Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Реализация статического веб-сервера для отдачи контента с диска»

Студент ИУ7-73Б (Группа) _____ (Подпись, дата) В. Марченко (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) Е. А. Тихомирова (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка X с., X рис., X табл., X источн., X прил.

КОМПЬЮТЕРНЫЕ СЕТИ, СТАТИЧЕСКИЙ СЕРВЕР, ВЕБ-СЕРВЕР, HTTP

Объектом разработки является статический веб-сервер для отдачи контента с диска.

Объектом исследования является ...

Цель работы: реализация статического веб-сервера для отдачи контента с диска.

В результате выполнения работы была реализован статический веб-сервер для отдачи контента с диска.

В ходе проведения исследования было установлено, что ...

Область применения результатов — ...

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей расчетно-пояснительной записке к курсовой работе применяют следующие сокращения и обозначения:

ПО	Программное обеспечение

ВВЕДЕНИЕ

Целью курсовой работы является реализация статического веб-сервера для отдачи контента с диска.

Задачами данной работы являются:

1) изучить понятие статического веб-сервера;
2) спроектировать архитектуру статического веб-сервера;
3) реализовать спроектированный статический веб-сервер для отдачи контента с диска;
4) провести нагрузочное тестирования при помощи apache benchmark с nginx.

1 Аналитическая часть

1.1 Статический веб-сервер

Понятие «веб-сервер» может относиться как к аппаратному, так и к программному обеспечению. Или даже к обеим частям, работающим совместно. С точки зрения аппаратного обеспечения, веб-сервер — это компьютер, который хранит файлы сайта (HTML-документы, CSS-стили, JavaScript-файлы, картинки и пр.) и доставляет их на устройство конечного пользователя (веб-браузер и т. д.). Он подключен к сети Интернет и может быть доступен через доменное имя. С точки зрения программного обеспечения, веб-сервер включает в себя несколько компонентов, которые контролируют доступ веб-пользователей к размещенным на сервере файлам, как минимум — это HTTP-сервер. HTTP-сервер — это часть программного обеспечения, которая понимает URL-адреса (веб-адреса) и HTTP (протокол, который браузер использует для просмотра веб-страниц) [1].

На самом базовом уровне, когда браузеру нужен файл, размещенный на веб-сервере, браузер запрашивает его через HTTP-протокол. Когда запрос достигает нужного веб-сервера (аппаратное обеспечение), сервер HTTP (программное обеспечение) принимает запрос, находит запрашиваемый документ (если нет, то сообщает об ошибке 404) и отправляет обратно, также через HTTP. На рисунке 1.1 показана схема взаимодействия веб-сервера и браузера [1].

