|  |  |
| --- | --- |
|  | **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

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

"\_\_\_"\_\_\_\_\_\_\_\_\_2025г.

**Факультет «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Техническое задание

Вариант 33

Студенты группы ИУ5-61Б:

Ларкин Б.В.

Цыпышев Т.А.

Серов С.С.

2025г.

1. **Наименование:**

Разработать распределенную систему 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:**

Ларкин Б.В. (прикладной уровень) – группа ИУ5-61Б

Цыпышев Т.А. (транспортный уровень) – группа ИУ5-61Б

Серов С.С. (канальный уровень) – группа ИУ5-61Б

1. **Цель разработки:**

Разработать распределенную систему для обмена собщениями в реальном времени, состоящую из трех уровней: прикладной, транспортный и канальный, каждый из которых реализуется как отдельный веб-сервис.

1. **Функциональные требования**
   1. Прикладной уровень:
      1. Страница приложения представляет собой чат, где пользователи могут отправлять и получать текстовые сообщения. Каждое сообщение будет отображать информацию об отправителе и времени отправки.
         1. Соединение с сервером Земли или Марса выбирается посредством подключения к порту 8001 или 8002 веб-приложения.
         2. При подключении к чату пользователю следует ввести свое имя. Это имя будет передаваться вместе с каждым новым текстовым сообщением;
         3. После успешной авторизации пользователи с "Земли" смогут отправлять текстовые сообщения на Марс через установленное WebSocket-соединение и напрямую на Землю. Для ввода текста предусмотрено поле ввода. Для отправки сообщения предусмотрена кнопка "Отправить";
         4. У пользователей с Марса отсутствует возможность отправки сообщений. При попытке отправки пользователь увидит ошибку от сервера.
         5. Если при получении сообщения возникнет ошибка, то это сообщение не будет отображаться, а пользователи увидят вместо него уведомление о сбое;
         6. При нажатии кнопки "Выйти" все данные (чат и логин) очищаются, а активное WebSocket-соединение закрывается;
         7. Дизайн приложения должен соответствовать сайту stepik.org и быть адаптирован для мобильных устройств: <https://www.figma.com/design/ecp3I8tS1ORT2gIRPeXJ93/Material-UI-for-Figma-(and-MUI-X)-(Community)?node-id=8603-8199&t=10V4Q609mCz5jgWP-1>
      2. WebSocket-сервера
         1. WebSocket-сервера управляют списком пользователей, поддерживающих активные соединения и принадлежащих данному серверу. Они хранят информацию об именах пользователей для каждого активного соединения;
         2. Сервера предоставляют возможность открывать и закрывать WebSocket-соединения. Полученные сообщения от клиентов с Земли сервер рассылает всем подключенным пользователям с Земли, за исключением тех, чье имя совпадает с именем отправителя, а также через многоуровневую структуру приложения — клиентам на Марсе;
      3. Реализация HTTP-метода Receive для получения сообщения с транспортного уровня:
         1. Каждое сообщение, передаваемое по HTTP, содержит информацию о том, кто является отправителем, когда было отправлено сообщение, есть ли ошибка в передаче, а также сам текст сообщения;
         2. После получения сообщения по HTTP, сервер передает его всем подключенным WebSocket-клиентам через широковещательную рассылку, кроме того клиента, который отправил это сообщение или клиентов, которые находятся на Земле;
   2. Транспортный уровень:
      1. Реализация HTTP-метода Send, который отвечает за разделение текстового сообщения на несколько частей (сегментов):
         1. Каждое сообщение будет разделяться на сегменты по 140 байт. Каждый сегмент будет последовательно передаваться через метод Code на канальный уровень;
         2. Каждый сегмент будет содержать информацию о времени отправки сообщения (чтобы все части можно было собрать в правильном порядке), общую длину сообщения, порядковый номер сегмента и сам текст сообщения;
      2. Реализация HTTP-метода Transfer, который собирает полученные сегменты в полное сообщение прикладного уровня:
         1. Полученные сегменты будут храниться в очереди, и раз в 1 секунду сервер будет собирать их в одно сообщение и передавать на прикладной уровень;
         2. Если некоторые сегменты не были получены или переданы с ошибкой, сервер будет отправлять полное сообщение с признаком ошибки;
   3. Канальный уровень:
      1. Сервис канала связи имитирует возможные потери данных. Он добавляет случайные ошибки в передаваемые сообщения:
         1. Сервис должен вносить ошибку с вероятностью 10% в один случайный бит каждого сформированного кадра;
         2. Сервис должен терять передаваемый кадр с вероятностью 2%;
      2. Реализация HTTP-метода Code, который выполняет кодирование и декодирование полученных от транспортного уровня сегментов:
         1. Каждый сегмент, полученный от транспортного уровня, кодируется с использованием циклического [7,4]-кода, что позволяет обнаруживать и исправлять ошибки при передаче данных;
         2. После того как ошибка внесена в кадр, он будет декодирован, исправлен и передан дальше на транспортный уровень в виде конкретного сегмента;
2. **Требования к составу технических средств:**
   1. Прикладной уровень:
      1. Серверная часть
         1. ПК с ОС Linux(5.2 и выше)
         2. Node.js (1.20 и выше)
      2. Клиентская часть
         1. Телефон с ОС Android(10 и выше)
         2. Веб-браузер: Chrome(100 и выше)
   2. Транспортный уровень:
      * 1. ПК с ОС Linux(5.2 и выше)
        2. GoLang (4.2 и выше)
        3. Kafka (2.12 и выше)
   3. Канальный уровень:
      * 1. ПК с ОС MacOS(6.4 и выше)
        2. GoLang (4.2 и выше)
3. **Этапы разработки:**
   1. Выбрать тему-вариант, определить команду и сформировать 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 (ТЗ, РПЗ, ПМИ, РП, РСА) – 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.