

Kursovaia\_Server

Создано системой Doxygen 1.9.4



1 Иерархический список классов	1
1.1 Иерархия классов	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Классы	7
4.1 Класс Calculator	7
4.1.1 Подробное описание	7
4.1.2 Конструктор(ы)	7
4.1.2.1 Calculator()	7
4.1.3 Методы	8
4.1.3.1 send_res()	8
4.2 Класс Client_Communicate	8
4.2.1 Подробное описание	8
4.2.2 Методы	9
4.2.2.1 connection()	9
4.2.2.2 generate_salt()	9
4.2.2.3 sha256()	9
4.3 Класс Connector_to_base	10
4.3.1 Подробное описание	10
4.3.2 Методы	10
4.3.2.1 connect_to_base()	10
4.3.2.2 get_data()	11
4.4 Класс crit_err	11
4.4.1 Подробное описание	12
4.4.2 Конструктор(ы)	12
4.4.2.1 crit_err()	12
4.5 Класс Interface	12
4.5.1 Подробное описание	12
4.5.2 Методы	13
4.5.2.1 comm_proc()	13
4.6 Класс Logger	13
4.6.1 Подробное описание	14
4.6.2 Конструктор(ы)	14
4.6.2.1 Logger()	14
4.6.3 Методы	14
4.6.3.1 getCurrentDateTime()	14
4.6.3.2 set_path()	15
4.6.3.3 writelog()	15
4.7 Класс no_crit_err	15

---

4.7.1 Подробное описание . . . . .	16
4.7.2 Конструктор(ы) . . . . .	16
4.7.2.1 no_crit_err() . . . . .	16
5 Файлы . . . . .	19
5.1 Файл Calculator.h . . . . .	19
5.1.1 Подробное описание . . . . .	20
5.2 Calculator.h . . . . .	20
5.3 Файл Client_Communicate.h . . . . .	21
5.3.1 Подробное описание . . . . .	21
5.4 Client_Communicate.h . . . . .	22
5.5 Файл Connector_to_base.h . . . . .	22
5.5.1 Подробное описание . . . . .	23
5.6 Connector_to_base.h . . . . .	23
5.7 Файл Errors.h . . . . .	24
5.7.1 Подробное описание . . . . .	25
5.8 Errors.h . . . . .	25
5.9 Файл Interface.h . . . . .	25
5.9.1 Подробное описание . . . . .	26
5.10 Interface.h . . . . .	26
5.11 Файл Logger.h . . . . .	27
5.11.1 Подробное описание . . . . .	27
5.12 Logger.h . . . . .	28
5.13 Файл main.cpp . . . . .	28
5.13.1 Подробное описание . . . . .	29
5.13.2 Функции . . . . .	29
5.13.2.1 main() . . . . .	29
Предметный указатель . . . . .	31

# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

Calculator . . . . .	7
Client_Communicate . . . . .	8
Connector_to_base . . . . .	10
Interface . . . . .	12
Logger . . . . .	13
std::runtime_error	
crit_err . . . . .	11
no_crit_err . . . . .	15



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">Calculator</a>	Класс для выполнения вычислений . . . . .	7
<a href="#">Client_Communicate</a>	Класс для управления коммуникацией клиента . . . . .	8
<a href="#">Connector_to_base</a>	Управляет подключениями к базе данных . . . . .	10
<a href="#">crit_err</a>	Представляет критическую ошибку . . . . .	11
<a href="#">Interface</a>	Управляет процессом коммуникации . . . . .	12
<a href="#">Logger</a>	Простой класс логирования . . . . .	13
<a href="#">no_crit_err</a>	Представляет некритическую ошибку . . . . .	15





## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

<a href="#">Calculator.h</a>	
Заголовок для класса <a href="#">Calculator</a>	19
<a href="#">Client_Communicate.h</a>	
Заголовок для класса <a href="#">Client_Communicate</a>	21
<a href="#">Connector_to_base.h</a>	
Заголовок для класса <a href="#">Connector_to_base</a>	22
<a href="#">Errors.h</a>	
Содержит классы для обработки ошибок	24
<a href="#">Interface.h</a>	
Заголовок для класса <a href="#">Interface</a>	25
<a href="#">Logger.h</a>	
Заголовок для класса <a href="#">Logger</a>	27
<a href="#">main.cpp</a>	
Основная точка входа для серверного приложения	28



