|  |  |
| --- | --- |
|  | **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 СИСТЕМА УПРАВЛЕНИЯ |
| КАФЕДРА | СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ |

**КУРСОВАЯ РАБОТА**

***НА ТЕМУ:***

|  |
| --- |
| ***Создание распределенной информационной*** |
| ***системы обмена текстовыми сообщениями*** |
|  |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ5-63Б |  |  |  | **С.В. Валова** |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |
| Студент | ИУ5-63Б |  |  |  | **В.Д. Рыбин** |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |
| Студент | ИУ5-63Б |  |  |  | **И.И. Лобанов** |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель курсового проекта |  |  |  |  | **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студенты группы ИУ5-63 Валова С.В., Лобанов И.И., Рыбин В.Д. \_

(Фамилия, имя, отчество)

Тема курсовой работы Создание распределенной информационной системы обмена\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_текстовыми сообщениями \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_УЧЕБНАЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

***Задание*** \_\_ Разработать систему обмена текстовыми сообщениями в реальном времени, состоящую из трех уровней: прикладной, транспортный и канальный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

***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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ С.В. Валова \_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ В.Д. Рыбин \_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ И.И. Лобанов \_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

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

**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)**

**КАЛЕНДАРНЫЙ ПЛАН**

**на выполнение курсовой работы**

|  |  |  |
| --- | --- | --- |
| по дисциплине | Сетевые технологии в АСОИУ | |
| Студенты группы | ИУ5-63Б Валова С.В., Рыбин В.Д., Лобанов И.И. | |
|  | (Фамилия, имя, отчество) | |
| Тема курсовой работы | |  |
| Создание распределенной информационной системы обмена текстовыми сообщениями | | |
|  | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Руководитель КР** | **Куратор** |
| 1. | Выбор темы; формирование команды и ТЗ | *\_24.02.24\_*  *Планируемая дата* | *24.02.24* |  |  |
| 2. | Создание макета Figma, диаграммы последовательности и формирование swagger | *30.03.24\_*  *Планируемая дата* | *30.03.24* |  |  |
| 3. | Оформление РПЗ, ПМИ, РСА, РП | *28.04.24\_*  *Планируемая дата* | *14.05.24* |  |  |
| 4. | Тестирование распределенной системы и подготовка проекта к демонстрации проектапрезеннеобходимости) | *22.05.24\_*  *Планируемая дата* | *22.05.24* |  |  |
| 5. | Защита курсовой работы | *27.05.24\_*  *Планируемая дата* | *27.05.24* |  |  |

Студент *\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руководитель работы *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись, дата) (подпись, дата)

Студент *\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Студент *\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

# **СОДЕРЖАНИЕ**

[**СОДЕРЖАНИЕ** 4](#_Toc166676994)

[**ВВЕДЕНИЕ** 5](#_Toc166676995)

[**СЕРВИС ПРИКЛАДНОГО УРОВНЯ** 7](#_Toc166676996)

[**СЕРВИС ТРАНСПОРТНОГО УРОВНЯ** 11](#_Toc166676997)

[**СЕРВИС КАНАЛЬНОГО УРОВНЯ** 14](#_Toc166676998)

[**ЗАКЛЮЧЕНИЕ** 15](#_Toc166676999)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 16](#_Toc166677000)

[**Приложение 1. Техническое задание** 17](#_Toc166677001)

[**Приложение 2. Программа и методика испытаний** 24](#_Toc166677002)

[**Приложение 3. Руководство пользователя** 28](#_Toc166677003)

[**Приложение 4. Руководство системного администратора** 33](#_Toc166677004)

# **ВВЕДЕНИЕ**

С развитием современных технологий и повсеместным использованием интернета, необходимость в эффективных и безопасных средствах обмена информацией становится все более актуальной. Согласно проведённому в августе 2023 года опросу от ВЦИОМ, 86% россиян, **пользующихся хотя бы одной социальной сетью или мессенджером, проводят в них время практически ежедневно [1].** Причем самым популярным контентом в социальных сетях являются новостные каналы, о чем говорят 77% опрошенных. Это указывает на рост актуальности социальных сетей не только как средство коммуникации, но и как способ получения информации об актуальных событиях.

