МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**Лабораторная работа №4**

**«**Разработка сервиса статистики..

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-221», «АВТФ»  Стеклов Данил Евгеньевич  «3» декабря 2023г | Проверил:  Доцент ЗИ  Архипова Анастасия Борисовна  «\_\_\_» \_\_\_\_\_\_ 2023г |

Новосибирск 2023

**Цели и задачи работы:** Требуется реализовать сервис для сокращения ссылок. Сервис должен принимать запросы по http протоколу

**Задание к работе:**

Требуется разработать сервис агрегации статистики переходов

по сокращенным ссылкам. Сервис статистики должен собирать статистику с сервиса сокращения ссылок о переходах по сокращённым ссылкам и

генерировать отчеты по запросу пользователя. Сервис должен содержать 2 конечные точки:

1. Отправка статистики

URL: /

Метод: POST

Запрос содержит следующие данные:

◦ IP адрес, с которого выполнен переход;

◦ URL на который перенаправлен пользователь + сокращение;

◦ Временная отметка.

Ответ может не содержать полезной нагрузки

2. Построение отчета

URL: /report

Метод: POST

Запрос содержит набор детализаций в формате JSON:

{

"Dimensions": ["URL", "SourceIP", "TimeInterval"]

}

Ответ должен содержать список записей в формате JSON. Также требуется доработать сервис сокращения ссылок, а именно в момент перехода по сокращенной ссылки должны отправляться данные, описанные выше в сервис сбора статистики.

**Задание на языке C++:**

**Проект Сервера статистики ссылок:**

**Source.cpp**

#include <iostream>

#include <vector>

#include <boost/asio.hpp>

#include <nlohmann/json.hpp>

#define ADDRESS "192.168.1.4"

using namespace boost::asio;

class Server {

public:

Server(io\_service& io\_service) : acceptor\_(io\_service, ip::tcp::endpoint(ip::tcp::v4(), 2056)) { //После многоточий идет инициализация сложных членов класса, к которым будет применен свой конструктор

StartAccept(io\_service);

}

private:

void StartAccept(io\_service& io\_service) {

while (true) {

ip::tcp::socket socket(io\_service);

acceptor\_.accept(socket);

HandleClient(socket);

}

}

void HandleClient(ip::tcp::socket& socket) {

std::cout << "Client connect" << std::endl;

boost::system::error\_code error;

streambuf request\_buffer; // временное хранилище данных

std::cout << "Read data start" << std::endl;

read\_until(socket, request\_buffer, "**\n\n**", error); // чтение данных из сокета до "\n"

std::cout << "Read data successfuly" << std::endl;

if (error) {

std::cerr << error << std::endl;

socket.close();

}

std::istream request\_stream(&request\_buffer);

std::string request\_line;

getline(request\_stream, request\_line);

if (request\_line.find("POST /STATISTIC") != std::string::npos) { // если пришел POST запрос просто перенаправляем данные в БД

std::cout << "POST" << std::endl;

std::string url;

std::string ip;

std::string time;

int complete = 0; //счетчик считанных аргументов

while (true) {

getline(request\_stream, request\_line); // извлекаем каждую строку

if (request\_line.empty()) {

std::cerr << "Request empty";

break;

}

if (request\_line.find("url=") != std::string::npos) {

size\_t position = request\_line.find("url=");

if (position != std::string::npos) {

url = request\_line.substr(position + 4);

complete++;

}

}

else if (request\_line.find("ip=") != std::string::npos) {

size\_t position = request\_line.find("ip=");

if (position != std::string::npos) {

ip = request\_line.substr(position + 3);

complete++;

}

}

else if (request\_line.find("time=") != std::string::npos) {

size\_t position = request\_line.find("time=");

if (position != std::string::npos) {

time = request\_line.substr(position + 5);

complete++;

}

}

if (complete == 3) break; //все нужные аргументы получены

}

if (complete != 3) {

std::cerr << "Mistake arg";

socket.close();

return;

}

try { //отправка в БД

io\_service io\_service\_client;

ip::tcp::resolver resolver(io\_service\_client);

ip::tcp::socket socket\_client(io\_service\_client);

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query (ADDRESS, "6379")));

streambuf request;

std::ostream request\_stream(&request);

request\_stream << "POST /DATABASE/STATISTIC QTP/1.0**\n**";

request\_stream << "url=" << url << "**\n**";

request\_stream << "ip=" << ip << "**\n**";

request\_stream << "time=" << time << "**\n**";

request\_stream << "**\n**";

write(socket\_client, request);

socket\_client.close();

}

catch (std::exception& e) {

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

}

socket.close();

}

else if (request\_line.find("POST /REPORT") != std::string::npos) {

std::cout << "POST /REPORT" << std::endl;

nlohmann::json data;

std::string priority\_1;

std::string priority\_2;

std::string priority\_3;

while (true) {

getline(request\_stream, request\_line); // извлекаем каждую строку

if (request\_line.empty()) {

std::cerr << "Request empty";

break;

}

if (request\_line.find("parametrs=") != std::string::npos) {

int position\_ip = request\_line.find("ip");

int position\_time = request\_line.find("time");

int position\_url = request\_line.find("url");

if (position\_ip == std::string::npos || position\_time == std::string::npos || position\_url == std::string::npos) {

std::cerr << "Invalid request"; //TODO: Возврат клиенту отчета об ошибке

priority\_1 = "ip";

priority\_2 = "time"; //Пока что, чтобы не вводить каждый раз

priority\_3 = "url";

break;

}

if (position\_ip < position\_time && position\_ip < position\_url) {

priority\_1 = "ip";

if (position\_time < position\_url) priority\_2 = "time", priority\_3 = "url";

else priority\_2 = "url", priority\_3 = "time";

}

else if (position\_time < position\_ip && position\_time < position\_url) {

priority\_1 = "time";

if (position\_ip < position\_url) priority\_2 = "ip", priority\_3 = "url";

else priority\_2 = "url", priority\_3 = "ip";

}

else if (position\_url < position\_ip && position\_url < position\_time) {

priority\_1 = "url";

if (position\_time < position\_ip) priority\_2 = "time", priority\_3 = "ip";

else priority\_2 = "ip", priority\_3 = "time";

}

break;

}

}

try { //отправка в БД

io\_service io\_service\_client;

ip::tcp::resolver resolver(io\_service\_client);

ip::tcp::socket socket\_client(io\_service\_client);

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query (ADDRESS, "6379")));

streambuf request;

std::ostream request\_stream(&request);

request\_stream << "GET / QTP/1.0**\n**";

request\_stream << "**\n**";

write(socket\_client, request);

streambuf response\_buffer;

read\_until(socket\_client, response\_buffer, "**\n\n**", error);

if (error) {

std::cerr << error << std::endl;

socket\_client.close();

}

std::istream response\_stream(&response\_buffer);

std::string response\_line;

getline(response\_stream, response\_line);

if (response\_line.find("QTP /") != std::string::npos) {

std::cout << "QTP" << std::endl;

getline(response\_stream, response\_line);

try {

data = nlohmann::json::parse(response\_line); //Для дальнейшей обработки JSON данных

} catch (const std::exception& e) {

std::cerr << "Error parsing JSON: " << e.what() << std::endl;

}

}

socket\_client.close();