## Глава 4

# Классы

### 4.1 Класс Calculator

Класс для выполнения вычислений

```
#include <Calculator.h>
```

Открытые члены

- [Calculator](#) (std::vector< int64\_t > input\_data)  
Конструктор объекта [Calculator](#).
- int64\_t [send\\_res](#) ()  
Получает результат вычислений

Открытые атрибуты

- int64\_t results  
Хранит результат вычислений

#### 4.1.1 Подробное описание

Класс для выполнения вычислений

#### 4.1.2 Конструктор(ы)

##### 4.1.2.1 Calculator()

```
Calculator::Calculator (  
    std::vector< int64_t > input_data )
```

Конструктор объекта [Calculator](#).

## Аргументы

input_data	Вектор входных данных для вычислений
------------	--------------------------------------

## 4.1.3 Методы

## 4.1.3.1 send\_res()

```
int64_t Calculator::send_res ( )
```

Получает результат вычислений

Возвращает

Результат вычислений

Объявления и описания членов классов находятся в файлах:

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

## 4.2 Класс Client\_Communicate

Класс для управления коммуникацией клиента

```
#include <Client_Communicate.h>
```

## Открытые члены

- int [connection](#) (int port, std::map< std::string, std::string > database, [Logger](#) \*l1)  
Управляет попыткой подключения

## Открытые статические члены

- static std::string [sha256](#) (std::string input\_str)  
Вычисляет SHA-256 хеш строки
- static std::string [generate\\_salt](#) ()  
Генерирует случайную соль

## 4.2.1 Подробное описание

Класс для управления коммуникацией клиента

### 4.2.2 Методы

#### 4.2.2.1 connection()

```
int Client_Communicate::connection (
    int port,
    std::map< std::string, std::string > database,
    Logger * l1 )
```

Управляет попыткой подключения

Аргументы

port	Порт для подключения
database	Карта, представляющая базу данных
l1	Объект логгера для записи логов

Возвращает

Код состояния операции подключения

#### 4.2.2.2 generate\_salt()

```
std::string Client_Communicate::generate_salt ( ) [static]
```

Генерирует случайную соль

Возвращает

Сгенерированная строка соли

#### 4.2.2.3 sha256()

```
std::string Client_Communicate::sha256 (
    std::string input_str ) [static]
```

Вычисляет SHA-256 хеш строки

Аргументы

input_str	Строка для хеширования
-----------	------------------------

Возвращает

SHA-256 хеш

Объявления и описания членов классов находятся в файлах:

- [Client\\_Communicate.h](#)
- [Client\\_Communicate.cpp](#)

### 4.3 Класс Connector\_to\_base

Управляет подключениями к базе данных

```
#include <Connector_to_base.h>
```

Открытые члены

- `int connect_to_base (std::string base_file="/home/stud/local_git/Kursovaia_Server/test/base.txt")`  
Подключается к базе данных
- `std::map< std::string, std::string > get_data ()`  
Получает данные из базы данных

Закрытые данные

- `std::map< std::string, std::string > data_base`

#### 4.3.1 Подробное описание

Управляет подключениями к базе данных

#### 4.3.2 Методы

##### 4.3.2.1 connect\_to\_base()

```
int Connector_to_base::connect_to_base (
    std::string base_file = "/home/stud/local_git/Kursovaia_Server/test/base.txt" )
```

Подключается к базе данных

Аргументы

base_file	Путь к файлу базы данных
-----------	--------------------------

Возвращает

Код состояния подключения

#### 4.3.2.2 get\_data()

```
std::map< std::string, std::string > Connector_to_base::get_data ( )
```

Получает данные из базы данных

Возвращает

Карта, содержащая записи базы данных

Объявления и описания членов классов находятся в файлах:

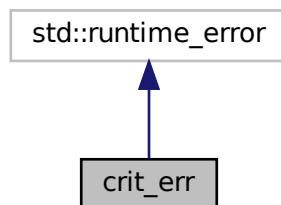
- [Connector\\_to\\_base.h](#)
- [Connector\\_to\\_base.cpp](#)

## 4.4 Класс crit\_err

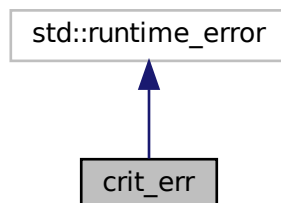
Представляет критическую ошибку

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

Граф наследования:crit\_err:



Граф связей класса crit\_err:



## Открытые члены

- [crit\\_err](#) (const std::string &s)  
Конструктор объекта [crit\\_err](#).

