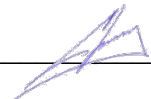


**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

СОГЛАСОВАНО

Преподаватель департамента
программной инженерии ФКН,
кандидат компьютерных наук

 С.А.Виденин
10_марта_2024 г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, кандидат технических наук

Н.А. Павлочев
10_марта_2024 г.


ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ НАСТРАИВАЕМЫЙ HTTP СЕРВЕР

Руководство оператора

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.04-01 34 01-1-ЛУ

Исполнитель
студент группы БПИ214

 / Е.К.Фортов/
10_марта_2024 г.

УТВЕРЖДЕН
RU.17701729.04.04-01 ТЗ 01-1-ЛУ

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ НАСТРАИВАЕМЫЙ HTTP СЕРВЕР

Руководство оператора
RU.17701729.04.04-01 34 01-1-ЛУ

Листов 11

<i>Инв. № подл</i>	
<i>Подп. и дата</i>	
<i>Взам. инв. №</i>	
<i>Инв. № дубл.</i>	
<i>Подп. и дата</i>	

Оглавление

1. НАЗНАЧЕНИЕ ПРОГРАММЫ.....	3
1.1. Функциональное назначение.....	3
1.2. Эксплуатационное назначение.....	3
1.2. Состав функций.....	3
2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ.....	5
2.1. Минимальный состав аппаратных средств.....	5
2.2. Минимальный состав программных средств.....	5
2.3. Требования к персоналу (пользователю).....	5
3. ВЫПОЛНЕНИЕ ПРОГРАММЫ.....	6
3.1. Загрузка программы.....	6
3.2. Запуск программы.....	6
3.3. Выполнение программы.....	6
3.3.1. Создание сервера.....	6
3.3.2. Добавление эндпоинтов к серверу.....	6
3.3.3. Запуск сервера.....	6
3.3.4. Остановка сервера (необязательно).....	6
3.3.5. Наследование отдельных компонентов сервера.....	7
4. СООБЩЕНИЯ ОПЕРАТОРУ.....	9
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.....	10

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1. НАЗНАЧЕНИЕ

1.1. Функциональное назначение

Функциональным назначением программы является предоставление программисту возможности быстро и удобно проектировать REST API на языке C++, не погружаясь в детали реализации REST API. Данный продукт является отличным решением для команд разработки, пишущих на C++, так как позволит им не менять стек разработки, в том числе и ЯП, при появлении требования в необходимости наличия REST API для каких бы то ни было целей.

1.2. Эксплуатационное назначение

Многие существующие IT продукты написаны на C++. Этот ЯП славится своей производительностью и универсальностью, однако многие более новые языки (например, Джава, C#, Го) позволяют разрабатывать такие же продукты в разы быстрее.

Зачастую командам невозможно поменять стек разработки по разным причинам (например, слишком большое наследие, функционал которого нельзя перенести на современный технологический стек с точки зрения бизнес-value и затраченного на разработку времени). Чтобы не оказаться вне рынка с текущим «устаревшим» продуктом, его разработчикам приходится имплементировать современные features на старом технологическом стеке. Для одной из таких features, а именно REST API, которое присутствует во многих промышленных системах, и предназначена данная библиотека. Она призвана значительно сокращать время на разработку REST API и добиваться наибольшего значения соотношения «бизнес-value / затраченное на разработку время».

Для подключения данной библиотеки достаточно импортировать один заголовочный файл, который, в свою очередь, будет подключать другие заголовочные файлы. Настраивать зависимости будет система автоматизации сборки проектов CMake. Такой способ подключения является наиболее современным и простым, и именно поэтому многие разработчики придерживаются такого подхода при написании собственных библиотек, фреймворков и модулей.

1.3. Состав функций

Программа должна давать пользователю возможность выполнять следующие функции:

- инстанцировать объект http сервера

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

- базово конфигурировать http сервер
- наследовать класс http сервера под свои нужды
- создавать status line http ответа из готовых шаблонов: определять методы GET и POST, код возвращаемого значения
- создавать заголовки http ответа из готовых шаблонов: content-type, content-length и т. д.
- создавать тела http ответа из готовых шаблонов, отдельных html файлов
- настраивать кастомное логирование с разными уровнями в отдельный файл
- настраивать кастомное логирование с разными уровнями в syslog
- кешировать http ответы
- обрабатывать ошибки
- читать комментарии в коде сервера, которых будет достаточно для использования всех возможностей сервера
- подключать сервер через .hrr файл с отдельной папкой (где будут все остальные файлы-зависимости лежать), настройка необходимых зависимостей идет через готовый CmakeLists.txt

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1. Минимальный состав аппаратных средств

Для бесперебойной работы программного продукта требуется компьютер с:

- Объемом свободной встроенной памяти не меньше 100 МБ,
- Объёмом оперативной памяти не меньше 1 ГБ.

2.2. Минимальный состав программных средств

Для бесперебойной работы программного продукта требуется компьютер с:

- 1) Установленной версией компилятора gcc - 14, clang — 3.4
- 2) Операционной системой со стабильной сборкой, выпущенной не позднее 2015 года
- 3) Необходимыми библиотеками (все, которые перечислены в include-тегах)

2.3. Требования к персоналу (пользователю)

Необходимое количество персонала – 1 человек.

Необходимая квалификация персонала – пользователь.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 ТЗ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

В данном разделе описан пример работы с программой.

3.1. Загрузка программы

Для загрузки данного ПО надо скачать файл ServeMe.hpp и обеспечить наличие подключенных к проекту необходимых библиотек (перечислены в include-тегах .hpp-файла).