}

catch (std::exception& e) {

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

}

nlohmann::json reports = nlohmann::json::array(); //Главный массив объектов

int id = 1;

for (const auto& visit : data["visits"]) {

// Вывод информации о каждом посещении

std::cout << "IP: " << visit["ip"] << ", Time: " << visit["time"] << ", URL: " << visit["url"] << std::endl;

nlohmann::json report\_priority\_1, report\_priority\_2, report\_priority\_3; // Создание нового объекта "report" для каждого посещения

if (priority\_1 == "ip") {

if (reports.empty()) {//Первая добавление

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = 0;

report\_priority\_1["ip"] = visit["ip"];

report\_priority\_1["time"] = NULL;

report\_priority\_1["url"] = NULL;

report\_priority\_1["count"] = 1;

if (priority\_2 == "time") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentIp = 0;

for (auto& old\_report : reports) {

if (old\_report["ip"] == visit["ip"]) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentIp = old\_report["id"].get<int>();

break;

}

}

if (idParrentIp == 0) {

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = 0;

report\_priority\_1["ip"] = visit["ip"];

report\_priority\_1["time"] = NULL;

report\_priority\_1["url"] = NULL;

report\_priority\_1["count"] = 1;

if (priority\_2 == "time") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else { //Если ip существовал

if (priority\_2 == "time") {

int idParrentTime = 0;

for (auto& old\_report : reports) {

int pid = old\_report["pid"].get<int>();

if ((old\_report["time"] == visit["time"]) && (pid == idParrentIp)) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentTime = old\_report["id"].get<int>();

break;

}

}

if (idParrentTime == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentIp;

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentUrl = 0;

for (auto& old\_report : reports) {

if ((old\_report["url"] == visit["url"]) && (old\_report["pid"].get<int>() == idParrentTime)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"];

break;

}

}

if (idParrentUrl == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentTime;

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

else { //Если сначала юрл, а не время

int idParrentUrl = 0;

for (auto& old\_report : reports) {

if ((old\_report["url"] == visit["url"]) && (old\_report["pid"].get<int>() == idParrentIp)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"].get<int>();

break;

}

}

if (idParrentUrl == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentIp;

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentTime = 0;

for (auto& old\_report : reports) {

if ((old\_report["url"] == visit["url"]) && (old\_report["pid"].get<int>() == idParrentUrl)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentTime = old\_report["id"];

break;

}

}

if (idParrentTime == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentUrl;

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

}

}

}

else if (priority\_1 == "time") {

if (reports.empty()) {//Первая добавление

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = 0;

report\_priority\_1["ip"] = NULL;

report\_priority\_1["time"] = visit["time"];

report\_priority\_1["url"] = NULL;

report\_priority\_1["count"] = 1;

if (priority\_2 == "ip") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentTime = 0;

for (auto& old\_report : reports) {

if (old\_report["time"] == visit["time"]) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentTime = old\_report["id"].get<int>();

break;

}

}

if (idParrentTime == 0) {

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = 0;

report\_priority\_1["ip"] = NULL;

report\_priority\_1["time"] = visit["time"];

report\_priority\_1["url"] = NULL;

report\_priority\_1["count"] = 1;

if (priority\_2 == "ip") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else { //Если time существовал

if (priority\_2 == "ip") {

int idParrentIp = 0;

for (auto& old\_report : reports) {

int pid = old\_report["pid"].get<int>();

if ((old\_report["ip"] == visit["ip"]) && (pid == idParrentTime)) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentIp = old\_report["id"].get<int>();

break;

}

}

if (idParrentIp == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentTime;

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentUrl = 0;

for (auto& old\_report : reports) {

if ((old\_report["url"] == visit["url"]) && (old\_report["pid"].get<int>() == idParrentIp)) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"];

break;

}

}

if (idParrentUrl == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentIp;

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = visit["url"];

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

else { //Если сначала юрл, а не ip

int idParrentUrl = 0;

for (auto& old\_report : reports) {

if ((old\_report["url"] == visit["url"]) && (old\_report["pid"].get<int>() == idParrentTime)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"].get<int>();

break;

}

}

if (idParrentUrl == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentTime;

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = visit["url"];

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentIp = 0;

for (auto& old\_report : reports) {

if ((old\_report["ip"] == visit["ip"]) && (old\_report["pid"].get<int>() == idParrentUrl)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentIp = old\_report["id"];

break;

}

}

if (idParrentIp == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentUrl;

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

}

}

}

else if (priority\_1 == "url") {

if (reports.empty()) {

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = NULL;

report\_priority\_1["ip"] = NULL;

report\_priority\_1["time"] = NULL;

report\_priority\_1["url"] = visit["url"];

report\_priority\_1["count"] = 1;

if (priority\_2 == "ip") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentUrl = 0;

for (auto& old\_report : reports) {

if (old\_report["url"] == visit["url"]) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"].get<int>();

break;

}

}

if (idParrentUrl == 0) {

report\_priority\_1["id"] = id++;

report\_priority\_1["pid"] = 0;

report\_priority\_1["ip"] = NULL;

report\_priority\_1["time"] = NULL;

report\_priority\_1["url"] = visit["url"];

report\_priority\_1["count"] = 1;

if (priority\_2 == "ip") {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

else {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = report\_priority\_1["id"];

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

}

reports.push\_back(report\_priority\_1);

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else { //Если url существовал

if (priority\_2 == "ip") {

int idParrentIp = 0;

for (auto& old\_report : reports) {

int pid = old\_report["pid"].get<int>();

if ((old\_report["ip"] == visit["ip"]) && (pid == idParrentUrl)) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentIp = old\_report["id"].get<int>();

break;

}

}

if (idParrentIp == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentUrl;

report\_priority\_2["ip"] = visit["ip"];

report\_priority\_2["time"] = NULL;

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentTime = 0;

for (auto& old\_report : reports) {

if ((old\_report["time"] == visit["time"]) && (old\_report["pid"].get<int>() == idParrentIp)) {

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentTime = old\_report["id"];

break;

}

}

if (idParrentTime == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentIp;

report\_priority\_3["ip"] = NULL;

report\_priority\_3["time"] = visit["time"];

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

else { //Если сначала время, а не ip

int idParrentTime = 0;

for (auto& old\_report : reports) {

if ((old\_report["time"] == visit["time"]) && (old\_report["pid"].get<int>() == idParrentUrl)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentUrl = old\_report["id"].get<int>();

break;

}

}

if (idParrentTime == 0) {

report\_priority\_2["id"] = id++;

report\_priority\_2["pid"] = idParrentUrl;

report\_priority\_2["ip"] = NULL;

report\_priority\_2["time"] = visit["time"];

report\_priority\_2["url"] = NULL;

report\_priority\_2["count"] = 1;

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = report\_priority\_2["id"];

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_2);

reports.push\_back(report\_priority\_3);

}

else {

int idParrentIp = 0;

for (auto& old\_report : reports) {

if ((old\_report["ip"] == visit["ip"]) && (old\_report["pid"].get<int>() == idParrentTime)) { //Надо сделать не прямой равенство по времени, а интервальное в минуту, я хочу обрезать строку с секнудами и таким образом сравнивать минуты

old\_report["count"] = old\_report.value("count", 1) + 1;

idParrentIp = old\_report["id"];

break;