### 4.4.1 Подробное описание

Представляет критическую ошибку

### 4.4.2 Конструктор(ы)

#### 4.4.2.1 crit\_err()

```
crit_err::crit_err (
    const std::string & s ) [inline]
```

Конструктор объекта [crit\\_err](#).

Аргументы

s	Сообщение об ошибке
---	---------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)

## 4.5 Класс Interface

Управляет процессом коммуникации

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

## Открытые члены

- Interface ()=default  
Конструктор по умолчанию
- int [comm\\_proc](#) (int argc, const char \*\*argv)  
Обрабатывает аргументы командной строки

### 4.5.1 Подробное описание

Управляет процессом коммуникации



### 4.5.2 Методы

#### 4.5.2.1 comm\_proc()

```
int Interface::comm_proc (
    int argc,
    const char ** argv )
```

Обрабатывает аргументы командной строки

Аргументы

argc	Количество аргументов
argv	Вектор аргументов

Возвращает

Код состояния обработки

Объявления и описания членов классов находятся в файлах:

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

## 4.6 Класс Logger

Простой класс логирования

```
#include <Logger.h>
```

Открытые члены

- int [writelog](#) (std::string s)  
Записывает лог
- int [set\\_path](#) (std::string path\_file)  
Устанавливает путь к файлу логирования
- [Logger](#) ()  
Конструктор по умолчанию
- [Logger](#) (std::string s)  
Конструктор класса [Logger](#) с заданным путем к файлу логов

Закрытые статические члены

- static std::string [getCurrentDateTime](#) (std::string s)  
Получает текущие дату и время в виде строки

## Закрытые данные

- `std::string path_to_logfile`

### 4.6.1 Подробное описание

Простой класс логирования

### 4.6.2 Конструктор(ы)

#### 4.6.2.1 `Logger()`

```
Logger::Logger (  
    std::string s ) [inline]
```

Конструктор класса [Logger](#) с заданным путем к файлу логов

Аргументы

s	Путь к файлу логов
---	--------------------

### 4.6.3 Методы

#### 4.6.3.1 `getCurrentDateTime()`

```
string Logger::getCurrentDateTime (  
    std::string s ) [static], [private]
```

Получает текущие дату и время в виде строки

Аргументы

s	Формат строки
---	---------------

Возвращает

Текущие дата и время

## 4.6.3.2 set\_path()

```
int Logger::set_path (
    std::string path_file )
```

Устанавливает путь к файлу логирования

Аргументы

path_file	Путь к файлу логов
-----------	--------------------

Возвращает

Код состояния установки пути

## 4.6.3.3 writelog()

```
int Logger::writelog (
    std::string s )
```

Записывает лог

Аргументы

s	Сообщение для логирования
---	---------------------------

Возвращает

Код состояния операции записи

Объявления и описания членов классов находятся в файлах:

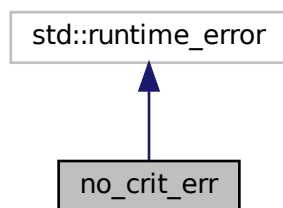
- [Logger.h](#)
- [Logger.cpp](#)

## 4.7 Класс no\_crit\_err

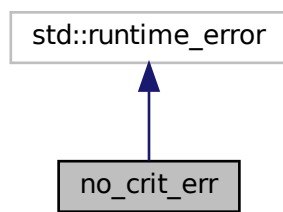
Представляет не критическую ошибку

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

Граф наследования: `no_crit_err`:



Граф связей класса `no_crit_err`:



Открытые члены

- `no_crit_err` (`const std::string &s`)  
Конструктор объекта `no_crit_err`.

#### 4.7.1 Подробное описание

Представляет некритическую ошибку

#### 4.7.2 Конструктор(ы)

##### 4.7.2.1 `no_crit_err()`

```
no_crit_err::no_crit_err (  
    const std::string & s ) [inline]
```

Конструктор объекта `no_crit_err`.

