МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра АСУ

ОТЧЕТ

по лабораторной работе №1

«Web API»

Дисциплина «Разработка клиент-серверных приложений», 6 семестр

Преподаватель: Клюшниченко А. В.

Группа: АВТ-114

Студенты:

Белетков А. Д.  
Галкин В. В.

г. Новосибирск

2024 год

**Задание**

1. С помощью языка программирования C++ написать web сервер с rest api

2. Добавить обработчики для выполнение следующих запросов:

POST /user req body: json - Добавление пользователя

GET /user/{userId} resp body: json - Получение данных выбранного пользователя

GET /users resp body: json - Получение списка пользователей

DELETE /user/{userId} resp body: empty - Удаление пользователя

(Обратить внимание, что userId - это параметр в пути запроса, который

содержит идентификатор пользователя)

3. Проверить полученный сервер при помощи утилиты curl

**Листинг сервера**

Main.cpp

#include "HTTPServer.hpp"

int main()

{

HttpServer::UPtr server = std::make\_unique<HttpServer>();

server->Start(7778);

return 0;

}

Routers.cpp

#include "Routers.hpp"

#include <thread>

#include <map>

std::map<int, nlohmann::json> users;

void route::RegisterResources(hv::HttpService &router)

{

router.POST("/user", [](HttpRequest \*req, HttpResponse \*resp)

{

nlohmann::json request;

nlohmann::json response;

try

{

request = nlohmann::json::parse(req->body);

int id = users.size();

users[id] = request;

response["msg"] = "User added successfully";

response["id"] = id;

}

catch(const std::exception& e)

{

response["error"] = "Invalid JSON";

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 400;

}

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 200;

});

router.GET("/user/{userId}",[](HttpRequest \*req, HttpResponse \*resp)

{

nlohmann::json response;

int userId = std::stoi(req->GetParam("userId"));

if (users.find(userId) != users.end())

{

response = users[userId];

}

else

{

response["error"] = "User not found";

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 404;

}

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 200;

});

router.GET("/users",[](HttpRequest \*req, HttpResponse \*resp)

{

nlohmann::json response = nlohmann::json::array();

for (auto &user : users)

{

nlohmann::json userJson;

userJson["id"] = user.first;

userJson["data"] = user.second;

response.push\_back(userJson);

}

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 200;

});

router.Delete("/user/{userId}", [](HttpRequest \*req, HttpResponse \*resp)

{

nlohmann::json response;

int userId = std::stoi(req->GetParam("userId"));

if (users.find(userId) != users.end())

{

users.erase(userId);

response["msg"] = "User deleted successfully";

}

else

{

response["error"] = "User not found";

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 404;

}

resp->SetBody(response.dump());

resp->content\_type = APPLICATION\_JSON;

return 200;

});

}

Routers.hpp

#ifndef \_ROUTERS\_HPP\_

#define \_ROUTERS\_HPP\_

#include "HttpService.h"

namespace route

{

void RegisterResources(hv::HttpService &router);

}

#endif

HTTPServer.cpp

#include "HTTPServer.hpp"

#include <thread>

HttpServer::HttpServer()

{

\_server = std::make\_unique<hv::HttpServer>();

route::RegisterResources(\_router);

\_server->registerHttpService(&\_router);

}

void HttpServer::Start(int port)

{

\_server->setPort(port);

\_server->setThreadNum(std::thread::hardware\_concurrency());

\_server->run();

}

HttpServer::~HttpServer()

{

\_server->stop();

}

HTTPServer.hpp

#ifndef \_HTTP\_SERVER\_HPP\_

#define \_HTTP\_SERVER\_HPP\_

#include "HttpServer.h"

#include "HttpService.h"

#include "Routers.hpp"

class HttpServer final

{

public:

using UPtr = std::unique\_ptr<HttpServer>;

HttpServer();

HttpServer(const HttpServer &) = delete;

HttpServer(HttpServer &&) = delete;

~HttpServer();

void Start(int port);

private:

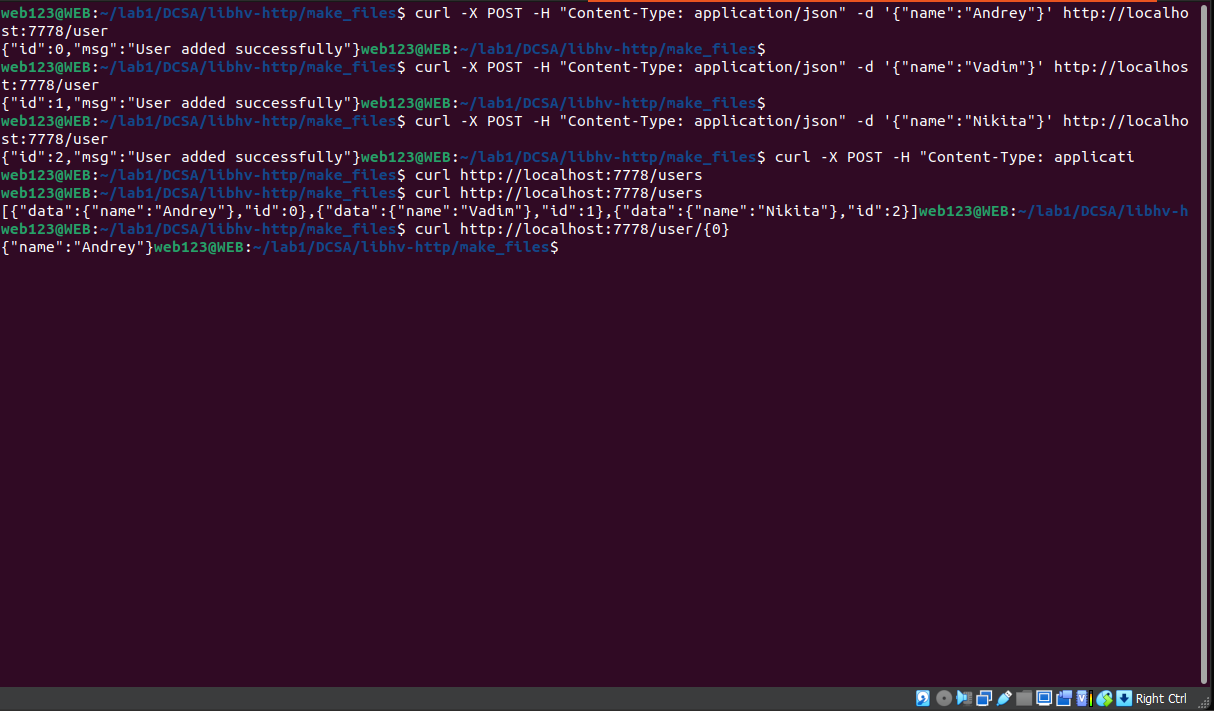
std::unique\_ptr<hv::HttpServer> \_server;

HttpService \_router;

};

#endif

**Результат добавления нескольких пользователей**

****

**Выводы**

В ходе выполнения данной лабораторной работы был разработан веб-сервер с REST API на языке программирования C++. Были добавлены обработчики для выполнения следующих запросов: добавление пользователя (POST /user), получение данных выбранного пользователя (GET /user/{userId}), получение списка пользователей (GET /users) и удаление пользователя (DELETE /user/{userId}).

Все обработчики были успешно реализованы и протестированы с использованием утилиты curl. Все запросы работают корректно, и сервер правильно обрабатывает все возможные сценарии использования.

Цель лабораторной работы была достигнута и наш проект можно использовать для дальнейшего увеличения функционала сервера, а также автоматизировать его работу.