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 РАДИОТЕХНИЧЕСКИЙ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Распределённая информационная система обмена сообщениями в реальном времени

Студент РТ5-61Б _____ **В.М. Кабанец**
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент РТ5-61Б _____ **В.Б. Краснов**
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент РТ5-61Б _____ **А.Е. Фруктин**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ **А.И. Канев**
(Подпись, дата) (И.О.Фамилия)

2024 г.

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
Заведующий кафедрой ___ИУ5____
(Индекс)
_____ ___В.И. Терехов__
(И.О.Фамилия)
«_09_» ____февраля_____ 2024 г.

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине ____Сетевые технологии в АСОИУ_____

Студент группы _РТ5-61Б_____Кабанец В.М. Краснов В. Б. Фруктин А.Е._____
(Фамилия, имя, отчество)

Тема курсовой работы Распределенная информационная система обмена сообщениями_____
в реальном времени

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
УЧЕБНАЯ_
Источник тематики (кафедра, предприятие, НИР) _____КАФЕДРА_

График выполнения работы: 25% к _3_ нед., 50% к _8_ нед., 75% к 12 нед., 100% к 15 нед.

Задание Разработать автоматизированную распределенную систему для обмена сообщениями в реальном времени

Оформление курсовой работы:

Расчетно-пояснительная записка на _____ листах формата А4.

Дата выдачи задания «_09_» ____февраля_____ 2024 г.

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

Студент _____ В. М. Кабанец
(Подпись, дата) (И.О.Фамилия)

Студент _____ В. Б. Краснов
(Подпись, дата) (И.О.Фамилия)

Студент _____ А. Е. Фруктин
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

История индустрии сообщений началась в 1961 году в Массачусетском технологическом институте с создания первой системы данных, позволяющей до 30 пользователей одновременно общаться в почти реальном времени. В 1971 году появился первый мессенджер в мире под названием EMISARI, который использовался госслужащими для обмена короткими текстовыми сообщениями при координации различных оперативных ситуаций в стране. Системы мгновенного обмена сообщениями продолжают оставаться важными и актуальными, причем более 2,2 миллиарда людей по всему миру используют хотя бы одно из популярных приложений для общения.

Цель данной работы - разработка системы для мгновенного обмена текстовыми сообщениями, которая будет включать веб-сервис и веб-приложение.

Система позволит пользователям обмениваться текстовыми сообщениями в режиме реального времени. Для того чтобы отправить или прочитать сообщение, пользователь должен зарегистрироваться в системе, указав свое имя. При отправке сообщения на экране отображаются имя отправителя, текст сообщения и время его отправки. Если при отправке возникает ошибка, пользователь увидит значок ошибки с сообщением «При отправке сообщения возникла ошибка». История сообщений не сохраняется при обновлении страницы. Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживаться кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть русифицируемы.

В ходе работы необходимо выполнить следующие задачи:

1. Разработать дизайн приложения.
2. Разработать сервис канального уровня на Go.
3. Разработать сервис транспортного уровня на Java.

4. Реализовать интерфейс приложения на React.

5. Подготовить набор документации, включающий РПЗ, ТЗ, ПМИ, РСА, РП.

ТРАНСПОРТНЫЙ УРОВЕНЬ

Транспортный уровень является связующим между прикладным и канальным уровнем. На него ложиться вся основная нагрузка бэкенда приложения. Пронаблюдать логику и важность транспортного уровня можно на диаграмме развертывания (рис. 1).

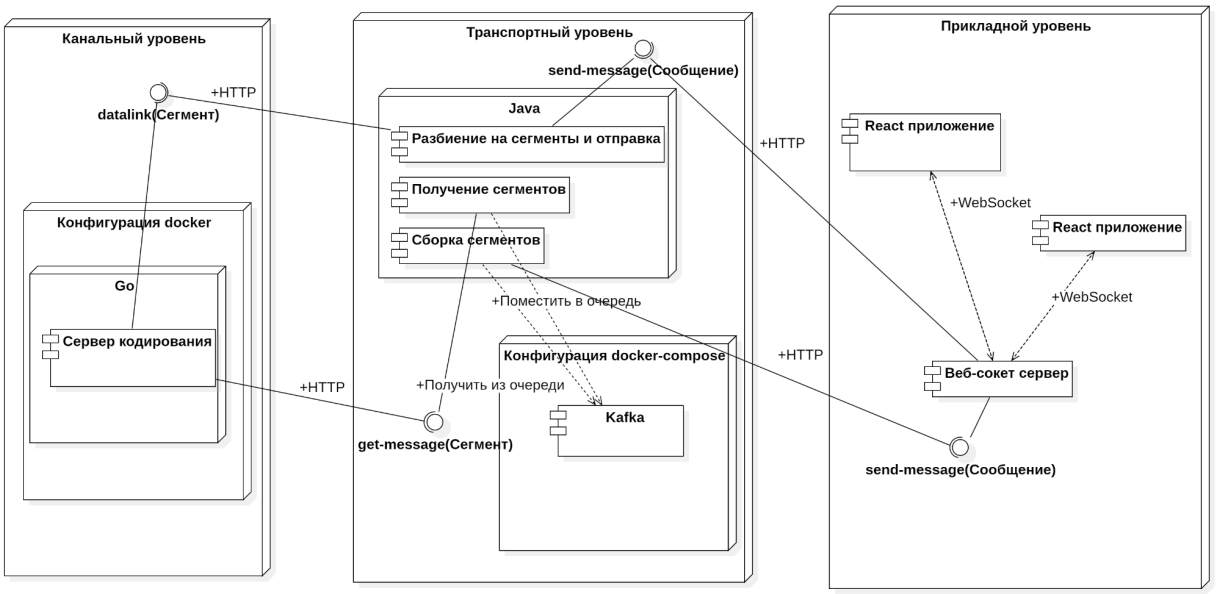

Рисунок 1 – Диаграмма развертывания приложения

Алгоритм взаимодействия транспортного уровня с прикладным и канальным можно пронаблюдать на диаграмме последовательности транспортного уровня (рис. 2).