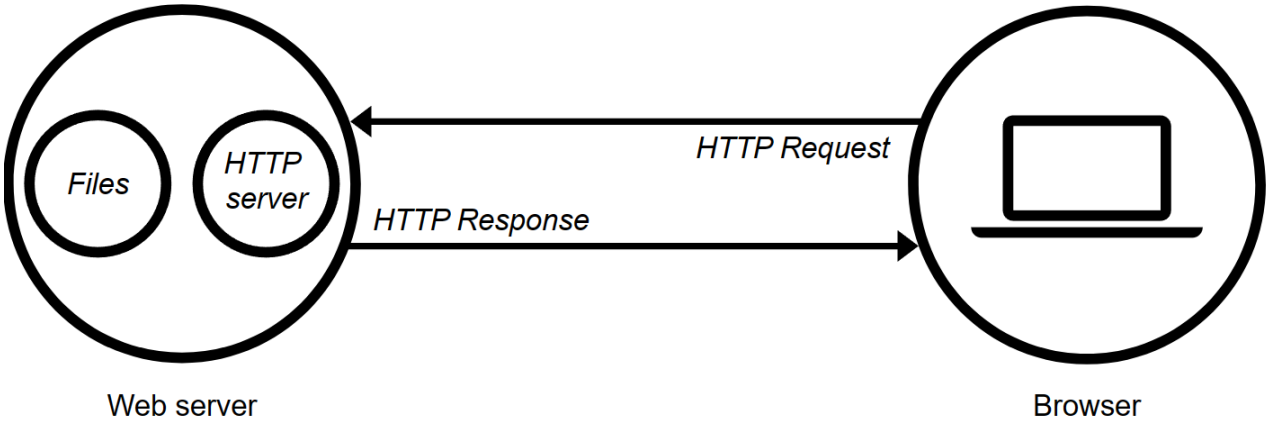

Рисунок 1.1 – Схема взаимодействия веб-сервера и браузера [1]

Статический веб-сервер, или стек, состоит из компьютера (аппаратное

обеспечение) с сервером HTTP (программное обеспечение). Это называют «статикой», потому что сервер посылает размещенные файлы в браузер «как есть» [1].

Динамический веб-сервер состоит из статического веб-сервера и дополнительного программного обеспечения, чаще всего сервера приложения и базы данных. Его называют «динамическим», потому что сервер приложений изменяет исходные файлы перед отправкой в браузер по HTTP [1].

Например, для получения итоговой страницы, которую пользователь просматривает в браузере, сервер приложений может заполнить HTML-шаблон данными из базы данных. Такие сайты, как MDN или Википедия, состоят из тысяч веб-страниц, но они не являются реальными HTML документами — лишь несколько HTML-шаблонов и гигантские базы данных. Эта структура упрощает и ускоряет сопровождение веб-приложений и доставку контента [1].

1.2 Структура HTTP-сообщений

HTTP сообщения — это обмен данными между сервером и клиентом. Есть два типа сообщений: запросы, отправляемые клиентом, чтобы инициировать реакцию со стороны сервера, и ответы от сервера [2].

Сообщения HTTP состоят из текстовой информации в кодировке ASCII, записанной в несколько строк. В HTTP/1.1 и более ранних версиях они пересылались в качестве обычного текста. В HTTP/2 текстовое сообщение разделяется на фреймы, что позволяет выполнить оптимизацию и повысить производительность [2].

Веб разработчики не создают текстовые сообщения HTTP самостоятельно — это делает программа, браузер, прокси или веб-сервер. Они обеспечивают создание HTTP сообщений через конфигурационные файлы (для прокси и серверов), APIs (для браузеров) или другие интерфейсы [2].

HTTP запросы и ответы имеют близкую структуру. Они состоят из следующих элементов [2].

1. Стартовая строка, описывающая запрос, или статус (успех или сбой). Это всегда одна строка.

2. Произвольный набор HTTP заголовков, определяющий запрос или описывающий тело сообщения.

3. Пустая строка, указывающая, что вся мета информация отправлена.

4. Произвольное тела, содержащее пересылаемые с запросом данные (например, содержимое HTML-формы) или отправляемый в ответ документ. Наличие тела и его размер определяется стартовой строкой и заголовками HTTP.

Стартовую строку вместе с заголовками сообщения HTTP называют головой запроса, а его данные — телом [2].

1.3 Сокеты

Сокеты (англ. socket — разъем) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения [3].

Каждый процесс может создать слушающий сокет (серверный сокет) и привязать его к какому-нибудь порту операционной системы. Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения. При этом сохраняется возможность проверить наличие соединений на данный момент, установить тайм-аут для операции и т. д [3].

Каждый сокет имеет свой адрес. ОС семейства UNIX могут поддерживать много типов адресов, но обязательными являются INET-адрес и UNIX-адрес. Если привязать сокет к UNIX-адресу, то будет создан специальный файл (файл сокета) по заданному пути, через который смогут сообщаться любые локальные процессы путем чтения/записи из него. Сокеты типа INET доступны из сети и требуют выделения номера порта [3].

Обычно клиент явно подсоединяется к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером [3].