Аргументы

s	Сообщение об ошибке
---	---------------------

Объявления и описания членов класса находятся в файле:

- [Errors.h](#)



## Глава 5

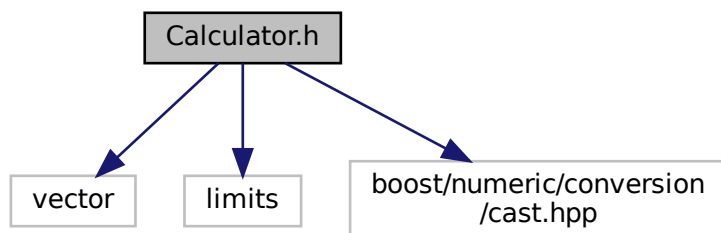
# Файлы

### 5.1 Файл Calculator.h

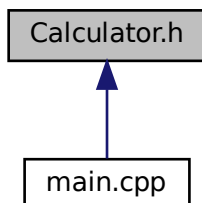
Заголовок для класса `Calculator`.

```
#include <vector>
#include <limits>
#include <boost/numeric/conversion/cast.hpp>
```

Граф включаемых заголовочных файлов для Calculator.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Calculator](#)

Класс для выполнения вычислений

### 5.1.1 Подробное описание

Заголовок для класса [Calculator](#).

Этот файл содержит определение класса [Calculator](#), который выполняет вычисления на входных данных.

Версия

1.0

Дата

12.12.2024

Автор

Пономарев А.А

Предупреждения

Убедитесь в корректной обработке больших наборов данных

## 5.2 Calculator.h

[См. документацию.](#)

```
1
10 #pragma once
11 #include <vector>
12 #include <limits>
13 #include <boost/numeric/conversion/cast.hpp>
14
18 class Calculator {
19 public:
21     int64_t results;
22
26     Calculator(std::vector<int64_t> input_data);
27
31     int64_t send_res();
32 };
```

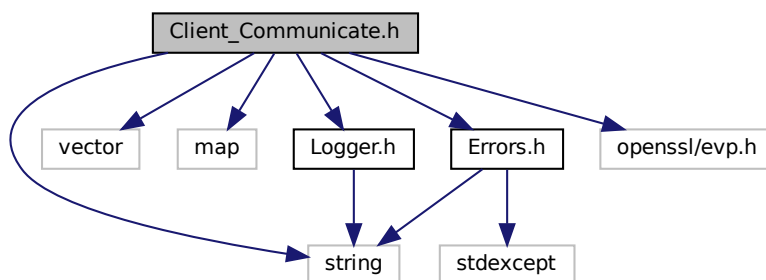


## 5.3 Файл Client\_Communicate.h

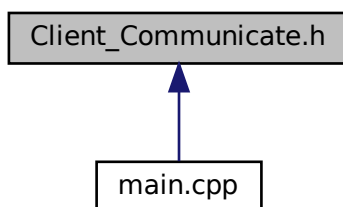
Заголовок для класса `Client_Communicate`.

```
#include <string>
#include <vector>
#include <map>
#include "Logger.h"
#include "Errors.h"
#include <openssl/evp.h>
```

Граф включаемых заголовочных файлов для Client\_Communicate.h:



Граф файлов, в которые включается этот файл:



### Классы

- class `Client_Communicate`

Класс для управления коммуникацией клиента

#### 5.3.1 Подробное описание

Заголовок для класса `Client_Communicate`.

Этот файл содержит определения утилит для клиент-серверной коммуникации.

Версия

1.0

Автор

Пономарев А.А.

Дата

12.12.2024

## 5.4 Client\_Communicate.h

[См. документацию.](#)

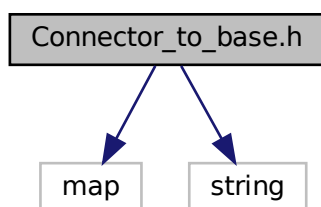
```
1
9 #pragma once
10 #include <string>
11 #include <vector>
12 #include <map>
13 #include "Logger.h"
14 #include "Errors.h"
15 #include <openssl/evp.h>
16
20 class Client_Communicate {
21 public:
26     static std::string sha256(std::string input_str);
27
31     static std::string generate_salt();
32
39     int connection(int port, std::map<std::string, std::string> database, Logger\* l1);
40 };
```

## 5.5 Файл Connector\_to\_base.h

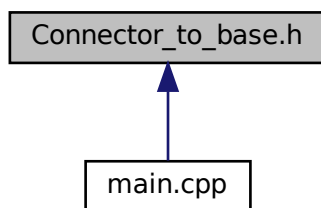
Заголовок для класса [Connector\\_to\\_base](#).

```
#include <map>
#include <string>
```

Граф включаемых заголовочных файлов для Connector\_to\_base.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Connector\\_to\\_base](#)

Управляет подключениями к базе данных

### 5.5.1 Подробное описание

Заголовок для класса [Connector\\_to\\_base](#).

