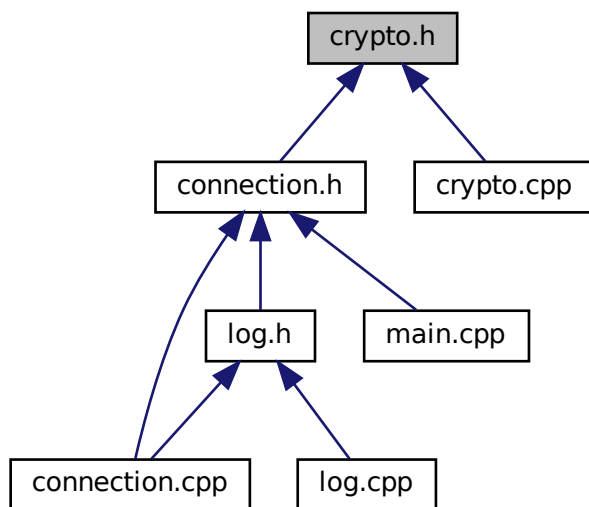
```
#include <cryptopp/osrng.h>
```

```
#include <cryptopp/sha.h>
```

Граф включаемых заголовочных файлов для crypto.h:



Граф файлов, в которые включается этот файл:



Функции

- string `auth` (string salt, string pass)
Функция аутентификации с использованием SHA224 хеширования

4.4.1 Подробное описание

Заголовочный файл для криптографических функций

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

Содержит объявления функций для хеширования с использованием библиотеки CryptoPP

4.4.2 Функции

4.4.2.1 auth()

```
string auth (
    string salt,
    string pass )
```

Функция аутентификации с использованием SHA224 хеширования

Аргументы

in	salt	Соль для хеширования
in	pass	Пароль пользователя

Возвращает

Хеш-строка в hex-формате

Использует алгоритм SHA224 для создания хеша от конкатенации соли и пароля

Аргументы

in	salt	Соль для хеширования
in	pass	Пароль пользователя

Возвращает

Хеш-строка в hex-формате

Вычисляет SHA224 хеш от конкатенации соли и пароля, результат возвращает в hex-формате <
Объект для вычисления SHA224 хеша

< Результирующий хеш

< Конкатенация соли и пароля

< Использование SHA224 для хеширования

< Кодирование результата в hex

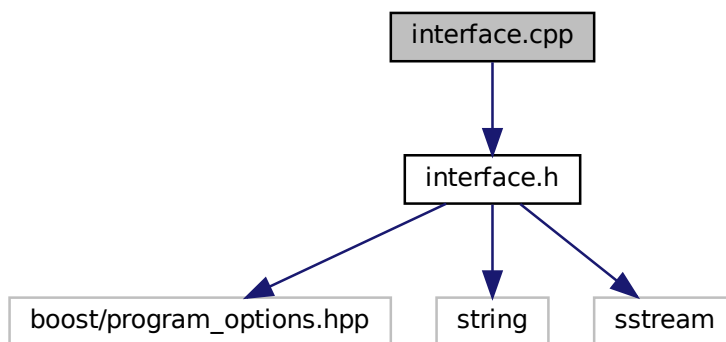
< Возврат хеша в hex-формате

4.5 Файл interface.cpp

Реализация пользовательского интерфейса

```
#include "interface.h"
```

Граф включаемых заголовочных файлов для interface.cpp:



4.5.1 Подробное описание

Реализация пользовательского интерфейса

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

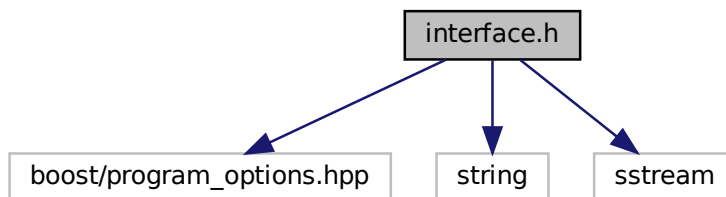
Реализует функциональность парсинга и валидации параметров командной строки