Все сокеты обычно ориентированы на применение датаграмм, но их точные характеристики зависят от интерфейса, обеспечиваемого протоколом. Обмен между сокетами происходит по схеме, изображенной на рисунке 1.2:

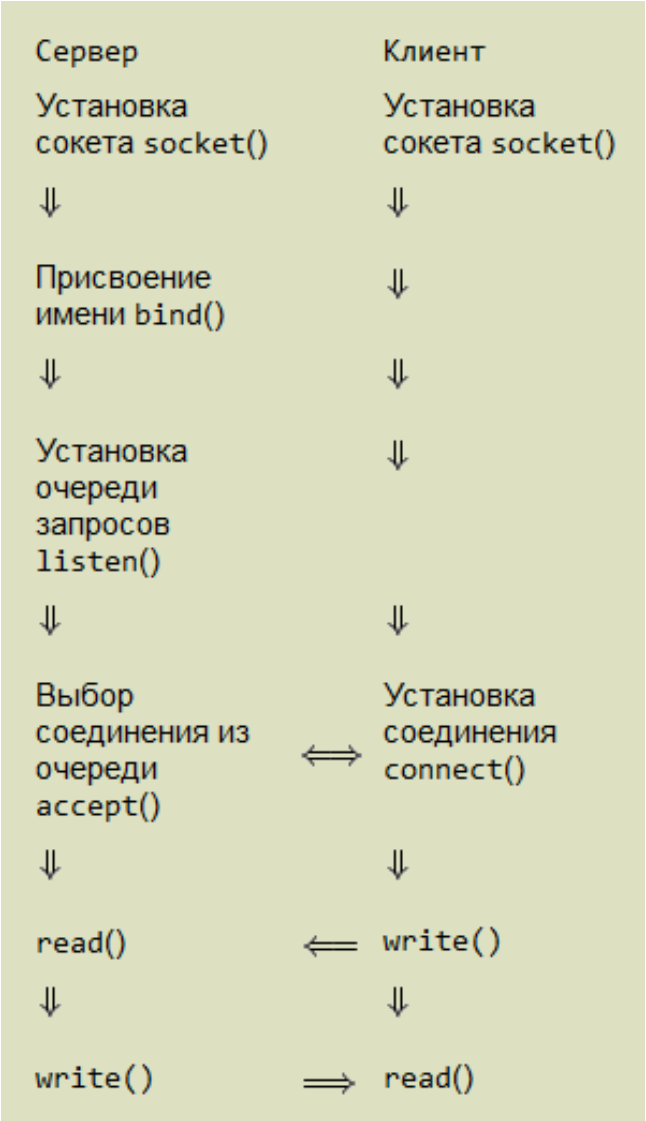

Рисунок 1.2 – Схема обмена между сокетами

1.4 Требования к статическому веб-серверу

По заданию на курсовую работу к статическому веб-серверу предъявляются следующие требования.

1. Поддержка GET и HEAD запросов (поддержка статусов 200, 403, 404).
2. Ответ на неподдерживаемые запросы статусом 405.
3. Выставление content-type в зависимости от типа файла (поддержка .html, .css, .js, .png, .jpg, .jpeg, .swf, .gif).
4. Корректная передача файлов размером в 100 МБ.
5. Сервер по умолчанию должен возвращать HTML-страницу на выбранную тему с CSS-стилем.

6. Учесть минимальные требования к безопасности статических серверов (предусмотреть ошибку в случае если адрес будет выходить за корневую директорию сервера).

7. Реализовать логгер.

8. Использовать язык C без сторонних библиотек.

9. Реализовать архитектуру prefork + pselect().

10. Статический веб-сервер должен работать стабильно.

1.5 Диаграмма вариантов использования

На рисунке 1.3 показана use case диаграмма для пользователей статического веб-сервера.