}

}

if (idParrentIp == 0) {

report\_priority\_3["id"] = id++;

report\_priority\_3["pid"] = idParrentTime;

report\_priority\_3["ip"] = visit["ip"];

report\_priority\_3["time"] = NULL;

report\_priority\_3["url"] = NULL;

report\_priority\_3["count"] = 1;

reports.push\_back(report\_priority\_3);

}

}

}

}

}

}

}

std::cout << reports.dump() << std::endl;

streambuf response;

std::ostream response\_stream (&response);

response\_stream << "HTTP/1.1 200 OK**\r\n**";

response\_stream << "Access-Control-Allow-Origin: \***\r\n**";

response\_stream << "Access-Control-Allow-Methods: GET, POST, PUT, DELETE**\r\n**";

response\_stream << "Access-Control-Allow-Headers: Content-Type, Authorization**\r\n**";

response\_stream << "Content-Type: text/plain**\r\n**";

response\_stream << "Content-Length: " << std::to\_string(reports.dump().length()) << "**\r\n**";

response\_stream << "**\r\n**";

response\_stream << reports.dump();

write(socket, response);

}

else {

std::cout << "SOMETHING = **\"**" << request\_line << "**\"**" << std::endl;

socket.close();

}

}

ip::tcp::acceptor acceptor\_;

};

int main() {

setlocale(LC\_ALL, "Ru");

try {

io\_service io\_service;

Server server(io\_service);

} catch (std::exception& e) {

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

}

return 0;

}

**Проект Сервера сокращателя ссылок:**

**Source.cpp**

#include <iostream>

#include <thread>

#include <ctime>

#include <boost/asio.hpp>

#include <boost/random.hpp>

#include <boost/date\_time/posix\_time/posix\_time.hpp>

#define ADDRESS "192.168.1.4" //поменять в случае смены локального адреса

using namespace boost::asio;

class Server {

public:

Server(io\_service& io\_service) : acceptor\_(io\_service, ip::tcp::endpoint(ip::tcp::v4(), 8080)) { //После многоточий идет инициализация членов класса, при вызове функции, в данном случае конструктора

StartAccept(io\_service);

}

private:

void StartAccept(io\_service& io\_service) {

while (true) {

ip::tcp::socket socket(io\_service);

acceptor\_.accept(socket);

std::thread(&Server::HandleClient, this, std::move(socket)).detach();

}

}

void HandleClient(ip::tcp::socket socket) {

std::cout << "Client connect" << std::endl;

boost::system::error\_code error;

streambuf request\_buffer; // временное хранилище данных

std::cout << "Read data start" << std::endl;

read\_until(socket, request\_buffer, "**\r\n\r\n**", error); // чтение данных из сокета до "\r\n\r\n"

std::cout << "Read data successfuly" << std::endl;

if (error) {

std::cerr << error << std::endl;

socket.close();

}

std::istream request\_stream(&request\_buffer);

std::string request\_line;

getline(request\_stream, request\_line);

if (request\_line.find("POST /SHORTEN") != std::string::npos) { // если пришел POST запрос

std::cout << "POST" << std::endl;

std::string encrypted\_long\_url;

while (true) {

getline(request\_stream, request\_line); // извлекаем каждую строку

if (request\_line.empty()) {

std::cerr << "Request empty";

break;

}

if (request\_line.find("url=") != std::string::npos) { //Если что то нашел(вернул позицию)

encrypted\_long\_url = request\_line.substr(request\_line.find("url=") + 4); // +4, чтобы пропустить "URL:"

break;

}

}

std::string short\_url;

try { //отправка в БД

io\_service io\_service\_client;

ip::tcp::resolver resolver(io\_service\_client);

ip::tcp::socket socket\_client(io\_service\_client);

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query (ADDRESS, "6379")));

streambuf request;

std::ostream request\_stream(&request);

request\_stream << "POST /DATABASE/REF QTP/1.0**\n**";

request\_stream << "command=GetShortURL**\n**";

request\_stream << "arg=" << encrypted\_long\_url;

request\_stream << "**\n\n**";

write(socket\_client, request);

streambuf response\_buffer;

read\_until(socket\_client, response\_buffer, "**\n\n**", error);

if (error) {

std::cerr << error << std::endl;

socket\_client.close();

}

std::istream response\_stream(&response\_buffer);

std::string response\_line;

getline(response\_stream, response\_line);

if (response\_line.find("QTP /") != std::string::npos) {

std::cout << "QTP" << std::endl;

while (true) {

getline(response\_stream, response\_line);

if (response\_line.find("url=") != std::string::npos) {

short\_url = response\_line.substr(response\_line.find("url=") + 4);

break;

}

}

}

socket\_client.close();

}

catch (std::exception& e) {

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

}

std::cout << short\_url << std::endl;

WriteResponse(short\_url, socket, 200);

std::cout << "Response send" << std::endl;

socket.close();

}

else if (request\_line.find("GET /") != std::string::npos) {

std::cout << "GET" << std::endl;

int start = request\_line.find("GET /") + 5;

int end = request\_line.find(' ', start);

std::string encrypted\_short\_url = "http://localhost:8080/" + request\_line.substr(start, end - start);

std::string long\_url;

try { //отправка в БД

io\_service io\_service\_client;

ip::tcp::resolver resolver(io\_service\_client);

ip::tcp::socket socket\_client(io\_service\_client);

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query (ADDRESS, "6379")));

streambuf request;

std::ostream request\_stream(&request);

request\_stream << "POST /DATABASE/REF QTP/1.0**\n**";

request\_stream << "command=GetLongURL**\n**";

request\_stream << "arg=" << encrypted\_short\_url;

request\_stream << "**\n\n**";

write(socket\_client, request);

streambuf response\_buffer;

read\_until(socket\_client, response\_buffer, "**\n\n**", error);

if (error) {

std::cerr << error << std::endl;

socket\_client.close();

}

std::istream response\_stream(&response\_buffer);

std::string response\_line;

getline(response\_stream, response\_line);

if (response\_line.find("QTP /") != std::string::npos) {

std::cout << "QTP" << std::endl;

while (true) {

getline(response\_stream, response\_line);

if (response\_line.find("url=") != std::string::npos) {

long\_url = response\_line.substr(response\_line.find("url=") + 4);

break;

}

}

}

socket\_client.close();

}

catch (std::exception& e) {

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

}

if (long\_url != "No Found") {

try { //отправка статистики

io\_service io\_service\_client;

ip::tcp::resolver resolver(io\_service\_client);

ip::tcp::socket socket\_client(io\_service\_client);

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query (ADDRESS, "2056")));

streambuf request;

boost::posix\_time::time\_facet\* facet = new boost::posix\_time::time\_facet("%Y-%m-%d %H:%M");

std::stringstream ss;

ss.imbue(std::locale(std::locale(), facet));

ss << boost::posix\_time::second\_clock::local\_time();

std::string time\_string = ss.str();

std::ostream request\_stream(&request);

request\_stream << "POST /STATISTIC QTP/1.0**\n**";

request\_stream << "url=" << long\_url <<" (" << encrypted\_short\_url << ")**\n**";

request\_stream << "ip=" << socket.remote\_endpoint().address().to\_string() << "**\n**";

request\_stream << "time=" << time\_string << "**\n**";

request\_stream << "**\n**";

write(socket\_client, request);

socket\_client.close();