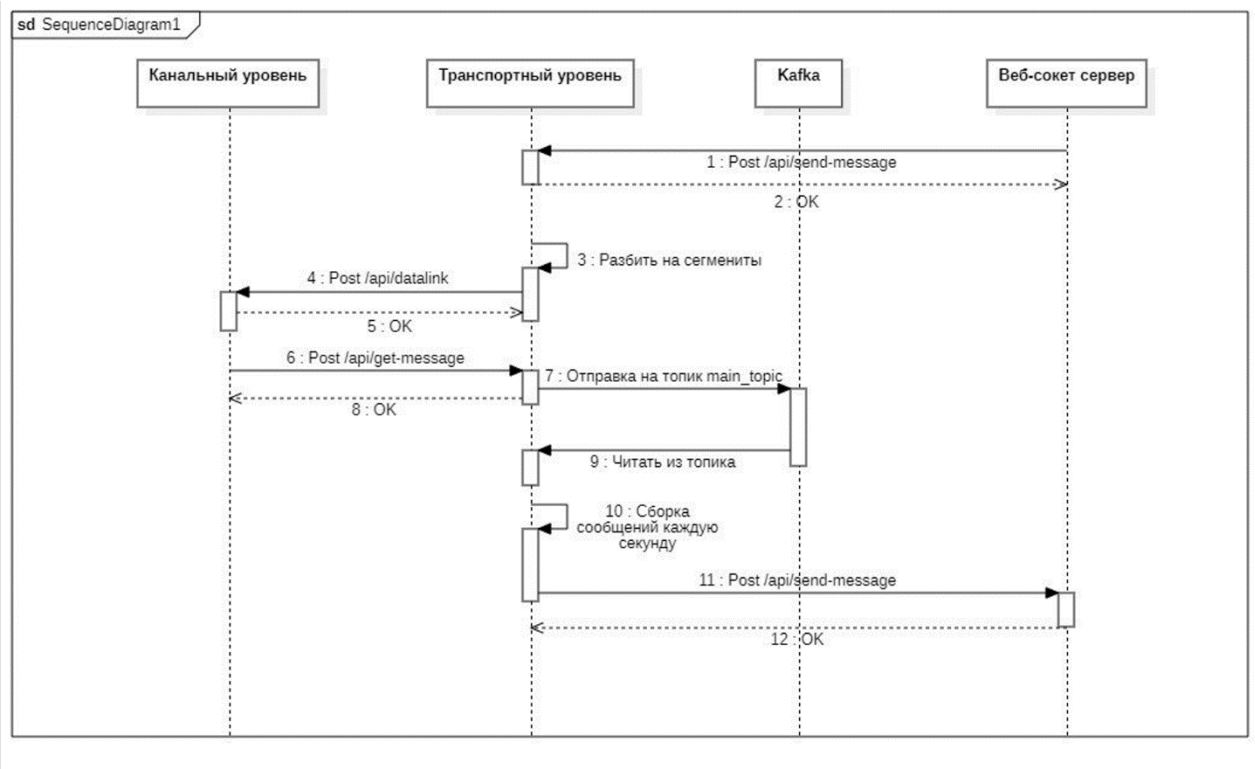

Рисунок 2 – Диаграмма последовательности транспортного уровня

Взаимодействие начинается с вызова метода send-message с прикладного уровня на транспортный, в теле которого передается сообщение отправленное одним из пользователей. После чего на транспортном уровне происходит сегментация сообщения в блоки по 100 байт и отправка каждого сегмента на канальный уровень посредством метода datalink.

Важной характеристикой транспортного уровня модели OSI является контроль перегрузки (congestion control). Контроль перегрузки – это различные алгоритмы, которые обеспечивают наиболее быструю скорость передачи данных между двумя узлами, передающими данные через TCP. Они управляют размером TCP-окна и могут ориентироваться на RTT (Round Trip Time — время от отправки запроса до получения ответа), потерю пакетов, время ожидания отправки пакета из очереди и т.д. В текущей упрощённой реализации транспортного уровня используется механизм очереди отправки пакета. Полученный с канального уровня сегмент помещается в Kafka. Kafka – это распределенный брокер сообщений, представляющий собой отказоустойчивые конвейеры, работающие по принципу «публикация/подписка» и позволяющие обрабатывать потоки событий. Из

очереди раз в 1 секунду они собираются в единое сообщение и передаются на прикладной уровень с вызовом метода Receive. В случае, когда за 2 цикла часть сегментов не была принята, то сообщение передаётся на прикладной уровень с признаком ошибки.

Методы, используемые на транспортном уровне (Таблица 1):

Таблица 1 – Методы транспортного уровня

Метод	Входные параметры	Описание
POST /api/send-message	sender string timestamp timestamp message string	Отправка сообщения с прикладного уровня на транспортный
POST /api/get-message	timestamp timestamp amountOfSegments integer sender string segmentNum integer message string error boolean	Отправка декодированного сегмента с канального уровня на транспортный

КАНАЛЬНЫЙ УРОВЕНЬ

Данный уровень эмитирует взаимодействие с удаленным сетевым узлом через канал с помехами. Для этого используется циклический код [15;11] для кодирования передаваемых сообщений. Циклический код —

линейный, блочный код, обладающий свойством цикличности, то есть каждая циклическая перестановка кодового слова также является кодовым словом. Используется для преобразования информации для защиты её от ошибок.

Алгоритм работы системы на канальном уровне отображен на диаграмме последовательностей (рис. 3)

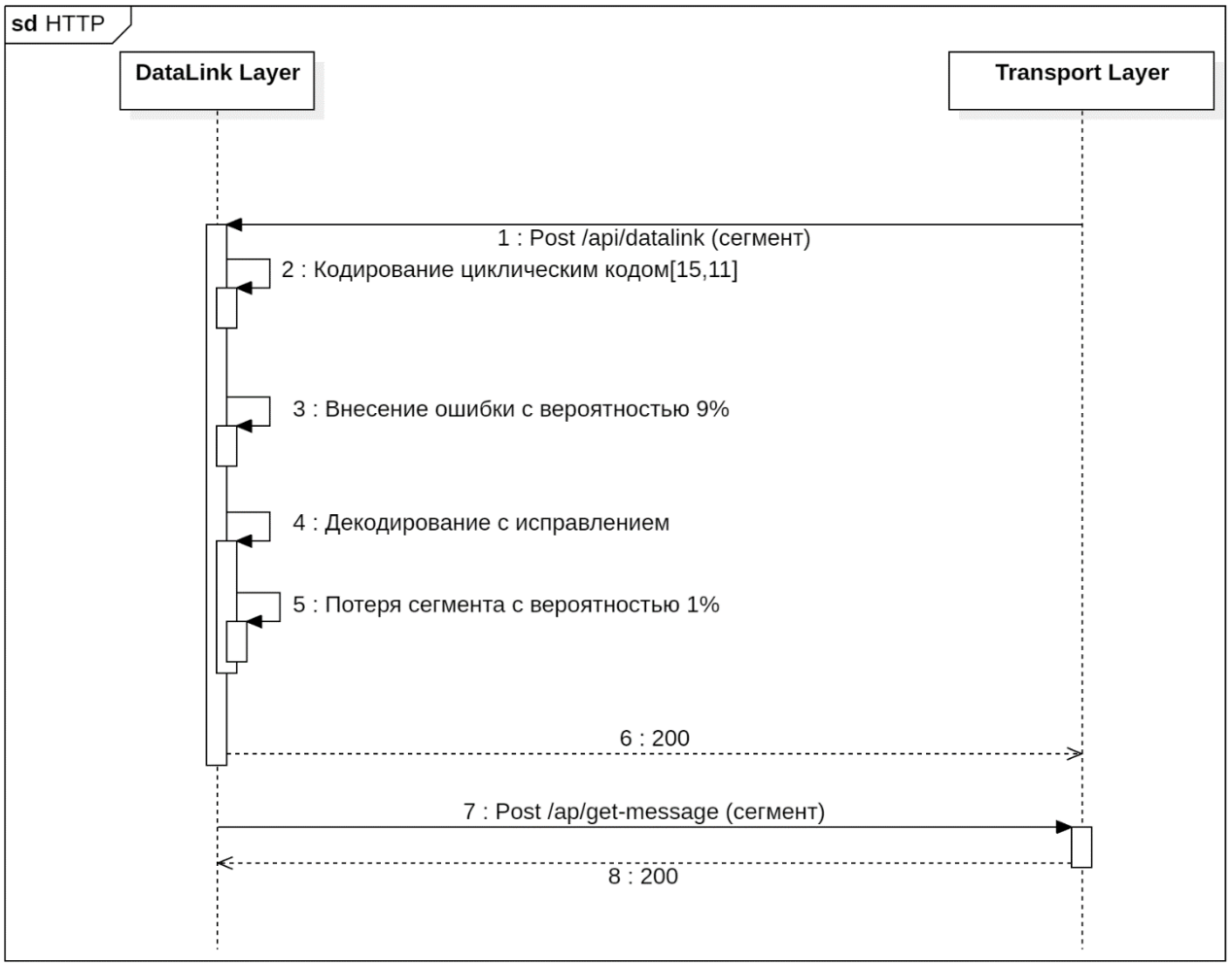

Рисунок 3 – Диаграмма последовательности канального уровня

Канальный уровень получает сегмент 100 байт от транспортного уровня, после чего кодирует сегмент циклическим кодом[15;11] и вносит ошибку с вероятностью 9% в случайный бит. Далее происходит декодирование сегмента и исправление однократной ошибки, в случае если она была допущена. С вероятностью 1% сегмент теряется на канальном уровне и, если он не потерялся, отправляется транспортному уровню с помощью метода get-message.