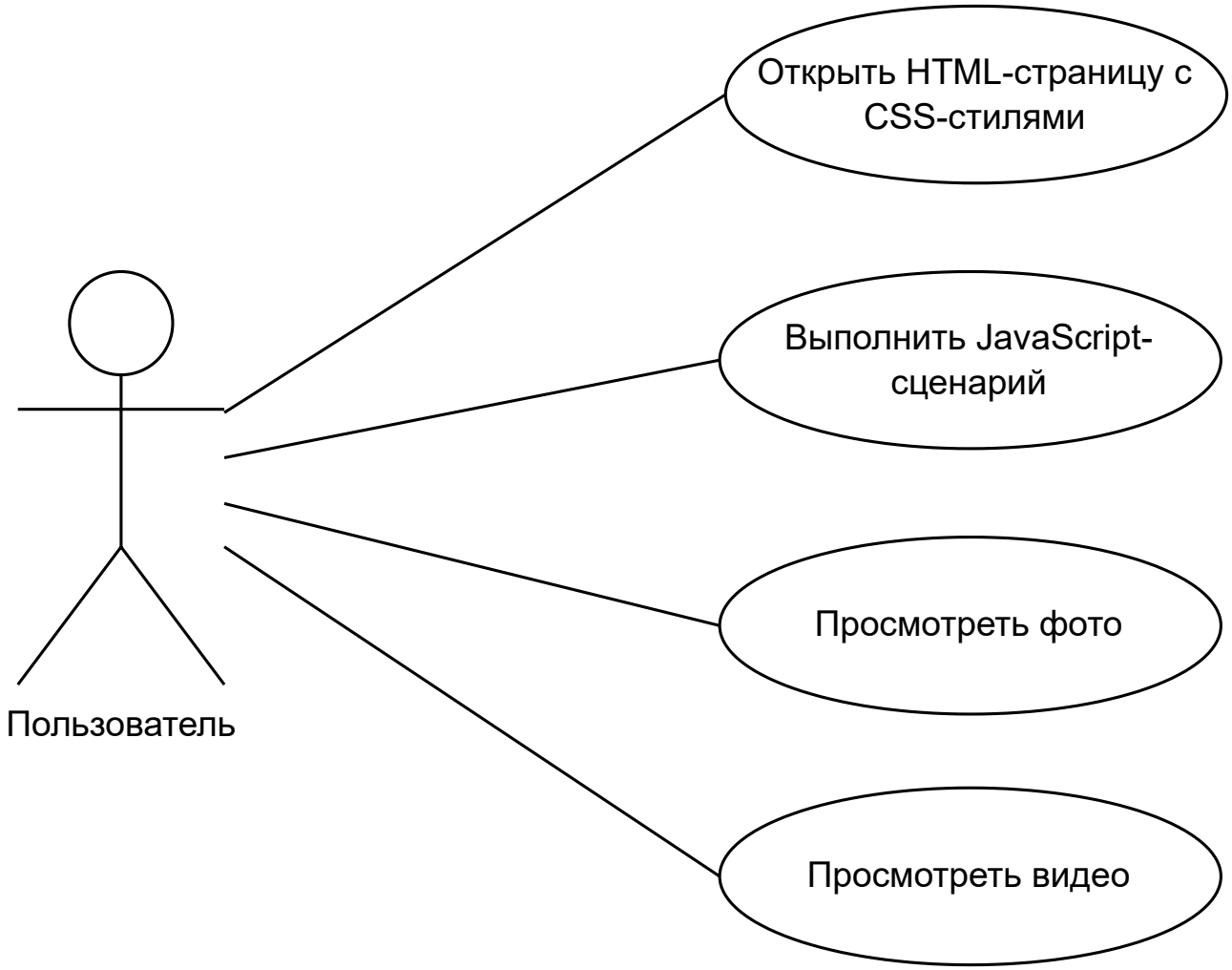

Рисунок 1.3 – Взаимодействие пользователя со статическим веб-сервером

Вывод из аналитической части

В ходе выполнения аналитической части курсовой работы был проведен анализ предметной области — изучено понятие статического веб-сервера, сформулированы требования к серверу и описаны пользовательские сценарии в виде use case диаграммы.

2 Конструкторская часть

2.1 Описание обработки HTTP-запросов

На рисунке 2.1 показана схема обработки HTTP-запросов.