}

catch (std::exception& e) {

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

}

WriteResponse(long\_url, socket, 302);

socket.close();

}

else {

WriteResponse(long\_url, socket, 404);

socket.close();

}

}

else {

std::cout << "SOMETHING = **\"**" << request\_line << "**\"**" << std::endl;

socket.close();

}

}

void WriteResponse(std::string& url, ip::tcp::socket& socket, int code) {

streambuf response;

std::ostream response\_stream (&response);

if (code == 200) {

response\_stream << "HTTP/1.1 200 OK**\r\n**";

response\_stream << "Access-Control-Allow-Origin: \***\r\n**";

response\_stream << "Access-Control-Allow-Methods: GET, POST, PUT, DELETE**\r\n**";

response\_stream << "Access-Control-Allow-Headers: Content-Type, Authorization**\r\n**";

response\_stream << "Content-Type: text/html**\r\n**";

response\_stream << "Content-Length: " << url.size() << "**\r\n**";

response\_stream << "**\r\n**";

response\_stream << url;

}

else if (code == 302) {

response\_stream << "HTTP/1.1 302 Found**\r\n**";

response\_stream << "Access-Control-Allow-Origin: \***\r\n**";

response\_stream << "Access-Control-Allow-Methods: GET, POST, PUT, DELETE**\r\n**";

response\_stream << "Access-Control-Allow-Headers: Content-Type, Authorization**\r\n**";

response\_stream << "Content-Type: text/html**\r\n**";

response\_stream << "Location: " << url << "**\r\n**";

response\_stream << "**\r\n**";

}

else if (code == 404) {

response\_stream << "HTTP/1.1 404 Not Found**\r\n**";

response\_stream << "Access-Control-Allow-Origin: \***\r\n**";

response\_stream << "Access-Control-Allow-Methods: GET, POST, PUT, DELETE**\r\n**";

response\_stream << "Access-Control-Allow-Headers: Content-Type, Authorization**\r\n**";

response\_stream << "Content-Type: text/html**\r\n**";

response\_stream << "Content-Length: 9**\r\n**";

response\_stream << "**\r\n**";

response\_stream << "Not Found";

}

write(socket, response);

}

ip::tcp::acceptor acceptor\_;

};

int main() {

setlocale(LC\_ALL, "Ru");

try {

io\_service io\_service;

Server server(io\_service);

} catch (std::exception& e) {

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

}

return 0;

}

**Проект Базы Данных:**

**Source.cpp**

#include <iostream>

#include <fstream>

#include <ctime>

#include <vector>

#include <thread>

#include <csignal>

#include <boost/asio.hpp>

#include <boost/random.hpp>

#include <nlohmann/json.hpp>

using namespace boost::asio;

class Visit {

public:

Visit(const std::string& url, const std::string& ip, const std::string& time) { //конструктор

url\_ = url;

ip\_ = ip;

time\_ = time;

}

std::string GetURL() {

return url\_;

}

std::string GetIP() {

return ip\_;

}

std::string GetTime() {

return time\_;

}

private:

std::string url\_;

std::string ip\_;

std::string time\_;

};

class HashTable { //Хеш-таблица с методом открытых адресаций без фукнции удаления для ссылок

public:

HashTable() { //конструктор

for (int i = 0; i < 255; ++i) {

keys[i] = "";

values[i] = "";

}

}

std::string GetShortURL(const std::string& value) {

for (int i = 0; i < 255; ++i) {

if (values[i] == value) return keys[i];

}

std::string new\_key = RandomURL();

SetURL(new\_key, value);

return new\_key;

}

std::string GetLongURL(const std::string& key) {

if (values[FuncHash(key)] != "") return values[FuncHash(key)];

return "No Found";

}

private:

void SetURL(const std::string& key, const std::string& value) {

int index = FuncHash(key);

bool is\_coincidence = false;

while (values[index] != "") {

if (keys[index] == key) { //если ключ совпадает с нашим

is\_coincidence = true;

break;

}

index = (index + FuncHashSec(key[0])) % 255; //если ячейка не подошла переходим к следующей

}

keys[index] = key;

values[index] = value;

}

std::string RandomURL() {

const std::string alphabet = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890";

boost::minstd\_rand generator(static\_cast<unsigned int>(std::time(0)));

boost::random::uniform\_int\_distribution<int> rand\_alph(0, alphabet.length() - 1);

boost::random::uniform\_int\_distribution<int> rand\_len(3, 5);

boost::random::variate\_generator<boost::minstd\_rand&, boost::random::uniform\_int\_distribution<int>> alph\_gen(generator, rand\_alph);

boost::random::variate\_generator<boost::minstd\_rand&, boost::random::uniform\_int\_distribution<int>> len\_gen(generator, rand\_len);

std::string url = "http://localhost:8080/";

int len\_url = len\_gen();

for (int i = 0; i < len\_url; ++i) {

url += alphabet[alph\_gen()];

}

return url;

}

int FuncHash(std::string key) {

int sum = 0;

for (int i = 0; key[i]; ++i)

sum += key[i];

return sum % 100;

}

int FuncHashSec(char ch) {

int sum = ch \* 37;

return sum % 100;

}

std::string keys[255];

std::string values[255];

};

class Server {

public:

Server(io\_service& io\_service) : acceptor\_(io\_service, ip::tcp::endpoint(ip::tcp::v4(), 6379)), data\_base() { //После многоточий идет инициализация сложных членов класса, к которым будет применен свой конструктор

StartAccept(io\_service);

}

private:

void StartAccept(io\_service& io\_service) {

while (true) {

ip::tcp::socket socket(io\_service);

acceptor\_.accept(socket);

std::thread(&Server::HandleClient, this, std::move(socket)).detach();

}

}

void HandleClient(ip::tcp::socket socket) {

std::cout << "Client connect" << std::endl;

boost::system::error\_code error;

streambuf request\_buffer; // временное хранилище данных

std::cout << "Read data start" << std::endl;

read\_until(socket, request\_buffer, "**\n\n**", error); // чтение данных из сокета до "\n\n"

std::cout << "Read data successfuly" << std::endl;

if (error) {

std::cerr << error << std::endl;

socket.close();

}

std::istream request\_stream(&request\_buffer);

std::string request\_line;

getline(request\_stream, request\_line);

if (request\_line.find("POST /DATABASE/REF") != std::string::npos) { // если пришел POST запрос

std::cout << "POST /DATABASE/REF" << std::endl;

std::string command;

std::string arg;

int complete = 0; //счетчик считанных аргументов

while (true) {

getline(request\_stream, request\_line); // извлекаем каждую строку

if (request\_line.empty()) {

std::cerr << "Request empty";

break;

}

if (request\_line.find("command=") != std::string::npos) {

size\_t position = request\_line.find("command=");

if (position != std::string::npos) {

command = request\_line.substr(position + 8);

complete++;

}

}

else if (request\_line.find("arg=") != std::string::npos) {

size\_t position = request\_line.find("arg=");

if (position != std::string::npos) {

arg = request\_line.substr(position + 4);

complete++;