Методы, используемые на канальном уровне (Таблица 2):

Таблица 2 – Методы канального уровня

Метод	Входные параметры	Описание
POST /api/datalink	timestamp timestamp amountOfSegments integer sender string segmentNum integer message string	Полученный от транспортного уровня json сегмента кодируется циклическим кодом [15, 11] в битовый формат. Вносится ошибка в 1 бит сегмента. Далее сегмент декодируется либо с потерей пакета с ошибкой, либо с исправлением ошибки. Затем сегмент передается на транспортный уровень.

ПРИКЛАДНОЙ УРОВЕНЬ

Прикладной уровень предназначен для отправки и приема данных. Он также является инструментом взаимодействия пользователя с системой и отображения всех данных, приходящих в реальном времени. При подключении вводится имя пользователя и выполняется установка WebSocket соединения. Чат является общим и не хранит историю сообщений. Если сообщение доставляется с признаком ошибки, то файл игнорируется, а чате отображается значок ошибки.

Для реализации пользовательского интерфейса прикладного уровня используется React с менеджером состояний Redux Toolkit и с библиотекой для запросов Axios. В качестве UIkit взят MUI, а за основу дизайна выбран Cian. Для сервера используется Node.js с фрейворком Express.js.

WebSocket - это протокол связи, предназначенный для обмена данными между браузером и веб-сервером в реальном времени. Он позволяет устанавливать постоянное соединение между сервером и клиентом, через которое могут передаваться двусторонние потоки данных. Главное отличие WebSocket от HTTP-запросов заключается в том, что WebSocket поддерживает постоянное соединение без необходимости постоянно отправлять запросы на сервер для обновления данных.

Принципы работы WebSocket:

1. Установка соединения по протоколу TCP: Клиент и сервер инициируют процесс установки надежного соединения на основе протокола TCP, обеспечивающего стабильную связь между ними.
2. Переход к протоколу WebSocket: После успешной установки соединения происходит автоматический переход с протокола HTTP на WebSocket, что позволяет использовать более эффективные механизмы обмена данными.
3. Поддержание постоянного соединения: WebSocket обеспечивает постоянное соединение между клиентом и сервером, что позволяет

передавать данные в режиме реального времени без необходимости частых запросов на установку связи.

4. Двусторонняя передача данных: Обе стороны, клиент и сервер, имеют возможность инициировать передачу данных без ожидания запроса от другой стороны, что обеспечивает более гибкую и интерактивную связь.
5. Поддержка различных типов данных: WebSocket позволяет передавать как текстовые, так и бинарные данные, что делает его универсальным средством для обмена информацией в режиме реального времени между приложением и сервером.

Алгоритм работы системы отображен на диаграмме последовательности (рис. 4).

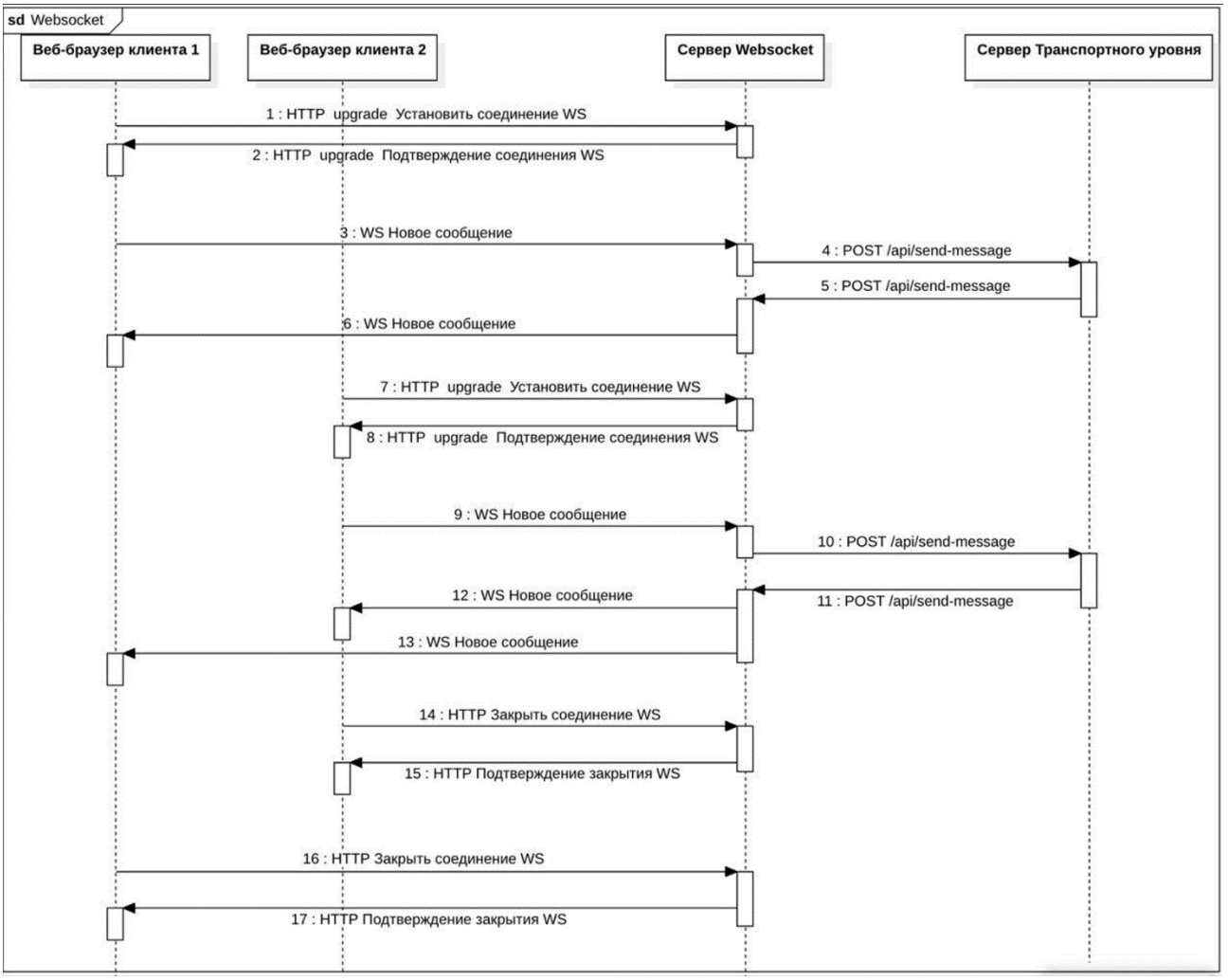

Рисунок 4 – Диаграмма последовательности прикладного уровня

Таблица 3 – Методы прикладного уровня

Метод	Входные параметры	Описание
POST /api/send-messsage	message string sender string time timestamp error boolean	Получение сообщения с транспортного уровня