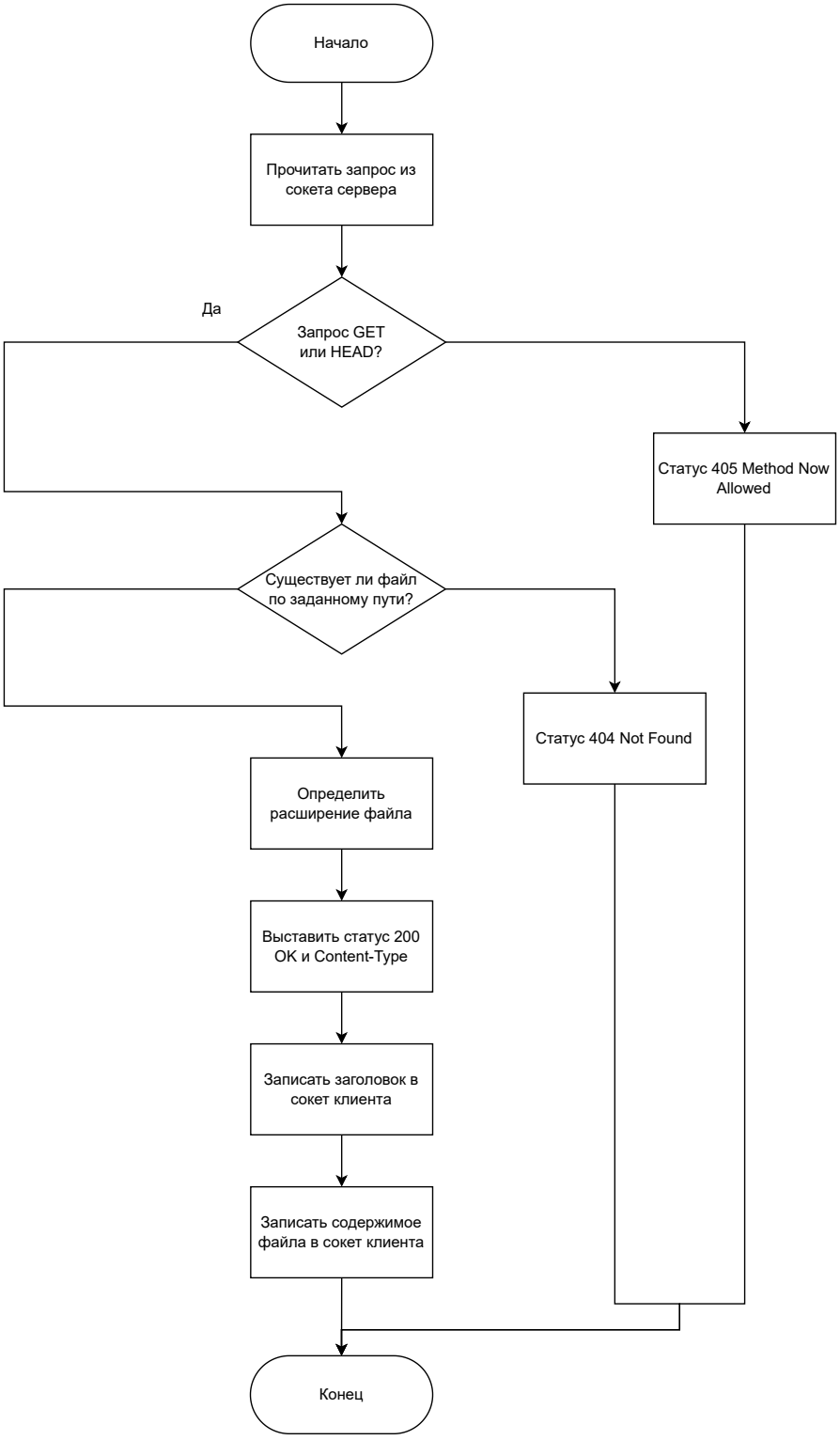

Рисунок 2.1 – Схема обработки HTTP-запросов

2.2 Описание запуска и настройки сервера

2.3 Описание логгирования

Вывод из конструкторской части

В ходе выполнения конструкторской части куросовой работы была создана диграмма базы данных; описаны сущности базы данных; описаны проектируемые ограничения целостности базы данных; описана функция на уровне базы данных, которая считает стоимость заказа, в виде схемы алгоритма и описана проектируемая ролевая модель на уровне базы данных, которая состоит из трех ролей — клиент, модератор и администратор.