4.6 Файл interface.h

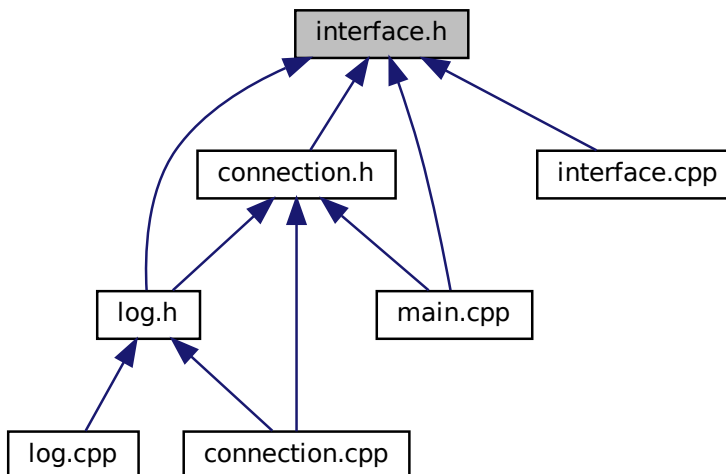
Заголовочный файл пользовательского интерфейса

```
#include <boost/program_options.hpp>
#include <string>
#include <sstream>
```

Граф включаемых заголовочных файлов для interface.h:



Граф файлов, в которые включается этот файл:



Классы

- struct [Params](#)
Структура для хранения параметров командной строки
- class [UserInterface](#)
Класс для обработки параметров командной строки

4.6.1 Подробное описание

Заголовочный файл пользовательского интерфейса

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

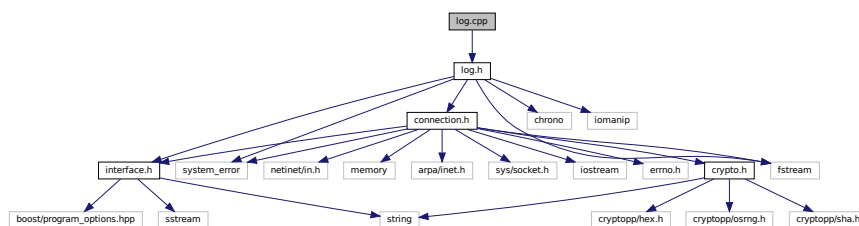
Использует boost::program_options для парсинга аргументов командной строки

4.7 Файл log.cpp

Реализация функций логирования

```
#include "log.h"
```

Граф включаемых заголовочных файлов для log.cpp:



Функции

- `std::string getCurrentTime ()`
Получение текущего времени в формате строки
- `void logError (const std::string &logFile, const std::string &errorMessage)`
Запись ошибки в лог-файл

4.7.1 Подробное описание

Реализация функций логирования

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

Реализует функции для записи логов с временными метками

4.7.2 Функции

4.7.2.1 getCurrentTime()

```
std::string getCurrentTime ( )
```

Получение текущего времени в формате строки

Возвращает

Строка с текущим временем в формате "ГГГГ-ММ-ДД ЧЧ:ММ:СС.ммм"

Форматирует текущее системное время с точностью до миллисекунд

4.7.2.2 logError()

```
void logError (
    const std::string & logFile,
    const std::string & errorMessage )
```

Запись ошибки в лог-файл

Аргументы

in	logFile	Имя файла для логирования
in	errorMessage	Сообщение об ошибке для записи

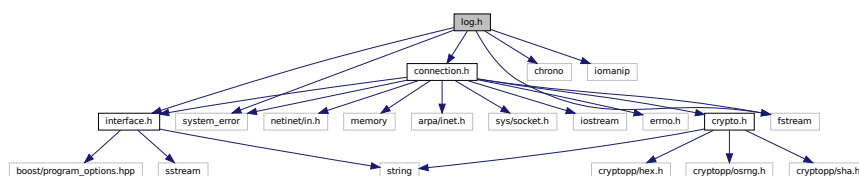
Добавляет запись в лог-файл с временной меткой и префиксом "ERROR"

4.8 Файл log.h

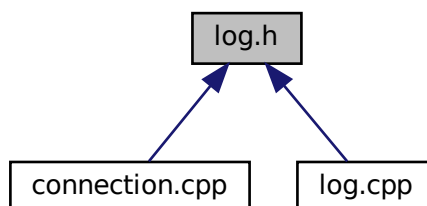
Заголовочный файл для функций логирования

```
#include "connection.h"
#include "interface.h"
#include <fstream>
#include <system_error>
#include <chrono>
#include <iomanip>
```

Граф включаемых заголовочных файлов для log.h:



Граф файлов, в которые включается этот файл:



Функции

- `std::string getCurrentTime ()`
Получение текущего времени в формате строки
- `void logError (const std::string &logFile, const std::string &errorMessage)`
Запись ошибки в лог-файл

4.8.1 Подробное описание

Заголовочный файл для функций логирования

Автор

Веселов А.Н.