В начале при подключении к системе происходит установка WebSocket соединения. Для этого пользователь отправляет через графический интерфейс запрос, передавая в нем свое имя. Далее пользователь получает подтверждение соединения WebSocket и перейдет на страницу чата. Затем пользователь вводит сообщение и, нажимая на кнопку «Отправить», отправляет запрос на отправку сообщения, содержащий имя отправителя, время отправки и текст, на транспортный уровень. Далее при отсутствии ошибки участники чата видят сообщение. Если сообщение пришло с признаком ошибки, то отправитель видит значок ошибки и сообщение об ошибке, а получатель не видит ничего. При нажатии на кнопки «Выйти» закрывается WebSocket соединение, сбрасывается имя пользователя и очищается чат.

При запуске приложения пользователь видит главную страницу (рис. 5).

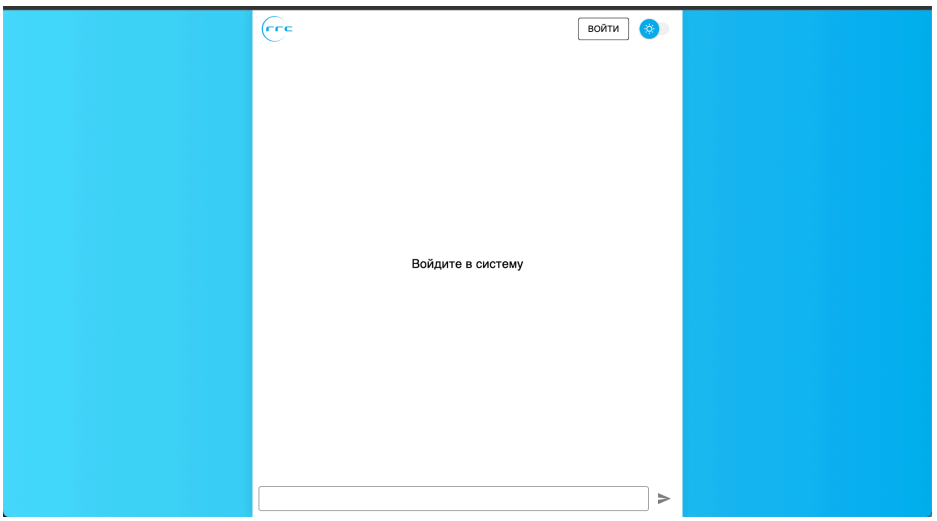

Рисунок 5 – Главная страница

Пользователь нажимает на кнопку «Войти», вводит свое имя, подтверждает. После этого открывается WebSocket соединение, пользователю становится доступным интрефейс (рис. 6).

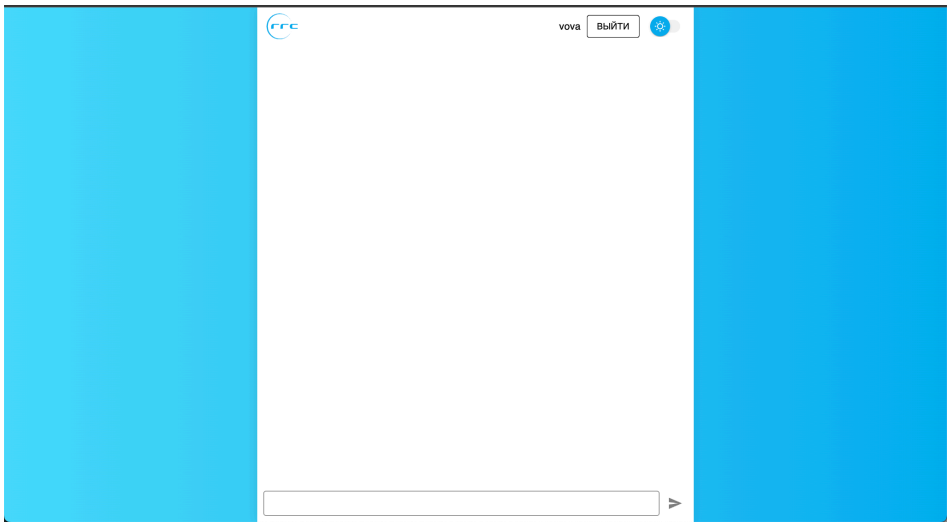

Рисунок 6 – Страница чата

Пользователь вводит в поле ввода сообщение и, нажимая на кнопку «Отправить», отправляет его. При отсутствии ошибки пользователи чата видят свое имя, текст сообщения и время отправки (рис. 7).

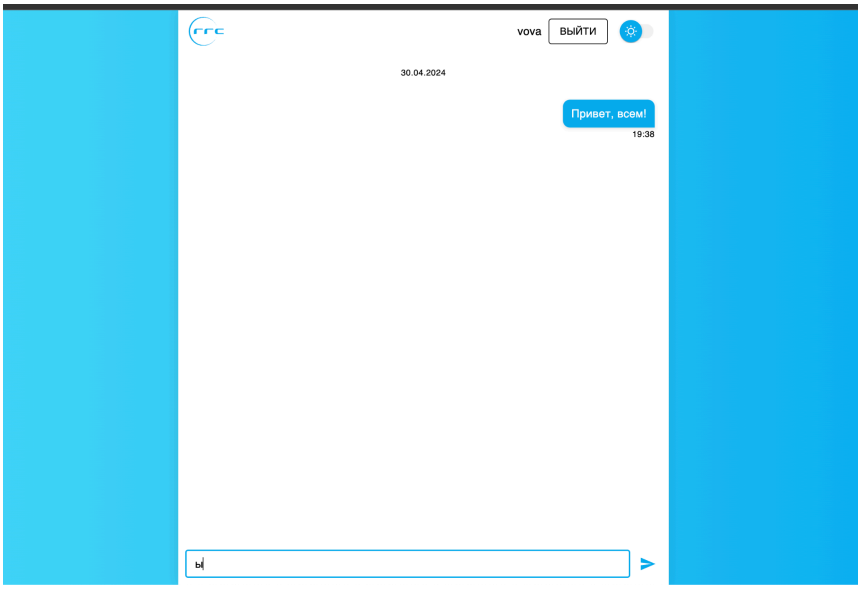

Рисунок 7 – Отправление сообщения без признака ошибки

Если сообщение пришло с признаком ошибки, отправитель видит значок ошибки и сообщение об ошибке (рис. 8).