}

}

if (complete == 2) break; //оба нужных аргумента получено

}

if (complete != 2) {

std::cerr << "Mistake arg";

socket.close();

return;

}

std::string response; //Ответ клиенту

if (command == "GetShortURL") {

response = data\_base.GetShortURL(Decrypt(arg));

}

else if(command == "GetLongURL") {

response = data\_base.GetLongURL(arg);

}

std::cout << "command = " << command << " arg = " << arg << " response = " << response << std::endl; //log

WriteUrlResponse(response, socket);

std::cout << "Response send" << std::endl;

socket.close();

}

else if (request\_line.find("POST /DATABASE/STATISTIC") != std::string::npos) {

std::cout << "POST /DATABASE/STATISTIC" << std::endl;

std::string url;

std::string ip;

std::string time;

int complete = 0; //счетчик считанных аргументов

while (true) {

getline(request\_stream, request\_line); // извлекаем каждую строку

if (request\_line.empty()) {

std::cerr << "Request empty";

break;

}

if (request\_line.find("url=") != std::string::npos) {

size\_t position = request\_line.find("url=");

if (position != std::string::npos) {

url = request\_line.substr(position + 4);

complete++;

}

}

else if (request\_line.find("ip=") != std::string::npos) {

size\_t position = request\_line.find("ip=");

if (position != std::string::npos) {

ip = request\_line.substr(position + 3);

complete++;

}

}

else if (request\_line.find("time=") != std::string::npos) {

size\_t position = request\_line.find("time=");

if (position != std::string::npos) {

time = request\_line.substr(position + 5);

complete++;

}

}

if (complete == 3) break; //все нужные аргументы получены

}

if (complete != 3) {

std::cerr << "Mistake arg";

socket.close();

return;

}

visits.emplace\_back(url, ip, time); //записываем в вектор объект типа Visit используя конструктор внутри уфнкции emplace\_back

using json = nlohmann::json;

// 1. Загрузим существующие посещения из файла

json jsonOutput;

std::ifstream inFile("visits.json");

inFile.seekg(0, std::ios::end);

if (inFile.tellg() != 0) {

// Файл не пустой, загружаем содержимое в переменную jsonOutput

inFile.seekg(0, std::ios::beg);

std::string fileContent((std::istreambuf\_iterator<char>(inFile)), std::istreambuf\_iterator<char>());

jsonOutput = json::parse(fileContent);

} else {

// Файл пуст, инициализируем пустым объектом

jsonOutput["visits"] = json::array();

}

inFile.close();

// 2. Добавим новое посещение

json jsonVisit; // Создаем JSON объект для каждого посещения.

jsonVisit["url"] = url; // Добавляем поле "url".

jsonVisit["time"] = time; // Добавляем поле "time".

jsonVisit["ip"] = ip; // Добавляем поле "ip".

jsonOutput["visits"].push\_back(jsonVisit); // Добавляем JSON объект для посещения в массив.

// 3. Запишем обновленные данные обратно в файл

std::ofstream outFile("visits.json");

outFile << jsonOutput.dump(4) << std::endl;

outFile.close();

socket.close();

}

else if (request\_line.find("GET /") != std::string::npos) {

std::cout << "GET /" << std::endl;

nlohmann::json data;

std::ifstream input\_file("visits.json");

// Проверка, удалось ли открыть файл

if (input\_file.is\_open()) {

// Чтение данных из файла в строку

std::string file\_contents((std::istreambuf\_iterator<char>(input\_file)), std::istreambuf\_iterator<char>());

// Десериализация JSON

try {

data = nlohmann::json::parse(file\_contents);

} catch (const std::exception& e) {

std::cerr << "Ошибка при десериализации JSON: " << e.what() << std::endl;

}

// Закрытие файла

input\_file.close();

} else {

std::cerr << "Не удалось открыть файл" << std::endl;

}

WriteJsonResponse(data, socket);

std::cout << "Response send" << std::endl;

socket.close();

}

else {

std::cout << "SOMETHING = **\"**" << request\_line << "**\"**" << std::endl;

socket.close();

}

}

std::string Decrypt(std::string& encrypted\_long\_url) {

std::string long\_url;

for (int i = 0; i < encrypted\_long\_url.size(); ++i) {

if (encrypted\_long\_url[i] == '%' && i + 2 < encrypted\_long\_url.size()) {

int first\_symbol = encrypted\_long\_url[i + 1] < 58 ? encrypted\_long\_url[i + 1] - '0' : encrypted\_long\_url[i + 1] - 'A' + 10;

int second\_symbol = encrypted\_long\_url[i + 2] < 58 ? encrypted\_long\_url[i + 2] - '0' : encrypted\_long\_url[i + 2] - 'A' + 10;

char symbol = (first\_symbol << 4) | second\_symbol;

long\_url.push\_back(symbol);

i += 2;

}

else if (encrypted\_long\_url[i] == '+') {

long\_url.push\_back(' ');

}

else {

long\_url.push\_back(encrypted\_long\_url[i]);

}

}

return long\_url;

}

void WriteUrlResponse(std::string& url, ip::tcp::socket& socket) {

streambuf response;

std::ostream response\_stream (&response);

response\_stream << "QTP /1.0**\n**";

response\_stream << "url=" << url;

response\_stream << "**\n\n**";

write(socket, response);

}

void WriteJsonResponse(nlohmann::json& json, ip::tcp::socket& socket) {

streambuf response;

std::ostream response\_stream (&response);

response\_stream << "QTP /1.0**\n**";

response\_stream << json.dump();

response\_stream << "**\n\n\n**";

write(socket, response);

}

ip::tcp::acceptor acceptor\_;

HashTable data\_base;

std::vector<Visit> visits;

};

int main() {

try {

io\_service io\_service;

Server server(io\_service);

} catch (std::exception& e) {

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

}

return 0;

}

**Проект сайта для сокращения ссылок:**

**1.home.html**

<!doctype html>

<html lang="en">

    <head>

        <meta charset="utf-8"/>

        <link href="main\_home.css" rel="stylesheet"/>

        <meta name="description" content="Простой способ сокращать ссылки и делиться ими"/>

        <title>Сокращение ссылок</title>

    </head>

    <body>

        <div class="screen\_\_main">

            <div class="screen\_\_body">

                <div class="main-screen">

                    <div class="main-screen\_\_container">

                        <div class="app-logo">

                            <svg width="48" fill="none" height="48" viewBox="0 0 48 48" xmlns="http://www.w3.org/2000/svg">

                                <g clip-path="url(#clip0\_953\_9187)">

                                    <mask id="mask0\_953\_9187" style="mask-type:alpha" maskUnits="userSpaceOnUse" x="0" y="0" width="48" height="48">

                                        <circle cx="24" cy="24" r="24" fill="white"></circle>

                                    </mask>

                                    <g mask="url(#mask0\_953\_9187)">

                                        <rect width="48" height="48" fill="#FC3F1D"></rect>

                                        <path d="M27.5389 13.4409H25.11C20.9461 13.4409 18.8642 15.5228 18.8642 18.6457C18.8642 22.1156 20.2522 23.8505 23.375 25.9325L25.804 27.6674L18.8642 38.424H13.3124L19.9052 28.7084C16.0883 25.9325 14.0064 23.5035 14.0064 18.9927C14.0064 13.4409 17.8232 9.62402 25.11 9.62402H32.3967V38.424H27.5389V13.4409Z" fill="white"></path>