3 Технологическая часть

3.1 Средства реализации

Для реализации статического веб-сервера был использован язык C (требование к курсовой работе). Сервер написан под операционные системы, работающие на базе ядра Linux. Для реализации архитектуры prefork + pselect() использовались системные вызовы fork() и pselect(). Для подготовки тестовых данных (контента для отдачи с диска) были использованы HTML, CSS и JavaScript.

3.2 Реализация обработки запросов

3.3 Реализация логгера

Вывод из технологической части

В ходе выполнения технологической части курсовой работы были выбраны средства реализации программного обеспечения; написан исходный код сущностей, ограничений целостности, проектируемой функции, ролевой модели базы данных; реализован интерфейс доступа к базе данных и протестирован разработанный функционал.

4 Исследовательская часть

4.1 Технические характеристики устройства

Технические характеристики устройства, на котором было проведено нагрузочное тестирование:

1) операционная система Linux Ubuntu;

2) оперативная память 4 ГБ;

3) процессор Intel® Core™ i7-4790K @ 4.00 ГГц.

Нагрузочное тестирование проводилось с использованием утилиты apache benchmark.

4.2 Нагрузочное тестирование

В таблице 4.1 приведены результаты нагрузочного тестирования.

Таблица 4.1 – Результаты нагрузочного тестирования

Кол-во строк	**Время выполнения запроса без индекса, мс**	**Время выполнения запроса с кластеризованным индексом, мс**	**Время выполнения запроса с некластеризованным индексом, мс**
100	468	440	450
1000	471	421	439
10000	510	425	445
50000	757	414	453
100000	1642	472	460
250000	10701	498	510

Вывод из исследовательской части

В ходе выполнения исследовательской части курсовой работы был проведен краткий обзор кластеризованных и некластеризованных индексов; определены таблицы и атрибуты, для которых имеет смысл создавать индексы, и

проведен эксперимент по измерению времени выполнения запроса без индекса, с кластеризованным и некластеризованным индексом. Согласно полученным при проведении эксперимента данным, при количестве строк в таблице до 10 тысяч существенной разницы во времени выполнения запроса не наблюдается. При большем количестве записей время выполнения запроса без использования индекса растет в разы быстрее, а время выполнения запроса с использованием двух типов индексов остается практически неизменным. Разница во времени выполнения запроса с кластеризованным и некластеризованным индексом практически отсутствует. При 250 тысячах строк в таблице использование индекса увеличивает скорость выполнения запроса в 21 раз при поиске первой записи и в 97 раз при поиске последней.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был реализован статический веб-сервера для отдачи контента с диска.

Были выполнены следующие задачи:

1) изучено понятие статического веб-сервера;
2) спроектирована архитектуру статического веб-сервера;
3) реализован спроектированный статический веб-сервер для отдачи контента с диска;
4) проведено нагрузочное тестирования при помощи apache benchmark с nginx.

В ходе проведения исследования было выявлено...

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Docs M. W.* Что такое веб-сервер. — 2023. — (Дата обращения: 10.11.2023). https://developer.mozilla.org/ru/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server.
2. *Docs M. W.* Сообщения HTTP. — 2023. — (Дата обращения: 10.11.2023). https://developer.mozilla.org/ru/docs/Web/HTTP/Messages.
3. *УрФУ К. И. И. Т. И.* Сокеты. — 2020. — (Дата обращения: 10.11.2023). https://lecturesnet.readthedocs.io/net/low-level/ipc/socket/intro.html.

ПРИЛОЖЕНИЕ А

Презентация