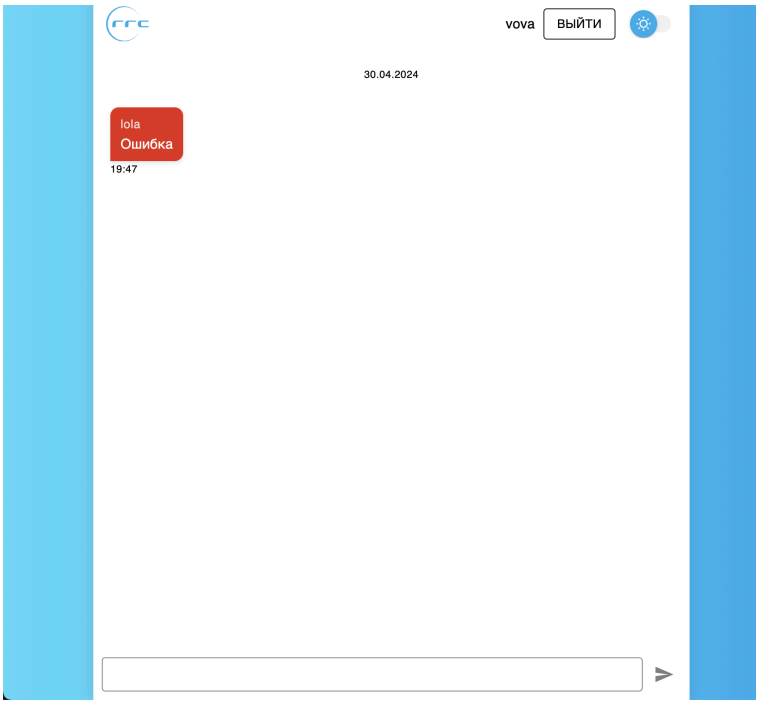

Рисунок 8 – Отправление сообщения с признаком ошибкой

При получении сообщения получатель видит имя отправителя, текст сообщения и время отправки (рис. 9).

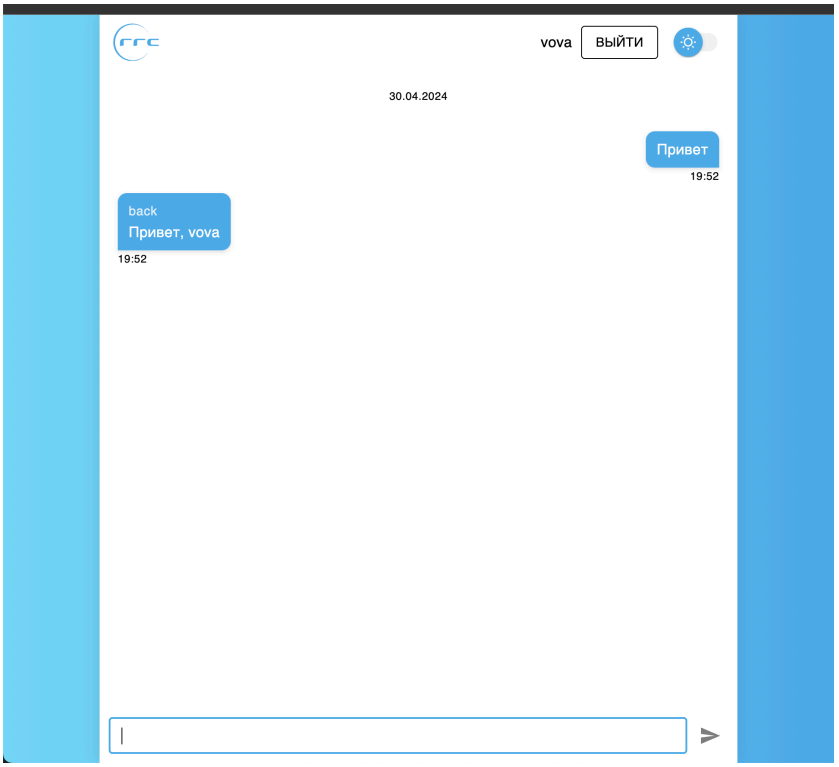

Рисунок 9 – Получение сообщения

При нажатии на кнопку «Выйти» происходит закрытие WebSocket соединения, сброс имени и очистка чата. Пользователь возвращается на стартовую страницу.

Заключение

В ходе работы были достигнуты следующие результаты:

1. Разработан дизайн приложения в Figma. Ссылка: https://www.figma.com/file/8HU3clsbPo91sJuR37hfzZ/Chat?type=design&node-id=0%3A1&mode=design&t=bWBTH5iLyIFByUH4-1
2. Был разработан сервис канального уровня на Go.
3. Был разработан сервис транспортного уровня на Java.
4. Разработан интерфейс приложения с использованием технологии React.
5. Подготовлен набор документации, включающий РПЗ, ТЗ, ПМИ, РСА, РП.
6. Исходный код проекта доступен в GitHub:

Прикладной уровень: https://github.com/CAPVOK/Networking_chat_front

Канальный уровень: https://github.com/lud0m4n/Network

Транспортный уровень: https://github.com/Sh1bari/transport_layer

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Java: [Электронный ресурс]. // URL: https://www.w3schools.com/java/ (Дата обращения: 7.04.2024)
2. Официальная документация React: [Электронный ресурс]. // URL: https://legacy.reactjs.org/docs/getting-started.html (Дата обращения: 15.04.2024)
3. Официальная документация MUI: [Электронный ресурс]. // URL: https://mui.com/material-ui/getting-started/overview/ (Дата обращения: 16.04.2024)
4. Руководство по Go: [Электронный ресурс] // Go Tour. URL: https://go.dev/tour (дата обращения: 11.04.2024)
5. Официальная документация Apache Kafka: [Электронный ресурс] // URL: https://kafka.apache.org/documentation/ (дата обращения: 11.04.2024)

ПРИЛОЖЕНИЕ 1

Техническое задание

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 «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Сетевые технологии в АСОИУ»

Техническое задание

Вариант 10

Студенты группы РТ5-61Б:

Кабанец В.М.

Краснов В.Б.

Фруктин А.Е.

2024г.

1. Наименование:

Распределённая система, представляющая собой веб приложение для отправки текстовых сообщений в реальном времени.

2. Основание для разработки:

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

3. Исполнители:

Кабанец В.М. (прикладной уровень) – группа РТ5-61Б

Краснов В.Б. (транспортный уровень) – группа РТ5-61Б

Фруктин А.Е. (канальный уровень) – группа РТ5-61Б

4. Цель разработки:

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

5. Функциональные требования

5.1. Прикладной уровень:

5.1.1. Страница приложения с окном чата для отправки и просмотра полученных текстовых сообщений с указанием отправителя и времени отправки;

5.1.1.1. При подключении к чату пользователь должен ввести имя с помощью кнопки Войти, которое будет передаваться с каждым новым сообщением;

5.1.1.2. После входа появляется возможность отправки сообщений по установленному WebSocket соединению, используя кнопки “Отправить” (иконка “Самолет”);

5.1.1.3. По кнопке “Выйти” чат и логин очищаются, а ws-подключение закрывается;

5.1.1.4. В случае, если сообщение пришло с признаком ошибки, сообщение не отображается, а вместо него у получателей появляется соответствующее сообщение;

5.1.1.5. Дизайн приложения соответствует сайту компании “RRC”;

5.1.2. WebSocket-сервер

5.1.2.1. Хранит имена пользователей для всех ws-подключений;

5.1.2.2. Позволяет устанавливать, закрывать ws-соединения, получать сообщения от клиентов и широковещательно рассылать сообщения подключенным клиентам;

5.1.3. Реализация HTTP-метода send-message для получения сообщения с транспортного уровня:

5.1.3.1. В json каждого сообщения указывается отправитель, время отправки, признак ошибки и полезная нагрузка;