Платформы, такие как WhatsApp, Telegram, и другие, обеспечивают мгновенный обмен сообщениями и файлами, что делает их незаменимыми инструментами для коммуникации как в личных, так и в рабочих целях. Исследование, проведённое сервисом по поиску работы Superjob в феврале 2023 года, показывает, что лишь 8% организаций не использует мессенджеры для коммуникации между сотрудниками [2]. Остальные так или иначе пользуются либо общедоступными, либо корпоративными мессенджерами. Использование мессенджеров способствует улучшению производительности и ускоряет процессы обмена информацией. Таким образом, разработка новых и усовершенствование существующих мессенджеров играют ключевую роль в обеспечении эффективного обмена информацией и поддержании связи между людьми в современном информационном обществе России.

Целью данного проекта является разработка и реализация распределённой информационной системы, которая позволит пользователям обмениваться файлами мгновенно и безопасно в реальном времени. Система разделена на три уровня: прикладной, транспортный и канальный. Прикладной уровень необходим для реализации удобного и понятного пользовательского интерфейса. Целью канального уровня является обработка переданных данных и имитация взаимодействия с удалённым сетевым узлом через канал с помехами. То есть он занимается кодированием и декодированием сигналов, внесением ошибки в процессе доставки по каналу, а также исправлением ошибки при отправке сообщения обратно, что является необходимым для обеспечения должного уровня надежности при работе мессенджера. Транспортный, в свою очередь, обеспечивает связь между прикладным и канальным уровнями. Он занимается передачей данных, а также проверкой целостности данных. Актуальность разработки данной системы подчеркивается повседневными потребностями пользователей в эффективном обмене файлами и сообщениями. Мессенджер может использоваться в самым различным способом. Например, как универсальный способ связи между преподавателем и студентами для уточнения деталей в процессе обучения или для передачи важной информации.

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, копирующий сайт одной из российских компаний.
2. Спроектировать систему и логику взаимодействия сервисов между собой.
3. Создать интерфейс, используя React.
4. Создать WebSocket-сервер на Express.js.
5. Создать сервис транспортного уровня на Golang 1.22.
6. Создать сервис канального уровня на Django 4.2.8.
7. Протестировать приложение при сервисах, развернутых на разных компьютерах.
8. Подготовить набор документации, включающий РПЗ, ТЗ, ПМИ, РП, РСА и набор диаграмм.

# **СЕРВИС ПРИКЛАДНОГО УРОВНЯ**

Прикладной уровень выполняет задачу получения текстового сообщения и передачи его на канальный уровень и наоборот. Когда пользователь попадает на основную страницу, то ему необходимо ввести логин, после этого он попадает на страницу чата, где можно увидеть сообщения, отправляемые пользователями в реальном времени. С помощью кнопки «Выйти» можно покинуть чат.

Фронтенд выполнен на языке TypeScript с использованием библиотеки React [3] — библиотека с открытым кодом для создания внешних пользовательских интерфейсов. React значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на компоненты – это может быть, как кнопка, так и целая страница. Для сборки и локальной разработки использовалась утилита vite - интерфейса командной строки (command line interface, CLI), который позволяет создавать предварительно настроенные React-проекты. Для управления состоянием в проекте используется библиотека redux-toolkit. Для отправки запросов на бэкенд используется библиотека Axios, предоставляющая простой в применении интерфейс для выполнения HTTP-запросов в браузере и на стороне сервера. Она развернута на стандарте Promise API и предназначена для работы с асинхронными запросами.

Так как чат должен отображать сообщения всем пользователям после его отправки, в проекте используется протокол WebSocket [4] – протокол для общения между клиентом и сервером, предоставляющий двухсторонне общение сверх протокола TCP. Клиент открывает WS-соединение один раз, а затем сервер может отдавать ответы тогда, когда посчитает нужным, в данном случае, когда придет новое сообщение, сервер отправит его всем открытым соединениям.