                                    </g>

                                </g>

                                <defs>

                                    <clipPath id="clip0\_953\_9187">

                                        <rect width="48" height="48" fill="white"></rect>

                                    </clipPath>

                                </defs>

                            </svg>

                            <h1 class="app-logo\_\_text">Кликер</h1>

                        </div>

                        <span class="main-screen\_\_text">Помогите клиентам быстро найти вашу страницу в интернете. Благодаря короткой ссылке клиентам не придётся видеть длинные url-адреса, занимающие много места.</span>

                        <form id="myForm" class="main-screen\_\_shortener-container">

                            <div class="shortener\_\_url-input-container">

                                <div class="text-input shortener\_\_url-input text-input\_view\_contrast text-input\_size\_s56">

                                    <span class="text-input\_\_content">

                                        <div class="text-input\_\_control-container">

                                            <input id="urlField" class="text-input\_\_control" placeholder="Введите ссылку, которую нужно сократить" type="text" value="" name="url"/>

                                        </div>

                                        <span class="text-input\_\_box"></span>

                                    </span>

                                </div>

                                <button type="submit" class="button button\_width\_auto button\_size\_s56 button\_view\_primary button\_mode\_text">

                                    <div class="button\_\_shape">

                                        <div class="button\_\_content">

                                            <span title="Сократить" class="button\_\_text">Сократить</span>

                                        </div>

                                    </div>

                                </button>

                            </div>

                        </form>

                        <div class="response\_\_container" style="display: none;">

                            <span id="result">

                                <a href=""></a>

                            </span>

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

        <script>

            $(document).ready(function() {

                $('#myForm').submit(function(e) {

                    e.preventDefault();

                    $.ajax({

                        type: "POST",

                        url: 'http://localhost:8080/SHORTEN',

                        data: $(this).serialize(),

                        success: function(data) {

                            var ss = $('<a>');

                            ss.attr('href', data); //Установите URL для ссылки на основе полученных данных

                            ss.text(data); //Текст ссылки

                            $('.response\_\_container #result').empty().append(ss);

                            $('.response\_\_container').show();

                        },

                        error: function(jqXHR, textStatus, errorThrown) {

                            console.log("Ошибка", textStatus, "а ты кто такой",errorThrown);

                        }

                    });

                });

            });

            </script>

    </body>

</html>

**2. main.css**

body {

    background-color: #232529;

}

.screen\_\_main {

    margin-top: 68px;

    display:flex

}

.screen\_\_body {

    min-width:0;

    padding-right:24px;

    padding-left:24px;

    position:relative;width:100%

}

.button {

  background-color: transparent;

  border: none;

  cursor: pointer;

  display: block;

  outline: none;

  padding: 0;

  position: relative;

}

.button\_\_shape {

  border-radius: 14px;

  padding: 14px;

  color: #3F444C;

  background-color: #b0bdd633;

  box-shadow: none;

  transform: scale(1);

}

.button:hover .button\_\_shape {

  color: #ffffff;

  background-color: #d5aa00;

}

.button\_\_text {

  font-size: 16px;

  font-weight: 400;

  line-height: 28px;

  overflow: hidden;

  padding-bottom: 2px;

  text-align: center;

  text-overflow: ellipsis;

  white-space: nowrap;

}

.main-screen {

  align-content: center;

  display: flex;

  height: 100%;

  justify-content: center;

}

.main-screen\_\_container {

  align-items: center;

  display: flex;

  flex-direction: column;

  width: 700px;

}

.main-screen\_\_text {

  color: #1a2b4d99;

  font-size: 16px;

  line-height: 24px;

  margin-top: 16px;

}

.main-screen\_\_shortener-container {

  margin-top: 16px;

  width: 100%;

}

.app-logo {

  align-items: center;

  display: flex;

  width: 100%;

}

.app-logo\_\_text {

  color: #2f3747;

  font-size: 32px;

  font-weight: 500;

  line-height: 40px;

  margin: 0 0 0 12px;

}

.shortener\_\_url-input-container {

  align-items: flex-start;

  display: flex;

  width: 100%;

}

.shortener\_\_url-input {

  margin-right: 12px;

  width: 100%;

}

.text-input\_\_content {

  box-sizing: border-box;

  display: flex;

  position: relative;

  vertical-align: top;

  width: 100%;

  z-index: 0;

  margin-right: 12px;

}

.text-input\_\_box {

  bottom: 0;

  box-sizing: border-box;

  height: auto;

  left: 0;

  position: absolute;

  right: 0;

  top: 0;

  transition: background-color 0.15s linear, border-color 0.15s linear;

  z-index: 1;

}

.text-input\_\_control-container {

  box-sizing: border-box;

  display: flex;

  padding: 16px;

  width: 100%;

}

.text-input\_\_control {

  background: none;

  border: 0 solid transparent;

  box-sizing: border-box;

  color: #2f3747;

  font-size: 16px;

  line-height: 24px;

  margin: 0;

  outline: 0;

  padding: 0 20px;

  position: relative;

  vertical-align: top;

  width: 100%;

  z-index: 2;

}

.text-input\_\_control::placeholder {

  color: #475a8080;

  opacity: 1;

}

.text-input\_view\_contrast .text-input\_\_box,

.text-input\_view\_default .text-input\_\_box {

  border-radius: 14px;

}

.text-input\_\_box {

  border-bottom: 1px solid #a5b1ca4d;

}

.text-input\_\_box:after {

  border-bottom: 2px solid #ffd21f;

  bottom: -1px;

  content: "";

  left: 0;

  pointer-events: none;

  position: absolute;

  right: 0;

  transform: scaleX(0);

  transition: transform 0.1s cubic-bezier(0, 0, 0.2, 1) 0ms;

}

.text-input\_view\_contrast .text-input\_\_box {

  background-color: #fff;

}

.response\_\_container {

  align-items: center;

  background-color: #fff;

  border-radius: 24px;

  display: flex;

  justify-content: center;

  margin-top: 16px;

  width: 700px;

  height: 120px;

  overflow-y: hidden;

}

#result {

  color: #2f3747;

  font-size: 20px;

  line-height: 32px;

}

a:hover {

  color: #d00;

}

a {

  color: #C3CCDE;

  text-decoration: none;

}

@font-face {

  font-display: optional;

  font-family: 'YS Text';

  font-style: normal;

  font-weight: 300;

  src: url(//yastatic.net/islands/\_/kxV2-EeUdyizF\_lxQ-hrmltgp3c.woff2) format('woff2'),

       url(//yastatic.net/islands/\_/p9QGkWz-vqtayeFDeI6z9Dxffpo.woff) format('woff');

}

@font-face {

  font-display: optional;

  font-family: 'YS Text';

  font-style: normal;

  font-weight: 400;

  src: url(//yastatic.net/islands/\_/PyVcRbwHetz0gOVWLonWH7Od8zM.woff2) format('woff2'),

       url(//yastatic.net/islands/\_/bIx8jOfCEfR-mECoDUEZywDBuHA.woff) format('woff');