Этот файл включает класс для управления подключениями к базе данных.

Версия

1.0

Автор

Пономарев А.А

Дата

12.12.2024

## 5.6 Connector\_to\_base.h

[См. документацию.](#)

```
1
9 #pragma once
10 #include <map>
11 #include <string>
12
16 class Connector_to_base {
17 private:
18     std::map<std::string, std::string> data_base;
19
20 public:
25     int connect_to_base(std::string base_file = "/home/stud/local_git/Kursovaia_Server/test/base.txt");
26
30     std::map<std::string, std::string> get_data();
31 };
```

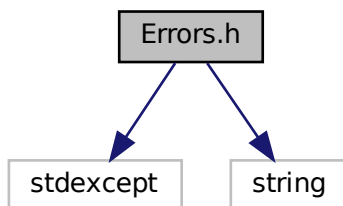
## 5.7 Файл Errors.h

Содержит классы для обработки ошибок

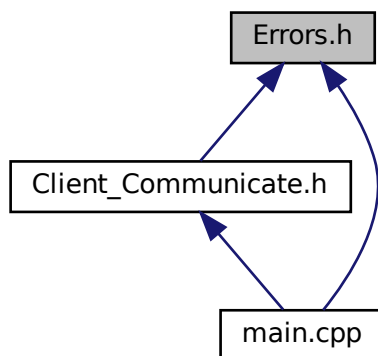
```
#include <stdexcept>
```

```
#include <string>
```

Граф включаемых заголовочных файлов для Errors.h:



Граф файлов, в которые включается этот файл:



### Классы

- class `crit_err`

Представляет критическую ошибку

- class `no_crit_err`

Представляет некритическую ошибку

### 5.7.1 Подробное описание

Содержит классы для обработки ошибок

Определяет критические и некритические ошибки для приложения

Версия

1.0

Автор

Пономарев А.А

Дата

12.12.2024

## 5.8 Errors.h

[См. документацию.](#)

```
1
9 #pragma once
10 #include <stdexcept>
11 #include <string>
12
16 class crit_err : public std::runtime_error {
17 public:
21     crit_err(const std::string& s) : std::runtime_error(s) {}
22 };
23
27 class no_crit_err : public std::runtime_error {
28 public:
32     no_crit_err(const std::string& s) : std::runtime_error(s) {}
33 };
```

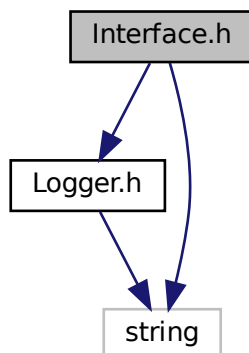
## 5.9 Файл Interface.h

Заголовок для класса [Interface](#).

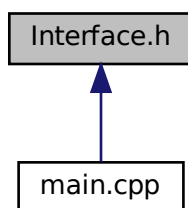
```
#include "Logger.h"
```

```
#include <string>
```

Граф включаемых заголовочных файлов для Interface.h:



Граф файлов, в которые включается этот файл:



## Классы

- class [Interface](#)  
Управляет процессом коммуникации

### 5.9.1 Подробное описание

Заголовок для класса [Interface](#).

Определяет интерфейс для обработки коммуникации

Версия

1.0

Автор

Пономарев А.А.

Дата

12.12.2024

## 5.10 Interface.h

[См. документацию.](#)

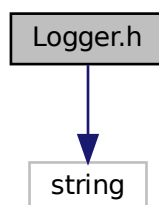
```
1
9 #pragma once
10 #include "Logger.h"
11 #include <string>
12
16 class Interface {
17 public:
19     Interface() = default;
20
26     int comm_proc(int argc, const char** argv);
27 };
```