Работа прикладного сервиса показана на диаграмме последовательности ниже (Рисунок 1):

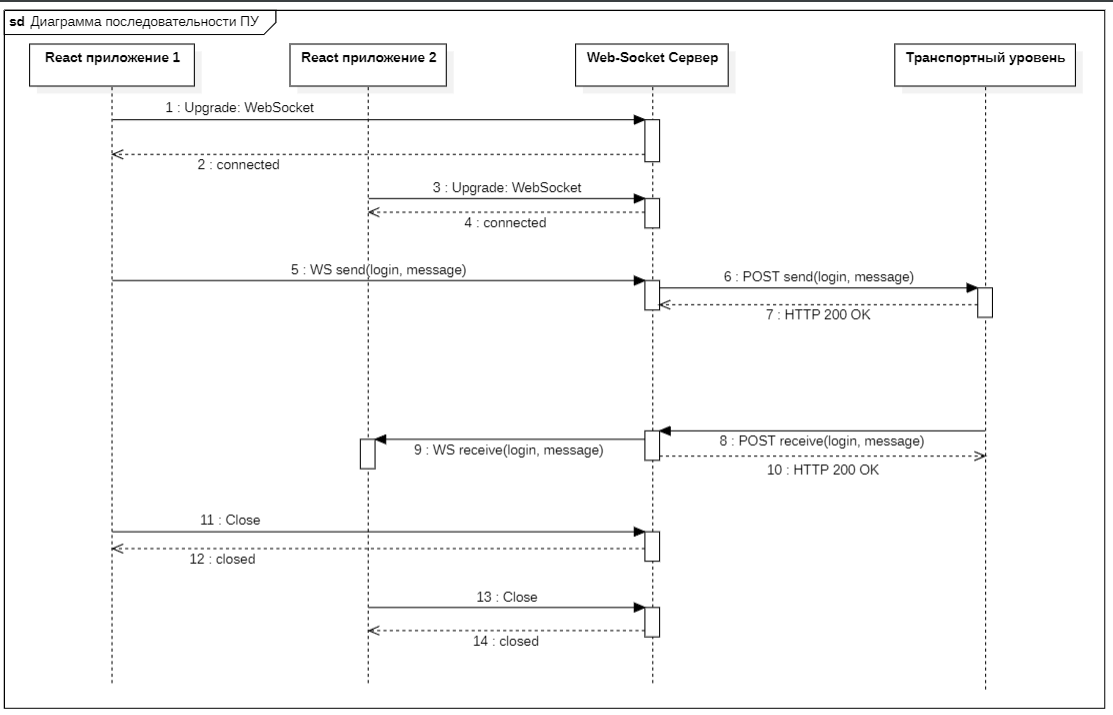


Рисунок 1 – Диаграмма последовательности прикладного уровня

При авторизации пользователь вводит login, который будет передаваться с каждым сообщением, на WebSocket-сервер отправляется запрос на установление соединения, оно устанавливается, и клиент получает ответ. Исходное HTTP-соединение заменяется на WebSocket-соединение. Пользователь отправляет сообщение, передавая свой логин и текст. Оно приходит на WebSocket-сервер, который передает его на транспортный уровень. Когда с транспортного уровня приходит сообщение, сервер отправляет его во все установленные соединения. Если сообщение пришло с признаком ошибки – оно не отображается, а в чате показывается текст о наличии ошибки.

При обновлении страницы история чата не сохраняется. С помощью кнопки «Выйти» можно отчистить чат и логин, отправляется запрос на закрытие соединения, если сервер подтверждает закрытие, то отправляет соответствующий ответ.

Методы, используемые на прикладном уровне (Таблица 1):

**Таблица 1 – Методы прикладного уровня**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** |
| 1 | POST | Получение сообщения с транспортного уровня | /receive | {  "login": string,  "message": string,  "error": boolean,  “date”: string,  } |

Ниже приведена первоначальная страница неавторизованного пользователя (Рисунок 2). Для присоединения к чату пользователю необходимо ввести в поле логин и нажать на кнопку «Войти», после он будет перенаправлен на страницу чата.

