|  |  |
| --- | --- |
|  | **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)** |

Утверждаю

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

"\_\_\_"\_\_\_\_\_\_\_\_\_2024г.

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Сетевые технологии в АСОИУ»

Техническое задание

Вариант 5

Студенты группы РТ5-61Б:

Саркисян С.З.

Сигал Д.Э.

Иниятуллин Р.М.

2024г.

1. **Наименование:**

Распределённая информационная система обмена файлами в реальном времени.

1. **Основание для разработки:**

Основанием для разработки является учебный план МГТУ им. Баумана кафедры ИУ5 на 6 семестр.

1. **Исполнители:**

Иниятуллин Р. М. (прикладной уровень) – группа ИУ5Ц-83Б

Саркисян С. З. (транспортный уровень) – группа ИУ5-61Б

Сигал Д. Э. (канальный уровень) – группа ИУ5-61Б

1. **Цель разработки:**

Разработать распределённую систему для обмена файлами в реальном времени, состоящую из трех уровней: прикладной, транспортный и канальный, каждый из которых реализуется отдельным веб-сервисом.

1. **Функциональные требования**
   1. Прикладной уровень:
      1. Страница приложения с окном чата для отправки и просмотра полученных файлов с указанием отправителя и времени отправки;
         1. При подключении к чату пользователь должен ввести имя с помощью кнопки Войти, которое будет передаваться с каждым новым файлом;
         2. После входа появляется возможность отправки файлов по установленному WebSocket соединению, используя кнопки “Добавить файл”, “Отправить”;
         3. По кнопке “Выйти” чат и логин очищаются, а ws-подключение закрывается;
         4. В случае, если сообщение пришло с признаком ошибки, файл не отображается, а вместо него у получателей появляется соответствующее сообщение;
         5. Дизайн приложения соответствует сайту …
      2. WebSocket-сервер
         1. Хранит имена пользователей для всех ws-подключений;
         2. Позволяет устанавливать, закрывать ws-соединения, получать файлы от клиентов и широковещательно рассылать файлы подключенным клиентам;
      3. Реализация HTTP-метода Receive для получения сообщения с транспортного уровня:
         1. В json каждого сообщения указывается отправитель, время отправки, признак ошибки и полезная нагрузка;
         2. Полученное по HTTP сообщение отправляется широковещательной WebSocket рассылкой всем подключенным ws-клиентам, кроме тех, у кого логин совпадает с именем отправителя;
   2. Транспортный уровень:
      1. Реализация HTTP-метода Send для сегментирования сообщений:
         1. Разбиение сообщения на сегменты по 1000 байт и их поочередная отправка на канальный уровень через метод Code;
         2. Каждый сегмент содержит время отправки (в качестве идентификатора сообщения), общую длину сообщения, номер данного сегмента в сообщении, полезную нагрузку;
      2. Реализация HTTP-метода Transfer для передачи сообщения на прикладной уровень:
         1. Формирование очереди для полученных сегментов, которые раз в 2 секунды собираются в сообщения прикладного уровня;
         2. Если часть из сегментов сообщения не была принята, оно передается на прикладной уровень с признаком ошибки;
   3. Канальный уровень:
      1. Сервис канального уровня эмулирует канал связи с потерями:
         1. Сервис должен вносить ошибку с вероятностью 11% в один случайный бит каждого сформированного кадра;
         2. Сервис должен терять передаваемый кадр с вероятностью 2%;
      2. Реализация HTTP-метода Code для кодирования и декодирования полученного от транспортного уровня сегмента:
         1. Полученный от транспортного уровня json сегмента кодируется циклическим [7,4]-кодом для получения кадра;
         2. После внесения ошибки в кадр он декодируется с исправлением ошибки и передается далее в виде сегмента на транспортный уровень;
2. **Требования к составу технических средств:**
   1. Прикладной уровень:
      1. Серверная часть
         1. ПК с ОС Windows(7.0 и выше)
         2. Node.js (1.20 и выше)
      2. Клиентская часть
         1. ПК с ОС Windows(7.0 и выше)
         2. Веб-браузер: Chrome(40 и выше)
   2. Транспортный уровень:
      * 1. ПК с ОС MacOS(12.0 и выше)
        2. Django (4.2 и выше)
        3. Kafka (2.12 и выше)
   3. Канальный уровень:
      * 1. ПК с ОС Linux(6.4 и выше)
        2. Django (4.2 и выше)
3. **Этапы разработки:**
   1. Выбрать тему-вариант, определить команду и разработать ТЗ – 3 неделя;
   2. Разработать макет figma, три диаграммы последовательности и описать HTTP-методы в swagger – 8 неделя;
   3. Разработать и отладить приложение, подготовить полный комплект документов – 12 неделя;
   4. Исправить замечания, защитить проект – 14 неделя.
4. **Техническая документация, предъявляемая по окончании работы:**

Расчётно-пояснительная записка, включающая в приложении комплект технической документации на программный продукт, содержащий:

– Приложение 1 – Техническое задание

– Приложение 2 – Программа и методика испытаний

– Приложение 3 – Руководство пользователя

– Приложение 4 – Руководство системного администратора

1. **Порядок приемки работы:**

Приемка работы осуществляется в соответствии с "Программой и методикой испытаний."

Работа защищается перед комиссией преподавателей кафедры.

1. **Дополнительные условия:**

Данное Техническое Задание может дополняться и изменяться в установленном порядке.