Версия

1.0

Дата

2025

Авторство

ИБСТ ПГУ

Содержит объявления функций для работы с системой логирования

4.8.2 Функции

4.8.2.1 `getCurrentTime()`

```
std::string getCurrentTime ( )
```

Получение текущего времени в формате строки

Возвращает

Строка с текущим временем в формате "ГГГГ-ММ-ДД ЧЧ:ММ:СС.ммм"

Строка с текущим временем в формате "ГГГГ-ММ-ДД ЧЧ:ММ:СС.ммм"

Форматирует текущее системное время с точностью до миллисекунд

4.8.2.2 `logError()`

```
void logError (
    const std::string & logFile,
    const std::string & errorMessage )
```

Запись ошибки в лог-файл

Аргументы

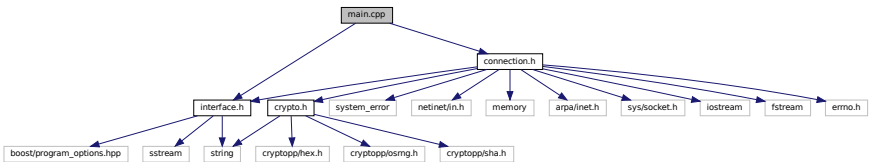
in	logFile	Имя файла для логирования
in	errorMessage	Сообщение об ошибке для записи
in	logFile	Имя файла для логирования
in	errorMessage	Сообщение об ошибке для записи

Добавляет запись в лог-файл с временной меткой и префиксом "ERROR"

4.9 Файл main.cpp

Главный файл серверного приложения

```
#include "connection.h"
#include "interface.h"
Граф включаемых заголовочных файлов для main.cpp:
```



Функции

- int `main` (int argc, const char **argv)
 Главная функция серверного приложения

4.9.1 Подробное описание

Главный файл серверного приложения

Автор
 Веселов А.Н.

Версия
 1.0

Дата
 2025

Точка входа в программу, обработка параметров и запуск сервера

4.9.2 Функции

4.9.2.1 main()

```
int main (  
    int argc,  
    const char ** argv )
```

Главная функция серверного приложения

Аргументы

in	argc	Количество аргументов командной строки
in	argv	Массив аргументов командной строки

Возвращает

0 при успешном выполнении, 1 при ошибке параметров

Предметный указатель

- auth
 - crypto.cpp, [15](#)
 - crypto.h, [18](#)
- Connection, [5](#)
 - connection, [5](#)
- connection
 - Connection, [5](#)
- connection.cpp, [11](#)
 - datawrite, [12](#)
 - findUserInFile, [13](#)
- connection.h, [13](#)
- crypto.cpp, [14](#)
 - auth, [15](#)
- crypto.h, [16](#)
 - auth, [18](#)
- datawrite
 - connection.cpp, [12](#)
- findUserInFile
 - connection.cpp, [13](#)
- getCurrentTime
 - log.cpp, [22](#)
 - log.h, [24](#)
- getDescription
 - UserInfo, [8](#)
- getParams
 - UserInfo, [8](#)
- interface.cpp, [19](#)
- interface.h, [20](#)
- log.cpp, [21](#)
 - getCurrentTime, [22](#)
 - logError, [22](#)
- log.h, [23](#)
 - getCurrentTime, [24](#)
 - logError, [24](#)
- logError
 - log.cpp, [22](#)
 - log.h, [24](#)
- main
 - main.cpp, [26](#)
- main.cpp, [25](#)
 - main, [26](#)
- Params, [6](#)
- Parser
 - UserInfo, [8](#)
- UserInfo, [8](#)
 - UserInfo, [7](#)
 - getDescription, [8](#)
 - getParams, [8](#)
 - Parser, [8](#)
 - UserInfo, [7](#)