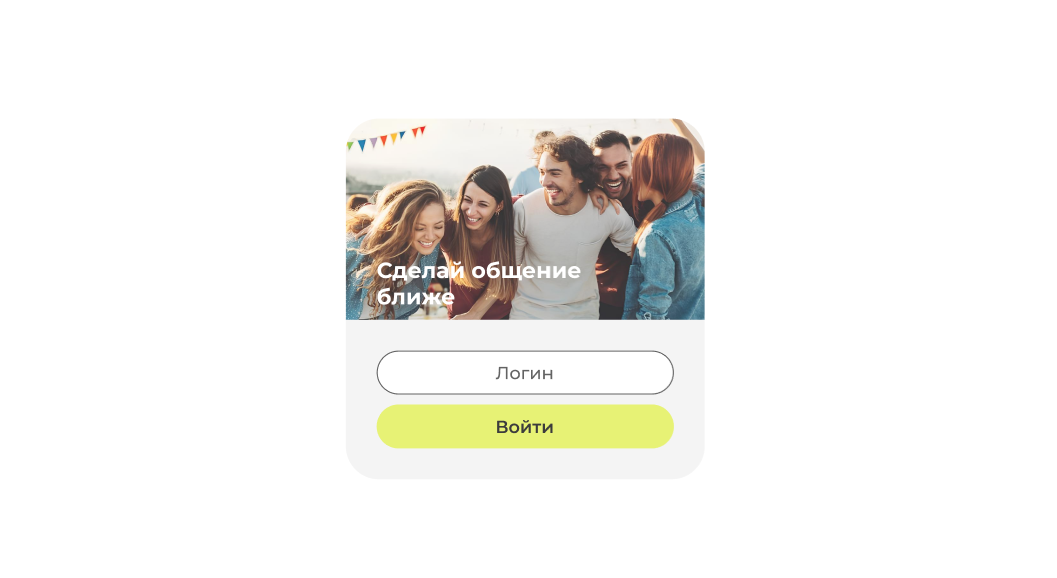


Рисунок 2 *–* Страница авторизации

Страница чата (Рисунок 3) для обмена текстовыми сообщениями в режиме реального времени отображает логин отправителя и текст сообщения, в случае наличия ошибки вместо сообщения будет текст о наличие ошибки.



Рисунок 3 *–* Страница обмена текстовыми сообщениями

С помощью кнопки «Выход» в правом верхнем углу можно очистить чат и логин (закрытие WebSocket соединения), чтобы потом сменить пользователя.

# **СЕРВИС ТРАНСПОРТНОГО УРОВНЯ**

Транспортный уровень в архитектуре мессенджера выполняет задачу по передаче сообщений между пользователями. Он разбивает сообщения на сегменты для транспортировки и после занимается их сборкой в единое целое с помощью относительного адреса пакета в сообщении. Когда данные приходят с канального уровня на транспортный, они временно сохраняются до тех пор, пока не будут собраны все части сообщения. Если какая-то часть не доходит в заданный временной интервал (при получении первого сегмента сообщения запускается таймер в 2 секунды, в течение которых собирается сообщение; в случае неудачи, есть еще 2 попытки), сообщение считается неполным, и система сообщает отправителю об ошибке.