5.1.3.2. Полученное по HTTP сообщение отправляется широковещательной WebSocket рассылкой всем подключенным ws-клиентам, кроме тех, у кого логин совпадает с именем отправителя;

5.2. Транспортный уровень:

5.2.1. Реализация HTTP-метода send-message для сегментирования текстовых сообщений:

5.2.1.1. Разбиение сообщения на сегменты по 100 байт и их поочередная отправка на канальный уровень через метод Code;

5.2.1.2. Каждый сегмент содержит время отправки (в качестве идентификатора сообщения), имя отправителя, номер данного сегмента в сообщении, общее число сегментов, текст сообщения;

5.2.2. Реализация HTTP-метода get-message для передачи сообщения на прикладной уровень:

5.2.2.1. При получении с канального уровня сегменты помещаются в очередь, раз в 1 секунду собираются в единое сообщение и передаются на прикладной уровень;

5.2.2.2. Если часть из сегментов сообщения не была принята, оно передается на прикладной уровень с признаком ошибки;

5.3. Канальный уровень:

5.3.1. Сервис канального уровня эмулирует канал связи с потерями:

5.3.1.1. Сервис должен вносить ошибку с вероятностью 9% в один случайный бит каждого сформированного кадра;

5.3.1.2. Сервис должен терять передаваемый кадр с вероятностью 1%;

5.3.2. Реализация HTTP-метода datalink для кодирования и декодирования полученного от транспортного уровня сегмента:

5.3.2.1. Полученный от транспортного уровня json сегмента кодируется циклическим кодом [15,11] для получения кадра;

5.3.2.2. После внесения ошибки в кадр он декодируется с исправлением ошибки и передается далее в виде сегмента на транспортный уровень;

6. Требования к составу технических средств:

6.1. Прикладной уровень:

6.1.1. Серверная часть

6.1.1.1. ПК с ОС Windows(7.0 и выше)

6.1.1.2. Node.js (1.20 и выше)

6.1.2. Клиентская часть

6.1.2.1. ПК с ОС Windows(7.0 и выше)

6.1.2.2. Веб-браузер: Chrome(40 и выше)

6.2. Транспортный уровень:

6.2.1.1. ПК с ОС Windows(7.0 и выше)

6.2.1.2. Java Spring Boot (3.2 и выше)

6.2.1.3. Kafka (2.12 и выше)

6.3. Канальный уровень:

6.3.1.1. ПК с ОС Windows(7.0 и выше)

6.3.1.2. Golang (1.21.7 и выше)

7. Этапы разработки:

7.1. Выбрать тему-вариант, определить команду и разработать ТЗ – 3 неделя;

7.2. Разработать макет figma, три диаграммы последовательности и описать HTTP-методы в swagger – 8 неделя;

7.3. Разработать и отладить приложение, подготовить полный комплект документов – 12 неделя;

7.4. Исправить замечания, защитить проект – 14 неделя.

8. 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

9. Порядок приемки работы:

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

Работа защищается перед комиссией преподавателей кафедры.

10. Дополнительные условия:

Данное Техническое Задание может дополняться и изменяться в установленном порядке.

ПРИЛОЖЕНИЕ 2

Программа и методика испытаний

ПРИЛОЖЕНИЕ Б ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

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 «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Сетевые технологии в АСОИУ»

Программа и методика испытаний

Вариант 10

Студенты группы РТ5-61Б:

Кабанец В.М.

Краснов В.Б.

Фруктин А.Е

2024г.

1. Объект испытаний:

Объектом испытания является распределенная информационная система обмена сообщениями в реальном времени.

2. Цель испытаний:

Целью проведения испытаний является доказательство работоспособности описанного в пункте 1 объекта испытаний.

3. Требования к объекту испытаний:

Требования к объекту испытаний представлены в документе «Техническое задание».

4. Требования к программной документации:

Во время проведения испытания должны быть представлены следующие документы:

1) Техническое задание;
2) Программа и методика испытаний.

5. Программа испытаний:

№	Номер пункта ТЗ	Выполняемое действие	Результат
1	5.1.1.1	Запуск приложения.	Приложение запущено. Открывается главная страница приложения с заблокированным функционалом.
2	5.1.2.1	Нажатие на кнопку «Войти», заполнение поля имени, подтверждение.	В правом верхнем углу отображается имя пользователя,

			функционал становится доступным.
3	5.1.2.1	Отправка сообщений. Ввод сообщения в специальное поле, нажатие на кнопку «Отправить».	Отображение имени пользователя голубым цветом, текста сообщения и времени отправки сообщения справа.
4	5.1.2.2	Получение сообщения от другого пользователя.	Отображение имени отправителя, текста сообщения и времени отправки сообщения слева.
5	5.1.2.3	Выход из чата. Нажатие на кнопку «Выйти».	Осуществляется блокировка функционала, очистка чата.
6	5.1.2.4	Отправка сообщения с ошибкой. Ввод сообщения в специальное поле, нажатие на кнопку «Отправить».	Отображение сообщения красного цвета с текстом «Ошибка».
7	5.1.3	Запоминание имени для WebSocket подключения. Ввод имени и нажатие на кнопку «Войти».	Открытие WebSocket соединения, запоминание имени пользователя.
8	5.1.4	Метод send-message.	В json сообщения указывается отправитель, текст сообщения, время отправки, признак ошибки. Сообщение отправляется всем участникам чата.
9	5.2.1	Метод send-message. Ввод сообщения, нажатие на кнопку «Отправить».	На транспортном уровне сообщение разбивается на сегменты по 100 байт и

			посегментно отправляется на канальный уровень. В сегменте содержатся время отправки, общая длина сообщения, номер данного сегмента, полезная нагрузка.
10	5.2.2	Метод get-message.	На транспортном уровне формируется очередь из полученнных сегментов. Сегменты собираются в сообщение раз в 1 секунду. Если часть сегментов не была принята, то сообщение передается на прикладной уровень с признаком ошибки.
11	5.3.1	Метод datalink.	Полученный от транспортного уровня json сегмента кодируется циклическим кодом [15, 11] в битовый формат. Вносится ошибка в 1 бит сегмента. Далее сегмент декодируется с исправлением ошибки. Затем сегмент передается на транспортный уровень.

ПРИЛОЖЕНИЕ 3

Руководство пользователя

ПРИЛОЖЕНИЕ В РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

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 «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Сетевые технологии в АСОИУ»

Руководство пользователя

Вариант 10

Студенты группы РТ5-61Б:

Кабанец В.М.

Краснов В.Б.

Фруктин А.Е.

2024г.

Введение

1.1. Область применения

Требования настоящего документа применяются при:

- предварительных комплексных испытаниях;
- опытной эксплуатации;
- приемочных испытаниях;
- промышленной эксплуатации.

1.2. Краткое описание возможностей

Распределённая система обмена сообщениями, представляющая собой чат для обмена сообщения между пользователями в реальном времени.

Распределенная система предоставляет возможность доступа к чату всем людям, которые перешли по ссылке и ввели свое имя в строку идентификации. При успешной идентификации пользователь получает возможность читать сообщения отправленные другими пользователями, начиная с того времени как он присоединился к чату. Также он может и сам писать и отправлять сообщения тем пользователям, которые находятся вместе с ним в чате.

Актуальность переписки поддерживается при помощи протокола WebSocket, который позволяет обновлять окно чата с сообщениями в реальном времени.

2. Назначение и условия применения распределенной системы.