}

@font-face {

  font-display: optional;

  font-family: 'YS Text';

  font-style: normal;

  font-weight: 500;

  src: url(//yastatic.net/islands/\_/7\_GKBdKFbUPzKlghJRv55xgz0FQ.woff2) format('woff2'),

       url(//yastatic.net/islands/\_/SmqPmIMEXrW4lOY8QrhTUVDbrro.woff) format('woff');

}

@font-face {

  font-display: optional;

  font-family: 'YS Text';

  font-style: normal;

  font-weight: 700;

  src: url(//yastatic.net/islands/\_/6Roy0LCd05cK4nNCipgzheYcNVU.woff2) format('woff2'),

       url(//yastatic.net/islands/\_/KtHQR1erf3spayoIM4M4ngg0e2E.woff) format('woff');

}

body, button, input, textarea {

  font-family: 'YS Text', arial, sans-serif;

}

**Проект сайта для получения статистики по ссылкам:**

**1.stat.html**

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href="main\_stat.css" rel="stylesheet"/>

    <title>Правка приоритов</title>

</head>

<body>

    <div id="parrent">

        <form id="myForm"  class="main-screen\_\_shortener-container">

            <div class="shortener\_\_url-input-container">

                <div class="text-input shortener\_\_url-input text-input\_view\_contrast text-input\_size\_s56">

                    <span class="text-input\_\_content">

                        <div class="text-input\_\_control-container">

                            <input id="urlField" class="text-input\_\_control" placeholder="Выберите приоритет(ip,url,time)" type="text" value="" name="url"/>

                        </div>

                        <span class="text-input\_\_box"></span>

                    </span>

                </div>

                <button type="submit" class="button button\_width\_auto button\_size\_s56 button\_view\_primary button\_mode\_text">

                    <div class="button\_\_shape">

                        <div class="button\_\_content">

                            <span title="Сократить" class="button\_\_text">Отправить запрос</span>

                        </div>

                    </div>

                </button>

            </div>

        </form>

        <div id="jsonResponseContainer"></div>

    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

        <script>

            $(document).ready(function() {

                $('#myForm').submit(function(e) {

                    e.preventDefault();

                    var formData = $(this).serializeArray();

                    var data = 'parametrs=' + formData[0].value + "\n\n";

                    $.ajax({

                        type: "POST",

                        url: 'http://localhost:2056/REPORT',

                        processData: false,

                        contentType: 'text/plain',

                        data: data,

                        success: function(response) {

                            // Отображение JSON-ответа на странице с помощью Highlight.js

                            var jsonObject = JSON.parse(response);

                            var formattedJson = JSON.stringify(jsonObject, null, 2); // Преобразование JSON-ответа в строку с отступами

                            $('#jsonResponseContainer').html('<pre>' + formattedJson + '</pre>'); // Вставка отформатированного JSON-ответа на страницу

                        }

                    });

                });

            });

        </script>

</body>

</html>

**2.main.stat.css**

body {

    background: #232529;

    font-family: Arial, sans-serif;

    display: flex;

    justify-content: center;

    align-items: center;

    min-height: 100vh;

    margin: 0;

}

#parrent {

    width: 50%;

    padding: 20px;

    border-radius: 10px;

    display: flex;

    flex-direction: column;

}

form {

    background-color: rgba(135, 150, 0, 0.473);

    width: 100%;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    display: flex;

    flex-direction: column;

    align-items: center;

}

.priority-input {

    margin-bottom: 10px;

    display: flex;

    flex-direction: column;

    align-items: center;

}

.priority-input label {

    margin-bottom: 10px;

}

.button {

    background-color: transparent;

    border: none;

    cursor: pointer;

    display: block;

    outline: none;

    padding: 0;

    position: relative;

  }

.button\_\_shape {

    border-radius: 14px;

    padding: 14px;

    background-color: #ffffff;

    color: #3F444C;

    box-shadow: none;

    transform: scale(1);

}

.button:hover .button\_\_shape {

    color: #ffffff;

    background-color: #d5aa00;

}

.button\_\_text {

    font-size: 16px;

    font-weight: 400;

    line-height: 28px;

    overflow: hidden;

    padding-bottom: 2px;

    text-align: center;

    text-overflow: ellipsis;

    white-space: nowrap;

}

.shortener\_\_url-input-container {

    align-items: flex-start;

    display: flex;

    width: 100%;

}

.shortener\_\_url-input {

    margin-right: 12px;

    width: 100%;

}

.text-input\_\_content {

    box-sizing: border-box;

    display: flex;

    position: relative;

    vertical-align: top;

    width: 100%;

    z-index: 0;

    margin-right: 12px;

}

.text-input\_\_box {

    bottom: 0;

    box-sizing: border-box;

    height: auto;

    left: 0;

    position: absolute;

    right: 0;

    top: 0;

    transition: background-color 0.15s linear, border-color 0.15s linear;

    z-index: 1;

}

.text-input\_\_control-container {

    box-sizing: border-box;

    display: flex;

    padding: 16px;

    width: 100%;

}

.text-input\_\_control {

    background: none;

    border: 0 solid transparent;

    box-sizing: border-box;

    color: #2f3747;

    font-size: 16px;

    line-height: 24px;

    margin: 0;

    outline: 0;

    padding: 0 20px;

    position: relative;

    vertical-align: top;

    width: 100%;

    z-index: 2;

}

.text-input\_\_control::placeholder {

    color: #475a8080;

    opacity: 1;

}

.text-input\_view\_contrast .text-input\_\_box,

.text-input\_view\_default .text-input\_\_box {

    border-radius: 14px;

}

.text-input\_\_box {

    border-bottom: 1px solid #a5b1ca4d;

}

.text-input\_\_box:after {

    border-bottom: 2px solid #ffd21f;

    bottom: -1px;

    content: "";

    left: 0;

    pointer-events: none;

    position: absolute;

    right: 0;

    transform: scaleX(0);

    transition: transform 0.1s cubic-bezier(0, 0, 0.2, 1) 0ms;

}

.text-input\_view\_contrast .text-input\_\_box {

    background-color: #fff;

}

#jsonResponseContainer {

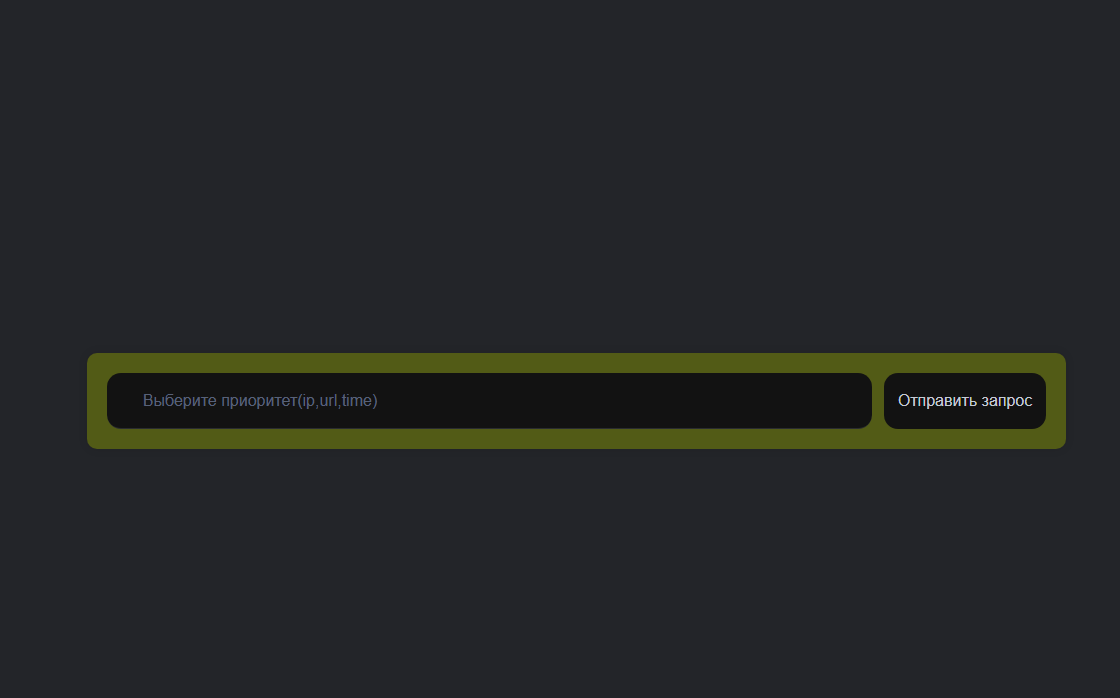
    font-size: 20px;