## 5.11 Файл Logger.h

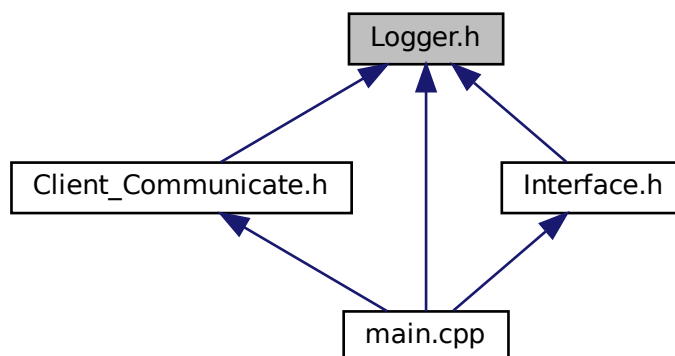
Заголовок для класса `Logger`.

```
#include <string>
```

Граф включаемых заголовочных файлов для `Logger.h`:



Граф файлов, в которые включается этот файл:



### Классы

- class `Logger`

Простой класс логирования

#### 5.11.1 Подробное описание

Заголовок для класса `Logger`.

Обеспечивает функциональность логирования для приложения

Версия

1.0

Автор

Пономарев А.А

Дата

12.12.2024

## 5.12 Logger.h

[См. документацию.](#)

```

1
9 #pragma once
10 #include <string>
11
12 class Logger {
13     static std::string getCurrentDateTime(std::string s);
14     std::string path_to_logfile;
15
16 public:
17     int writelog(std::string s);
18
19     int set_path(std::string path_file);
20
21     Logger() { path_to_logfile = " "; };
22     Logger(std::string s) { path_to_logfile = s; };
23 };

```

## 5.13 Файл main.cpp

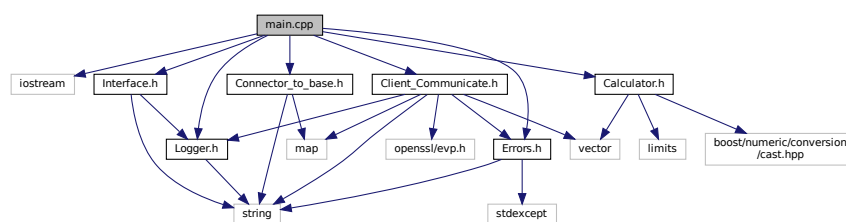
Основная точка входа для серверного приложения

```

#include <iostream>
#include "Connector_to_base.h"
#include "Interface.h"
#include "Client_Communicate.h"
#include "Calculator.h"
#include "Errors.h"
#include "Logger.h"

```

Граф включаемых заголовочных файлов для main.cpp:





## Функции

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

Основная функция для запуска сервера

### 5.13.1 Подробное описание

Основная точка входа для серверного приложения

Инициализирует сервер и запускает процесс коммуникации.

Версия

1.0

Автор

Пономарев А.А

Дата

12.12.2024

### 5.13.2 Функции

#### 5.13.2.1 main()

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

Основная функция для запуска сервера

Аргументы

argc	Количество аргументов
argv	Вектор аргументов

Возвращает

Код состояния работы сервера



# Предметный указатель

- Calculator, [7](#)
  - Calculator, [7](#)
  - send\_res, [8](#)
- Calculator.h, [19](#)
- Client\_Communicate, [8](#)
  - connection, [9](#)
  - generate\_salt, [9](#)
  - sha256, [9](#)
- Client\_Communicate.h, [21](#)
- comm\_proc
  - Interface, [13](#)
- connect\_to\_base
  - Connector\_to\_base, [10](#)
- connection
  - Client\_Communicate, [9](#)
- Connector\_to\_base, [10](#)
  - connect\_to\_base, [10](#)
  - get\_data, [11](#)
- Connector\_to\_base.h, [22](#)
- crit\_err, [11](#)
  - crit\_err, [12](#)
- Errors.h, [24](#)
- generate\_salt
  - Client\_Communicate, [9](#)
- get\_data
  - Connector\_to\_base, [11](#)
- getCurrentDateTime
  - Logger, [14](#)
- Interface, [12](#)
  - comm\_proc, [13](#)
- Interface.h, [25](#)
- Logger, [13](#)
  - getCurrentDateTime, [14](#)
  - Logger, [14](#)
  - set\_path, [14](#)
  - writelog, [15](#)
- Logger.h, [27](#)
- main
  - main.cpp, [29](#)
- main.cpp, [28](#)
  - main, [29](#)
- no\_crit\_err, [15](#)
  - no\_crit\_err, [16](#)
- send\_res
  - Calculator, [8](#)
  - set\_path
    - Logger, [14](#)
  - sha256
    - Client\_Communicate, [9](#)
  - writelog
    - Logger, [15](#)