2.1. Назначение распределенной системы.

Распределенная система предназначена для возможности вести переписку нескольким пользователям в реальном времени.

Для использования Распределенной системы необходимо выполнение следующих условий:

2.2. Системные требования

Для работы клиента необходим Yandex Browser или Google Chrome.

3. Условия выполнения программы

Для работы программы требуется браузер Google Chrome или любой иной поддерживающий современные функции JavaScript, а также стабильное интернет-соединение.

4. Выполнение программы

4.1. Инсталляция/деинсталляция

Потребуется Web-браузер, рекомендуется, Yandex Browser или Google Chrome.

4.2. Запуск программы

На компьютере необходимо запустить браузер и ввести в адресную строку http://localhost:5173/.

5. Описание операций

5.1. Идентификация в системе

Доступно для: все пользователи.

Операция: открыть веб страницу.

Для идентификации в системе необходимо открыть главную страницу, нажать «Войти», ввести свое имя и подтвердить (рис.1).

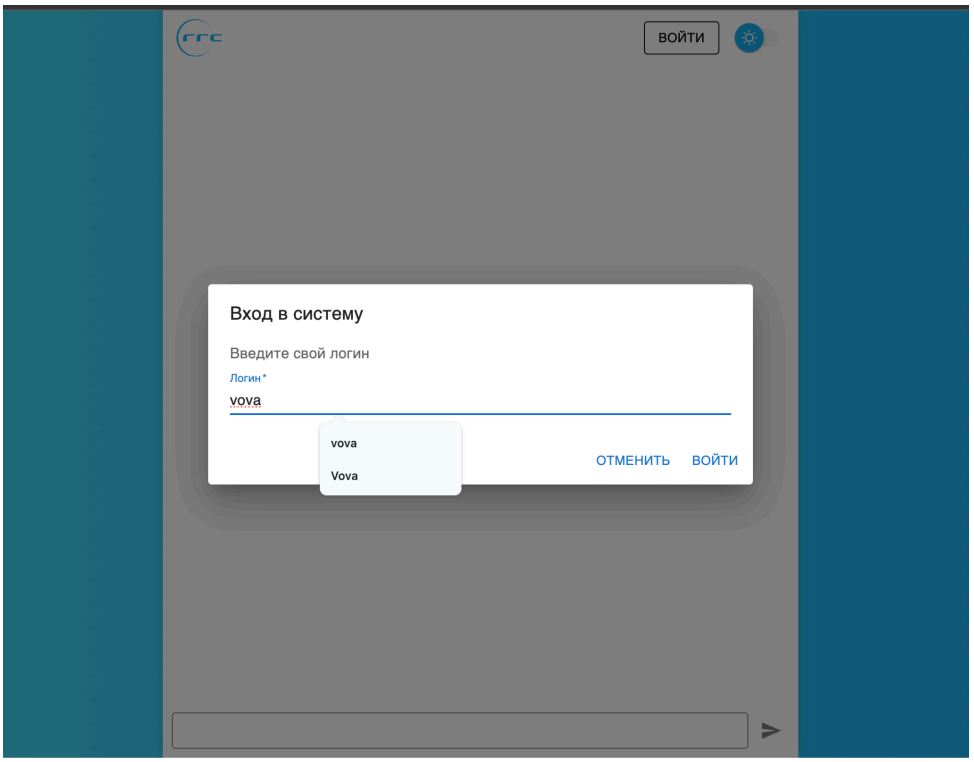

Рисунок 1 – окно идентификации

5.2. Просмотр и отправка сообщений

Доступно для: идентифицированным пользователям.

Операция 1: для отправки сообщения нужно вести его в строку для сообщений и нажать на значок «отправить» (рис.2).

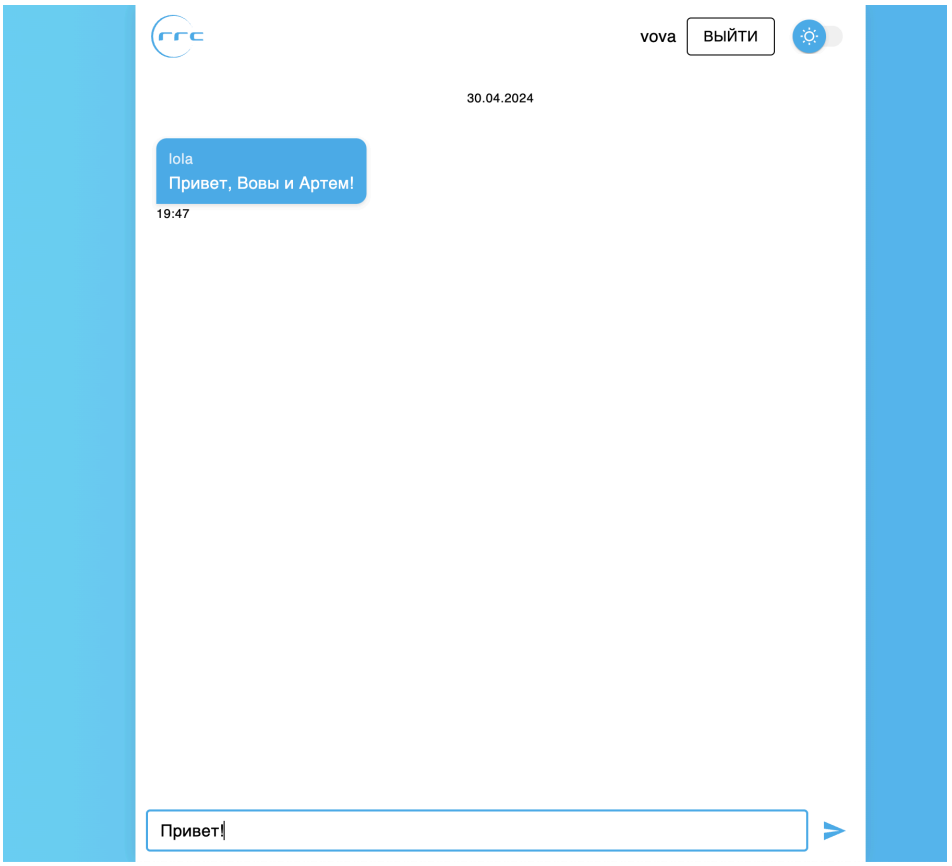

Рисунок 2 – отправка сообщения

Операция 2: для получения сообщений от других пользователей нужно находиться на главной странице (рис.3) и ждать сообщения.