3.2. Запуск программы

Для подключения HTTP-сервера к основной программе необходимо прописать #include «profiler.hpp» (см. рис. 1):

```
#include "HttpRESTAPI.hpp"
```

3.3. Выполнение программы

3.3.1. Создание сервера

Для создания сервера необходимо проинстанцировать объект класса RESTAPIAPP из namespace Utils. Необходимо указать порт, на котором будет в дальнейшем запущен сервер.

```
using namespace Utils;  
RESTAPIAPP app( port: 8080);
```

3.3.2. Добавление эндпоинтов к серверу

Для создания эндпоинтов нужно указать название эндпоинта, ответ в виде raw text или файла (полный путь), а также метод - «GET» или «POST».

```
app.AddEndpoint( path: "/data", response: "Some data!", method: "GET");  
app.AddEndpoint( path: "/data_from_file", response: "@file:/Users/egorfortov/CLionProjects/HTTP_Server_Egon_Fortov/index.html", method: "GET");  
app.AddEndpoint( path: "/submit", response: "Submitted!", method: "POST");
```

3.3.3. Запуск сервера

Непосредственный старт сервера:

```
app.RunServer();
```

3.3.4. Остановка сервера (необязательно)

```
app.StopServer(); // not necessary
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

3.3.5. Наследование отдельных компонентов сервера

Также, для переиспользования написанного кода, допускается создавать свои собственные классы и наследовать их от существующих в данном фреймворке. Также можно наследовать интерфейсы и создавать новую функциональность. Так, например, сделано во фреймворке:

```
namespace Interfaces {
    class HttpServerInterface {
    public:
        virtual void addEndpoint(const std::string &path, const std::string &response, Method method) = 0;
    };
    class LoggerInterface {
    public:
        virtual void log(Level level, const std::string &message) noexcept = 0;
    };
    class HttpSessionInterface {
    public:
        virtual void start() = 0;
    };
    class RESTAPIAppInterface {
    public:
        virtual void AddEndpoint(const std::string &path, const std::string &response, const std::string &method) = 0;
        virtual void RunServer() noexcept = 0;
        virtual void StopServer() noexcept = 0;
    };
} // namespace Interfaces
```

```
class Logger : Interfaces::LoggerInterface {
public:
    Logger(const std::string &program_name = "HTTPServer", const std::string &log_file_name = "log.txt",
        bool syslog_enabled = true) try : syslogEnabled(syslog_enabled) {
        try {
            if (syslog_enabled) {
                openlog(program_name.c_str(), LOG_CONS | LOG_NDELAY | LOG_PID, LOG_USER);
            }
            logFile.open(s: log_file_name, mode: std::ios::out | std::ios::app);
#ifdef DEBUG
            std::cout << getPrefix(Level::Debug) << " Logger object created";
#endif
        } catch (...) {
            std::cerr << getPrefix(level: Level::Critical) + " Failed to open log file and/or system log\n";
        }
    } catch (...) {
        std::cerr << "Failed to create Logger object\n";
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата


```
class HttpSession : public std::enable_shared_from_this<HttpSession>, Interfaces::HttpSessionInterface {
public:
    HttpSession(boost::asio::ip::tcp::socket socket,
                const endpoints &endpoints,
                Logger::Ptr logger,
                CACHE& cache,
                bool enable_cache = true)
    try : socket(std::move(socket)), endpoints_(endpoints), logger(logger), cache(cache), enable_cache(enable_cache) {
#ifdef DEBUG
        logger->log(Level::Debug, "HttpSession object created");
#endif
    } catch (...) {
        logger->log(level: Level::Error, message: "Failed to create HttpSession object");
    }

    void start() override {
        do_read();
    }
};
```

```
class HttpServer : Interfaces::HttpServerInterface {
public:
    HttpServer(boost::asio::io_context &io_context,
                Logger::Ptr logger,
                CACHE& cache,
                short port = 8080,
                bool enable_cache = true)
    try : acceptor_( &io_context, endpoint: boost::asio::ip::tcp::endpoint( internet_protocol: boost::asio::ip::tcp::v4(), port_num: port)),
        socket_( &io_context),
        enable_cache(enable_cache),
        logger(logger),
        cache(cache)
    {
        do_accept();
#ifdef DEBUG
        logger->log(Level::Debug, "HttpServer object created");
#endif
    } catch (...) {
        logger->log(level: Level::Critical, message: "Failed to create HttpServer object");
    }
};
```

```
class RESTAPIAPP : Interfaces::RESTAPIAPPInterface {
public:
    RESTAPIAPP(uint32_t port = 8080, const std::string& logfileName="log.txt")
    try {
        logger = std::make_shared<Logger>(logfileName);
        server = std::make_shared<HttpServer>(&io_context, logger, &cache, port);
#ifdef DEBUG
        logger->log(Level::Debug, "RESTAPIAPP object created");
#endif
    } catch (...) {
        std::cerr << getPrefix(level: Level::Critical) << " Failed to build RESTAPIAPP object";
    }
};
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

4. СООБЩЕНИЯ ОПЕРАТОРУ

В программе допускаются исключения и выбросы сигналов, которые могут происходить по причине некорректного пользования программой, а также по любой причине, по которой могут происходить ошибки при работе с используемыми библиотеками (см. документацию используемых библиотек).

Например, ошибка может возникнуть при попытке запустить сервер на занятом порту.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

- 1) Статья про REST API в целом [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/483202/>, свободный. (дата обращения: 9.03.2024)

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 34				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]