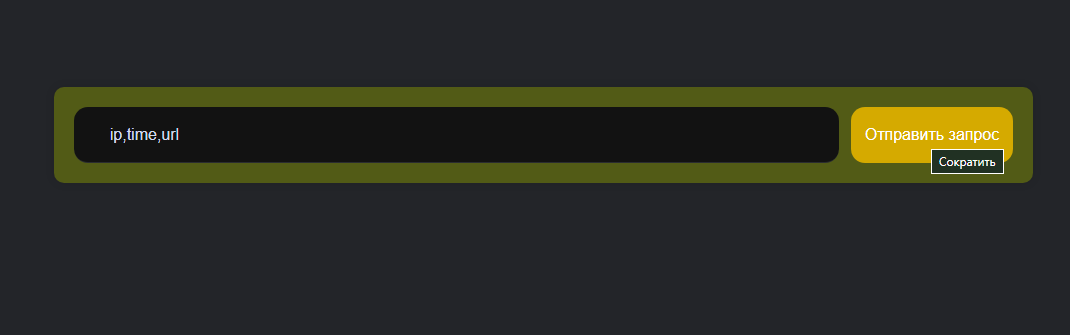
}

**Пример:**

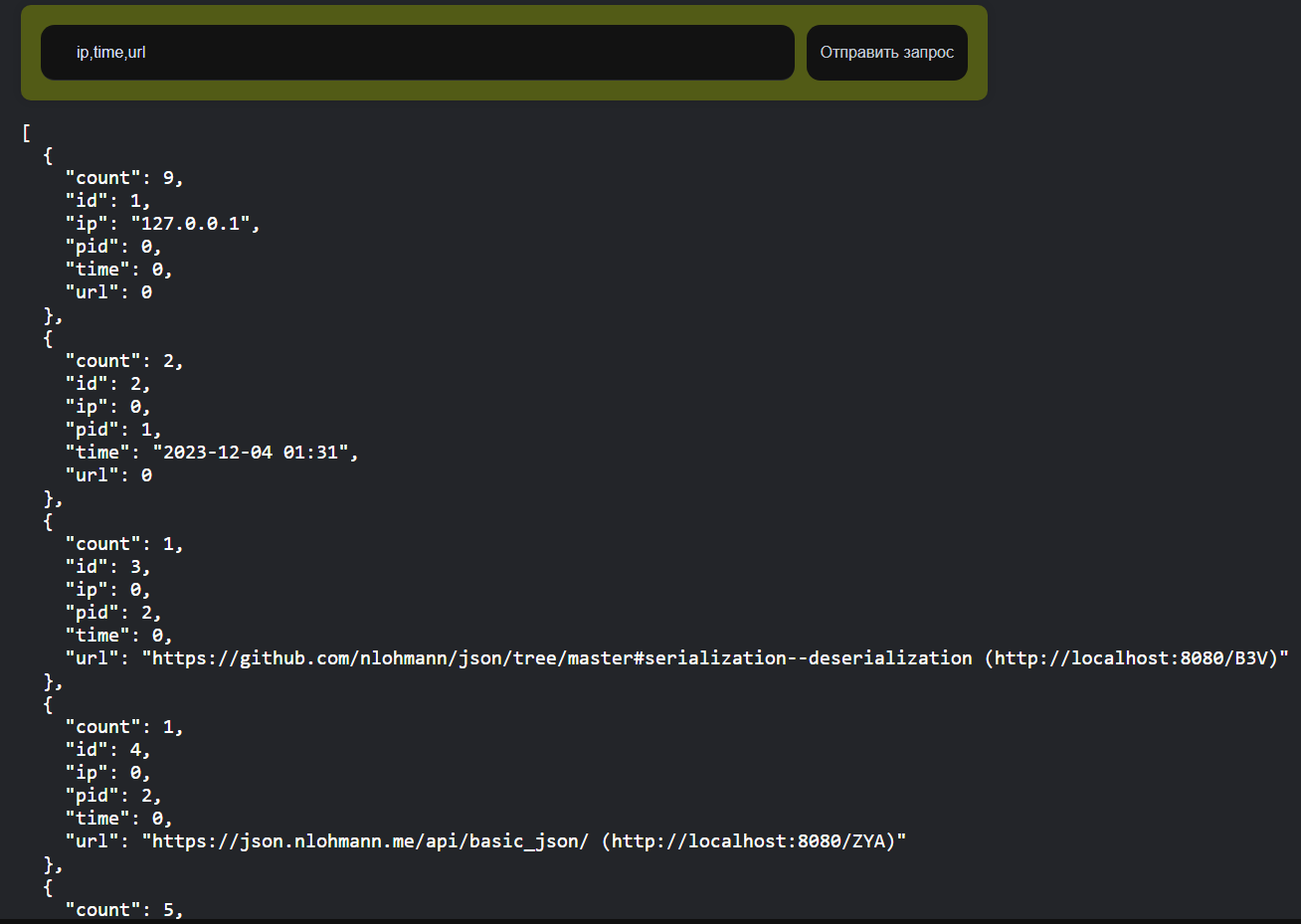
В процессе переходов по ссылка был сформирован файл visits.json

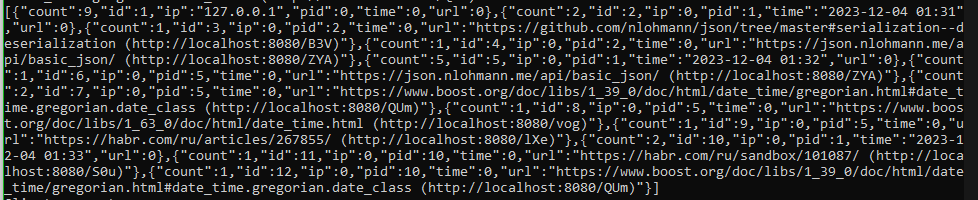


Далее заходим на сайт по получению статистики 

Вводим наш запрос, например ip,time,url 

Получаем ответ



Json объект в консоли 

Введем для пример другой запрос на детализацию, например time,url,ip