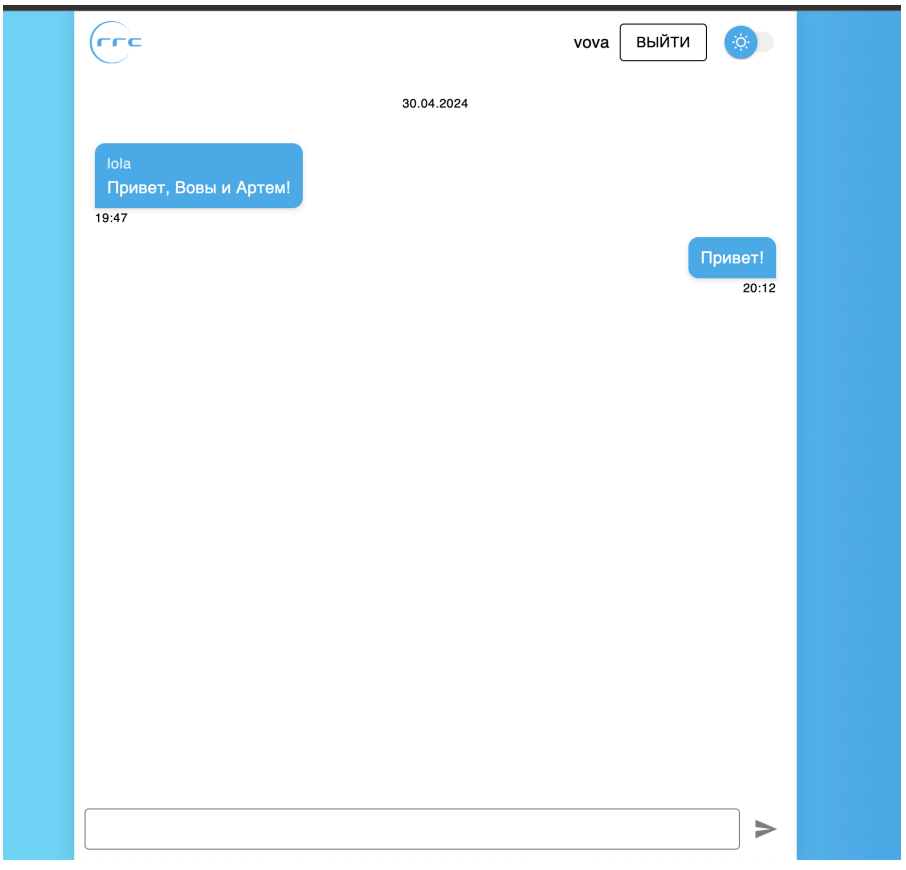

Рисунок 3 – окно чата

5.3. Выход из чата

Доступно для: идентифицированным пользователям.

Операция: для выхода из чата необходимо нажать кнопку «выйти» (рис. 4) в правом верхнем углу страницы чата.

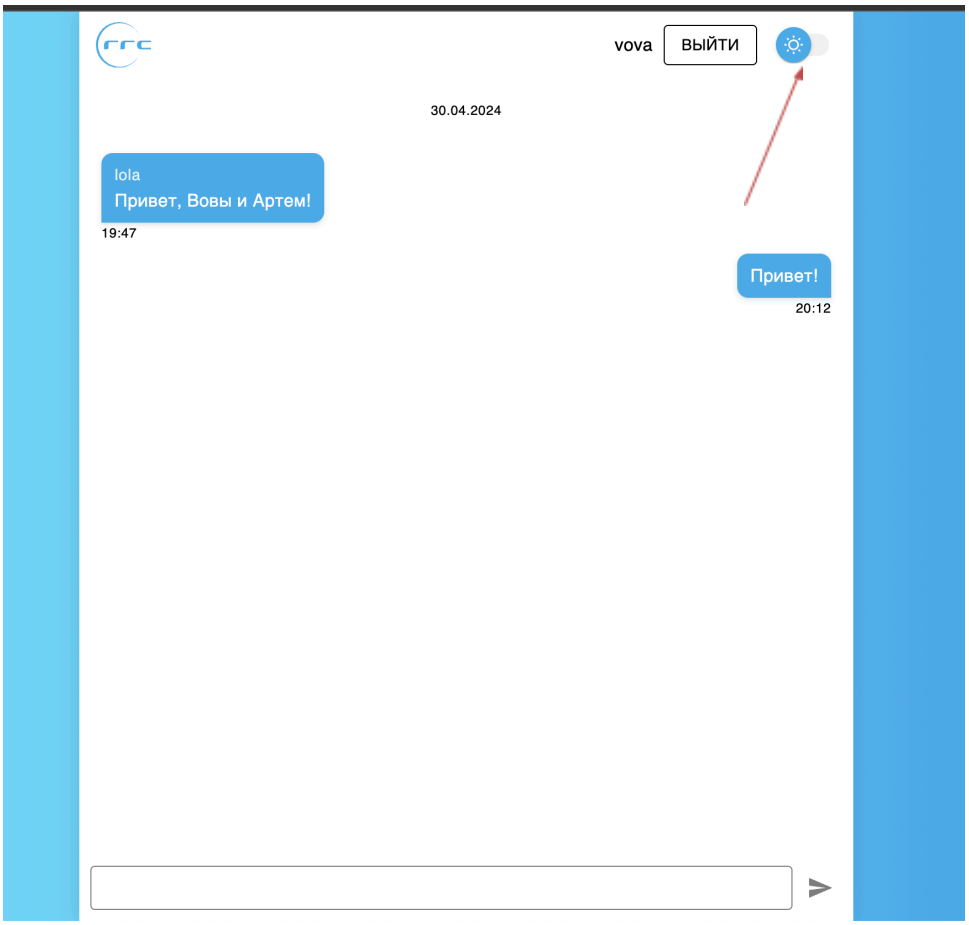

Рисунок 4 – кнопка «выйти»

ПРИЛОЖЕНИЕ 4

Руководство системного администратора

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 «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Сетевые технологии в АСОИУ»

Руководство системного администратора

Вариант 10

Студенты группы РТ5-61Б:

Кабанец В.М.

Краснов В.Б.

Фруктин А.Е.

2024г.

1. Системные требования

Для работы клиента необходим Yandex Browser или Google Chrome.

1.1. Требования к ОС

– Windows

Windows 7, Windows 8, Windows 8.1, Windows 10 или более поздней версии.

Примечание: также поддерживаются следующие серверные ОС: Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2 или Windows Server 2016.

– MacOS

macOS High Sierra 10.13 или более поздней версии

– Linux

Ubuntu 18.04 (64-разрядная версия) или более поздней версии, Debian 10 или более поздней версии, openSUSE 15.2 или более поздней версии, Fedora Linux 32 или более поздней версии.

Требования к оборудованию

Центральный процессор: с частотой от 1 ГГц

Оперативная память: 8 Гб и выше

Пространство на жестком диске: 1 Гб

Платформа: Microsoft Windows, Mac OS, Linux x32/x64

1.2. Требования к ПО

1.2.1. Docker desktop версии 4.17.1;

1.2.2. Golang версии 1.21.0 и выше;

1.2.3. Java Spring Boot 3.2 и выше;

1.2.4. Node.js версии 1.20 и выше;

1.2.5. Kafka версии 2.12 и выше.

2. Порядок развёртывания серверов:

2.1. Сервис канального уровня

2.1.1. Выполнить команду docker pull lud0m4n/network_datalink:1;

2.1.2. Выполнить команду docker run -d -p 8085:8081 --name datalinkback lud0m4n/network_datalink:1;

После этого на порте 8085 будет доступен сервис канального уровня.

2.2. Сервис транспортного уровня

2.2.1. Выполнить команду docker compose up -d –build;

2.2.2. Выполнить команду mvn spring-boot:run;

После этого на порте 8082 будет доступен сервис транспортного уровня.

2.3. Сервис фронтенда прикладного уровня

2.3.1. Выполнить команду npm i;

2.3.2. Выполнить команду npm run dev.

После этого на порте 5173 будет доступен сервис фронтенда прикладного уровня.

2.4. Сервис бекенда прикладного уровня

2.4.1. Выполнить команду npm i;

2.4.2. Выполнить команду npm run dev.

После этого на порте 5001 будет доступен сервис фронтенда.