Клиентская часть

cpp

#include <boost/asio.hpp>

#include <iostream>

using boost::asio::ip::tcp;

int main() {

try {

// Создание контекста ввода-вывода

boost::asio::io\_context io\_context;

// Разрешение адреса и порта сервера

tcp::resolver resolver(io\_context);

tcp::resolver::results\_type endpoints = resolver.resolve("localhost", "12345");

// Создание сокета и подключение к серверу

tcp::socket socket(io\_context);

boost::asio::connect(socket, endpoints);

// Сообщение для отправки

std::string message = "Hello, server!";

boost::asio::write(socket, boost::asio::buffer(message));

// Чтение ответа от сервера

char reply[512];

size\_t reply\_length = boost::asio::read(socket, boost::asio::buffer(reply, message.length()));

std::cout << "Reply: " << std::string(reply, reply\_length) << std::endl;

} catch (std::exception& e) {

std::cerr << "Exception: " << e.what() << std::endl;

}

return 0;

}

Тестирование

Подключение и обработка данных

1

cpp

#include <cassert>

#include "database.h" // Заголовочный файл с функцией для регистрации

void test\_user\_registration() {

Database db;

db.connect("chat.db");

bool result = db.registerUser("test\_user", "password123");

assert(result == true); // Ожидаем, что регистрация успешна

}

2

cpp

void test\_client\_server\_communication() {

// Запускаем сервер в отдельном потоке

std::thread server\_thread(start\_server); // Функция для запуска сервера

std::this\_thread::sleep\_for(std::chrono::seconds(1)); // Ждем, пока сервер запустится

// Создаем клиент и отправляем сообщение

Client client;

client.connect("localhost", 12345);

std::string response = client.sendMessage("Hello, server!");

assert(response == "Hello, server!"); // Проверяем, что ответ сервера соответствует ожиданиям

server\_thread.join(); // Ожидаем завершения сервера

}

Проверка безопасности пароля

cpp

#include <bcrypt/BCrypt.hpp>

std::string hashPassword(const std::string& password) {

return BCrypt::generateHash(password);

}

Проверка корректности запросов

src/database.cpp

cpp

#include <sqlite3.h>

// Функция для регистрации пользователя

bool registerUser(const std::string& username, const std::string& password) {

sqlite3\* db;

sqlite3\_open("chat.db", &db);

std::string sql = "INSERT INTO Users (username, password) VALUES (?, ?);";

sqlite3\_stmt\* stmt;

sqlite3\_prepare\_v2(db, sql.c\_str(), -1, &stmt, nullptr);

sqlite3\_bind\_text(stmt, 1, username.c\_str(), -1, SQLITE\_STATIC);

sqlite3\_bind\_text(stmt, 2, hashPassword(password).c\_str(), -1, SQLITE\_STATIC);

int rc = sqlite3\_step(stmt);

sqlite3\_finalize(stmt);

sqlite3\_close(db);

return rc == SQLITE\_DONE;

}