МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**Лабораторная работа**

**«**Разработка сервиса сокращения ссылок.

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

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-221», «АВТФ»  Сизов Александр Сергеевич  «21» ноября 2023г | Проверил:  Ассистент ЗИ  Исаев Глеб Андреевич  «\_\_\_» \_\_\_\_\_\_ 2023г |

Новосибирск 2023

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

**Задание к работе:**

1. Сокращение ссылки

URL: /

Метод: POST

Запрос содержит ссылку которую нужно сократить.

Ответ содержит сокращенную ссылку.

2. Переход по сокращённой ссылке

URL: /<сокращение>

Метод: GET

Ответ должен перенаправлять пользователя на исходную

страницу (см. перенаправления)

Для хранения сокращенных ссылок используем СУБД из практики

2.

2. Реализовать клиента для СУБД на том языке, на которым вы

пишите сервис сокращения ссылок.

**Задание на языке C:**

**Проект Сервера:**

**Source.cpp**

#include <iostream>

#include <thread>

#include <ctime>

#include <boost/asio.hpp>

#include <boost/random.hpp>

#include <unordered\_map>

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::resolver::query query(ip::tcp::v4(), "8080");

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

connect(socket\_client, resolver.resolve(ip::tcp::resolver::query ("127.0.0.1", "6379")));

streambuf request;

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

request\_stream << "POST /DATABASE 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 ("127.0.0.1", "6379")));

streambuf request;

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

request\_stream << "POST /DATABASE 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") {

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\_;

std::unordered\_map<std::string, std::string> data\_base;

};

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 <ctime>

#include <boost/asio.hpp>

#include <boost/random.hpp>

using namespace boost::asio;

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

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);

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\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") != std::string::npos) { // если пришел POST запрос

std::cout << "POST" << 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

WriteResponse(response, 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 WriteResponse(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);

}

ip::tcp::acceptor acceptor\_;

HashTable data\_base;

};

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:**

**1.home.html**

<!doctype html>

<html lang="en">

    <head>

        <meta charset="utf-8"/>

        <link href="main.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) {

                            // $('#result').html(data);

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

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

                            ss.text(data); // Текст ссылки

                            // Обновление содержимого div

                            $('.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'),

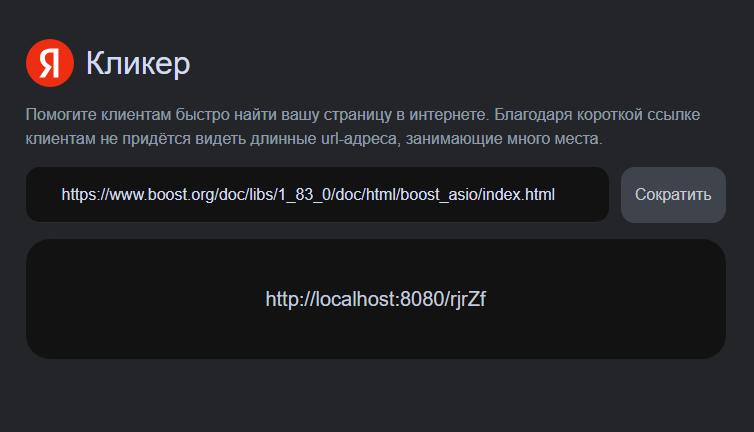
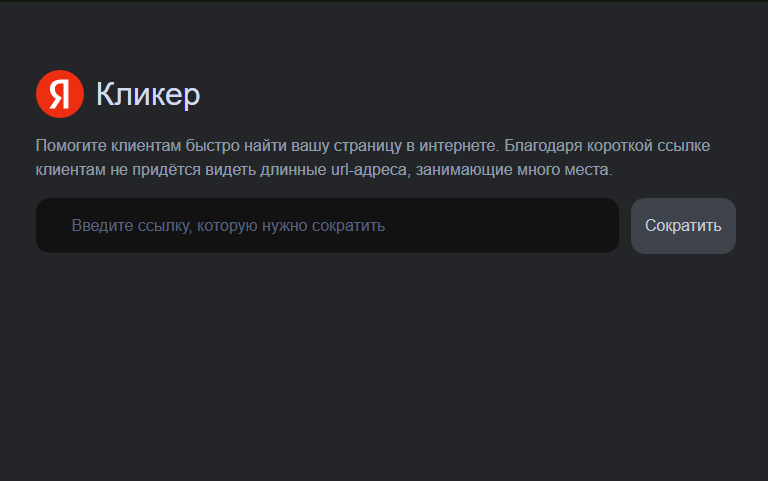
       url(//yastatic.net/islands/\_/KtHQR1erf3spayoIM4M4ngg0e2E.woff) format('woff');

}

body, button, input, textarea {

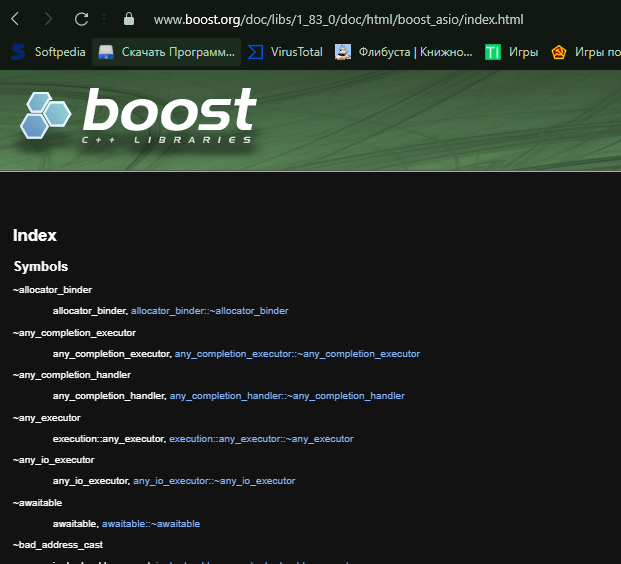
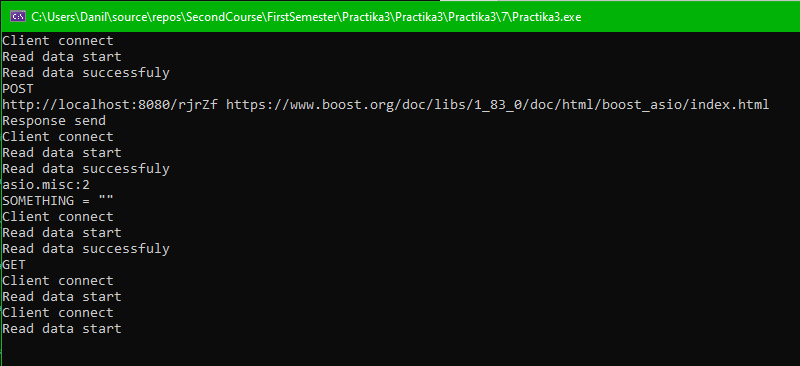
  font-family: 'YS Text', arial, sans-serif;

}

**Пример:**

Исходная ссылка: <https://www.boost.org/doc/libs/1_83_0/doc/html/boost_asio/index.html>

Полученная ссылка: <http://localhost:8080/rjrZf>

**Л**ог консоли сервера об обработке запроса: