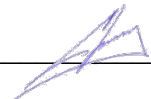


**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

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

Программа и методика испытаний

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.04-01 ТЗ 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

Программа и методика испытаний

RU.17701729.04.04-01 ТЗ 01-1-ЛУ

Листов 15

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

Оглавление

1. ОБЪЕКТ ИСПЫТАНИЙ.....	3
1.1. Наименование программы.....	3
1.2. Краткая характеристика области применения.....	3
2. ЦЕЛЬ ИСПЫТАНИЙ.....	4
3. ТРЕБОВАНИЯ К ПРОГРАММЕ.....	5
3.1. Требования к функциональным характеристикам.....	5
3.1.1. Требования к составу выполняемых функций.....	5
3.1.2. Требования к организации входных данных.....	5
3.1.3. Требования к организации выходных данных.....	6
3.2. Требования к интерфейсу.....	6
3.3. Требования к надежности.....	6
4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ.....	7
5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ.....	8
5.1. Технические средства, используемые во время испытаний.....	8
5.2. Программные средства, используемые во время испытаний.....	8
5.3. Порядок проведения испытаний.....	8
5.4. Загрузка программы для испытания.....	8
6. МЕТОДЫ ИСПЫТАНИЙ.....	9
6.1. Испытание выполнения требований к программной документации.....	9
6.2. Испытание выполнения требований к функциональным характеристикам.....	9
6.3. Испытание выполнения требований к интерфейсу.....	13
6.4. Испытание выполнения требований к надёжности.....	13
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.....	14

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

1.

1.1. Наименование программы

Наименование программы – «Высокопроизводительный Настраиваемый HTTP Сервер» («High Perfomance Customizable HTTP Server»).

1.2. Краткая характеристика области применения

Данный IT продукт представляет из себя высокоуровневую C++ библиотеку, которая дает возможность быстро проектировать и разворачивать REST API на языке C++, минуя такие низкоуровневые детали, как сокеты, потоки, контексты и т.д.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

2.

Целью испытаний является проверка корректности выполнения программой функций, перечисленных в разделе «Требования к программе».

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

3.

3.1.

Error: Reference source not found

Программа должна давать пользователю возможность выполнять следующие функции:

- инстанцировать объект http сервера
- 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 .hpp файл с отдельной папкой (где будут все остальные файлы-зависимости лежать), настройка необходимых зависимостей идет через готовый CmakeLists.txt

Error: Reference source not found

Входные данные — это исходный код на C++, в который портируется http сервер (исходный код может быть представлен в файлах с расширениями .cpp, .h, .hpp). В качестве системы автоматизации сборки проекта рекомендуется использовать CMake, так как в таком случае будет намного проще настроить зависимости, необходимые для подключаемого http сервера. Исходный код проекта до подключения данного фреймворка должен компилироваться успешно и проект должен собираться корректно.

В свою очередь, после подключения фреймворка (после успешного подключения всех необходимых для его работы файлов и успешной настройки необходимых зависимостей) в исходном коде создается объект http сервера. При проектировании REST

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

API для создания очередного эндпоинта необходимо воспользоваться соответствующим методом созданного http сервера.

Касательно требований к входным данным, программисту необходимо ознакомиться с внутренней справкой / документацией http сервера, которая исчерпывающе описывает, как с помощью него проектировать REST API.

3.1.3. Требования к организации выходных данных

Результат работы сервера должен представлять собой действующий REST API, а также файлы с логированием. Отследить корректность работы можно с помощью логов, настроенных на максимально возможный уровень — DEBUG, а также с помощью непосредственно функционального тестирования написанного REST API. Если в логах была обнаружена хоть одна ошибка, сервер имеет неопределенное поведение.

3.2. Требования к интерфейсу

Графический интерфейс у данного сервиса фактически отсутствует, так как все команды прописываются именно в исходном файле.

3.3. Требования к надежности

Дополнительные требования к надежности не предъявляются, кроме тех, которые предъявляются к использующимся библиотекам. При корректном использовании фреймворка (то есть при обеспечении необходимых условий для корректной отработки используемых функций библиотек) программа не должна завершаться аварийно.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации:

- «Высокопроизводительный Настраиваемый HTTP Сервер». Техническое задание (ГОСТ 19.201-78 [2])
- «Высокопроизводительный Настраиваемый HTTP Сервер». Программа и методика испытаний (ГОСТ 19.301-79 [3])
- «Высокопроизводительный Настраиваемый HTTP Сервер». Текст программы (ГОСТ 19.401-78 [4])
- «Высокопроизводительный Настраиваемый HTTP Сервер». Пояснительная записка (ГОСТ 19.404-79 [5])
- «Высокопроизводительный Настраиваемый HTTP Сервер». Руководство оператора (ГОСТ 19.505-79 [6])

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ

5.1. Технические средства, используемые во время испытаний

Ноутбук Macbook Pro 14 дюймов, M1.

5.2. Программные средства, используемые во время испытаний

На ноутбуке имеется:

- ОС Mac OS Ventura 13.2.1
- ОЗУ 16 ГБ
- компилятор clang последней стабильной версии

5.3. Порядок проведения испытаний

Испытания должны проводиться в следующем порядке:

- проверка требований к программной документации,
- проверка требований к функциональным характеристикам,
- проверка требований к интерфейсу,
- проверка требований к надёжности.

5.4. Загрузка программы для испытания

Тестируемая программа — main.cpp. В нее подключен ServeMe.hpp, настроен корректно CmakeLists.txt, проект компилируется и собирается успешно.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

6. МЕТОДЫ ИСПЫТАНИЙ

6.1. Испытание выполнения требований к программной документации

Состав программной документации проверяется визуально, проверяется наличие программной документации в системе LMS. Также визуально проверяется соответствие документации требованиям ГОСТ.

Все документы удовлетворяют представленным требованиям.

6.2. Испытание выполнения требований к функциональным характеристикам

Из технического задания имеем:

Программа должна давать пользователю возможность выполнять следующие функции:

- инстанцировать объект http сервера:

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

- базово конфигурировать http сервер:

```
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: boo
                  socket_( &io_context),
                  enable_cache(enable_cache),
                  logger(logger),
                  cache(cache)
    {
```

- наследовать класс http сервера под свои нужды:

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
class MyServer : Utils::HttpServer {  
    // my custom server implementation  
};
```

- создавать status line http ответа из готовых шаблонов: определять методы GET и POST, код возвращаемого значения:

Эта возможность обеспечивается готовыми шаблонами:

```
namespace Templates::Responses {  
    const auto OK : string(const string &, const string &) const = [](const std::string &body = "Hello, World!", const std::string &content_type = "text/html") {  
        return "HTTP/1.1 200 OK\r\nContent-Length: " + std::to_string( val: body.length()) + "\r\nContent-Type: " + content_type + "\r\n\r\n" + body;  
    };  
    const auto NOT_OK : string(const string &) const = [](const std::string &body = "404 Not Found!") -> string {  
        return "HTTP/1.1 404 Not Found\r\nContent-Length: 14\r\n\r\n" + body;  
    };  
};  
// namespace Templates::Responses
```

- создавать заголовки http ответа из готовых шаблонов: content-type, content-length: аналогично предыдущему пункту
- создавать тела http ответа из готовых шаблонов, отдельных html файлов:

Есть возможность указать конкретный .html-файл:

```
app.AddEndpoint( path: "/data_from_file", response: "@file:/Users/egorfortov/CLionProjects/HTTP_Server_Egor_Fortov/index.html", method: "GET");
```

- настраивать кастомное логирование с разными уровнями в отдельный файл:

Уровни логирования:

```
enum class Level {  
    Debug = 0,  
    Info,  
    Warning,  
    Error,  
    Critical  
};
```

Функция, которая отвечает за логирование в отдельный файл:

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
void writeToFile(Level level, const std::string &message) { // @TODO later: asynchronous write
    std::string prefix = std::move(getPrefix(level));
    char buffer[80] = {0};
    std::time_t result = std::time(nullptr);
    std::strftime(buffer, 80, "%Y-%m-%d %H:%M:%S", std::localtime(&result));
    std::lock_guard lock(& mu);
    logFile << buffer << " " << prefix << " " << message << std::endl;
}
```

- настраивать кастомное логирование с разными уровнями в syslog:

Функция, которая отвечает за логирование в syslog:

```
void writeToSyslog(Level level, const std::string &message) { // @TODO later: asynchronous write
    int priority = getPriority(level);
    char buffer[80] = {0};
    std::time_t result = std::time(nullptr);
    std::strftime(buffer, 80, "%Y-%m-%d %H:%M:%S", std::localtime(&result));
    std::lock_guard lock(& mu);
    syslog(priority, "%s", (std::string(s: buffer) + message).c_str()); // @TODO: check workability
}
```

- кешировать http ответы:

Для кеширования создан отдельный объект «cache», в который можно сохранять запросы-ответы и читать из него:

CACHE cache;

```
if (enable_cache && cache.find( k: method) != cache.end()) {
    do_write( response: cache[method]);
    logger->log( level: Level::Info, message: "Endpoint " + path + " of type " + method + " responding...");
} else {
    std::string body = std::move(getBody( str: it->second.first, logger));
    std::string response = Templates::Responses::OK(body);
    do_write(response);
    logger->log( level: Level::Info, message: "Endpoint " + path + " of type " + method + " responding...");
    if (enable_cache) {
        cache.insert( x: std::make_pair( &: method, &: response));
    }
}
```

- обрабатывать ошибки:

Обработка ошибок (в частности, исключений) встречается очень часто во всех компонентах REST API.

- читать комментарии в коде сервера, которых будет достаточно для использования всех возможностей сервера:

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

Некоторые строки кода REST API снабжены исчерпывающими комментариями. Более того, до кода в .hpp файле приведена документация, которая раскрывает возможности данного сервера и помогает пользователю быстрее вникнуть в суть проекта и быстро использовать/переиспользовать приведенный код.

- подключать сервер через .hpp файл с отдельной папкой (где будут все остальные файлы-зависимости лежать), настройка необходимых зависимостей идет через готовый CmakeLists.txt:

Было принято решение поместить весь код программы в один .hpp-файл с целью простоты подключения и использования.

Подключение файла:

```
#include "ServeMe.hpp"
```

В CmakeLists.txt можно также настроить подключение этого файла к проекту:

```
cmake_minimum_required(VERSION 3.0)
project(HTTP_Server_Egor_Fortov)

set(CMAKE_CXX_STANDARD 17)

set(Boost_NO_SYSTEM_PATHS ON)

# Установка переменной BOOST_ROOT на основе текущего каталога
set(BOOST_ROOT ${CMAKE_CURRENT_SOURCE_DIR}/boost)

# Настройка пути для поиска заголовочных файлов Boost
set(BOOST_INCLUDEDIR ${BOOST_ROOT})

# Путь для поиска библиотек Boost
set(BOOST_LIBRARYDIR ${BOOST_ROOT}/lib)

# Подключение заголовочных файлов Boost
find_package(Boost REQUIRED)
include_directories(${Boost_INCLUDE_DIRS})

add_executable(HTTP_Server_Egor_Fortov ServeMe.hpp main.cpp temp.h)

# Линковка с библиотеками Boost
target_link_libraries(HTTP_Server_Egor_Fortov ${Boost_LIBRARIES})
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

6.3. Испытание выполнения требований к интерфейсу

Требования к интерфейсу проверяются визуально. Представленный интерфейс приложения в предыдущем разделе удовлетворяет всем требованиям.

6.4. Испытание выполнения требований к надёжности

Программа была протестирована > 20 раз на разных тестовых кодах. Ошибок не обнаружено.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

- 1) ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
- 2) ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 3) ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 4) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 5) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 6) ГОСТ 19.301-79 Программа и методика испытаний. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 7) ГОСТ 19.401-78 Текст программы. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 8) ГОСТ 19.404-79 Пояснительная записка. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 9) ГОСТ 19.505-79 Руководство оператора. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
- 10) ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.04-01 51				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

[illegible]