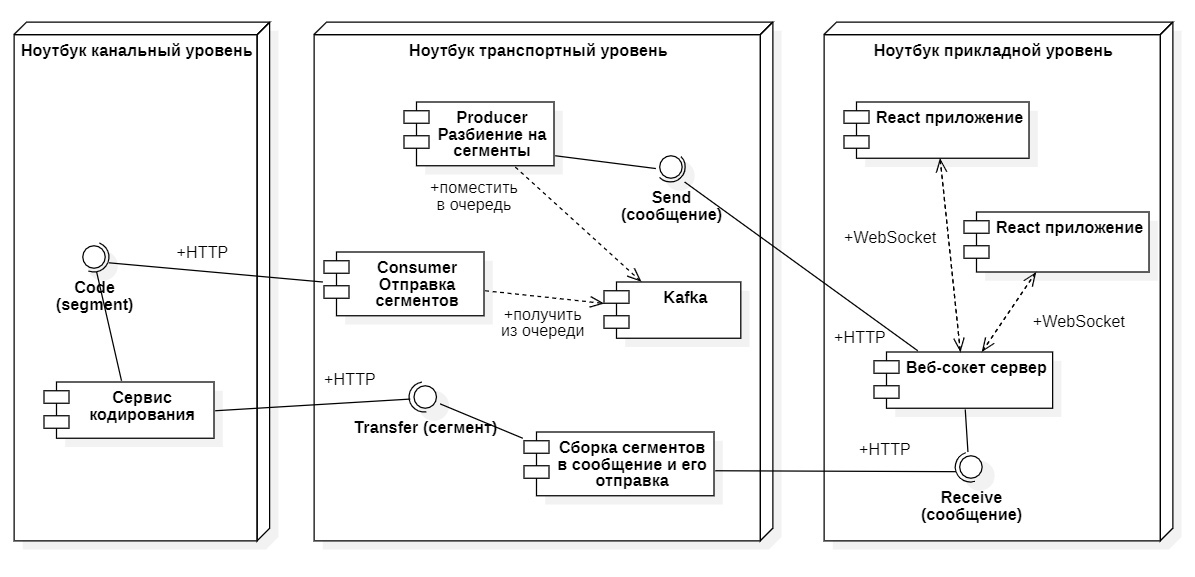
Сервера транспортного уровня написаны на Golang 1.22, используя для сетевых запросов стандартную библиотеку net/http. Временное хранение сегментов происходит в очереди в Kafka - распределенной системе обмена сообщениями между серверными приложениями в режиме реального времени. Благодаря высокой пропускной способности, масштабируемости и надежности применяется в компаниях, работающих с большими объемами данных. Взаимодействие с другими уровнями сервиса происходит при помощи REST API. Связь сервисов показана на диаграмме развёртывания:

Рисунок 4 – Диаграмма развертывания

Когда пользователь отправляет сообщение, оно в виде JSON объекта передаётся с прикладного уровня на сервер сегментации. Здесь сообщение разбивается на сегменты по 100 байт, которые после отправляются в топик Kafka, на который подписан Consumer. Сервер отправки сегментов, получая сегмент от брокера, добавляется к нему поля: время отправки, количество сегментов, номер данного сегмента, логин пользователя, а также идентификатор сообщения (используется UUID V4), что гарантирует принадлежность сегментов к их сообщению. Затем эти сегменты отправляются на канальный уровень. С канального уровня сегменты попадают на сервер сборки сообщения. В течение 2 секунд он проверяет, получены ли все сегменты одного сообщения, если да, то они объединяются и сообщение отправляется на прикладной уровень, если же какие-то сегменты были утеряны при передаче через канальный уровень, то на прикладной уровень отправляется сообщение об ошибке (Важно: на сборку сообщения дается 3 попытки, т.е. суммарно 6 секунд). Взаимодействие серверов транспортного уровня между собой и с другими частями системы отражено на диаграмме последовательности:

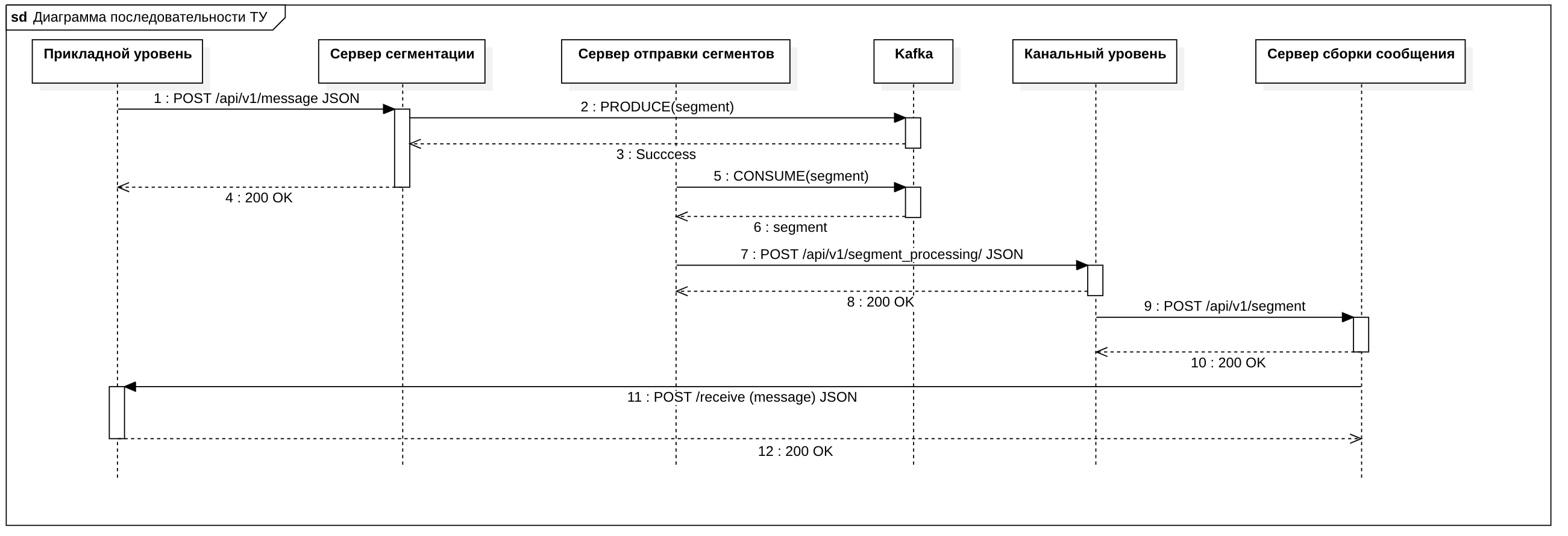


Рисунок 5 - Диаграмма последовательности транспортного уровня

**Методы, используемые на транспортном уровне (Таблица 2):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** |
| 1 | POST | Получение сообщения от прикладного уровня (WebSocket-сервера) | /api/v1/message | {  "login":string,  "message": string,  “date”: string,  } |
| 2 | POST | Получение сегмента от канального уровня | /api/v1/segment | {  “data”: []byte,  “segment\_id”: int,  “number\_of\_segments”: int,  “date\_send”: string,  “login”: string,  “message\_id”: string,  “error: boolean,  } |

Мы используем простейшую модель TCP Congestion Control - алгоритм, который управляет потоками данных с целью предотвращения перегрузки сетевых ресурсов. Если узлы сети не справляются с объемом передаваемой информации, это может привести к потере пакетов, увеличению задержек и снижению общей производительности сети. Алгоритм управляет размером TCP-окна и может 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 нашей курсовой работе мы не ограничиваем размер TCP-окна, но ограничиваем длину TCP-сегмента, передаваемого по каналу связи, а также смотрим на ошибки при передаче сегмента с канального уровня и ожидаем сегменты сообщения при сборке (3 раза делаем запрос в Kafka).

# **СЕРВИС КАНАЛЬНОГО УРОВНЯ**

Алгоритм работы канального уровня системы отображен на диаграмме последовательности (рис. 5). В основе системы лежит веб-сервис, реализующий внутри себя имитацию взаимодействия с удаленным сетевым узлом через канал с помехами. Для этого используется кодирование [7, 4]-кодом Хемминга [6]. Код Хемминга является самокорректирующемся и самокорректирующимся кодом, то есть кодом, позволяющим автоматически обнаруживать и исправлять ошибки при передаче данных.

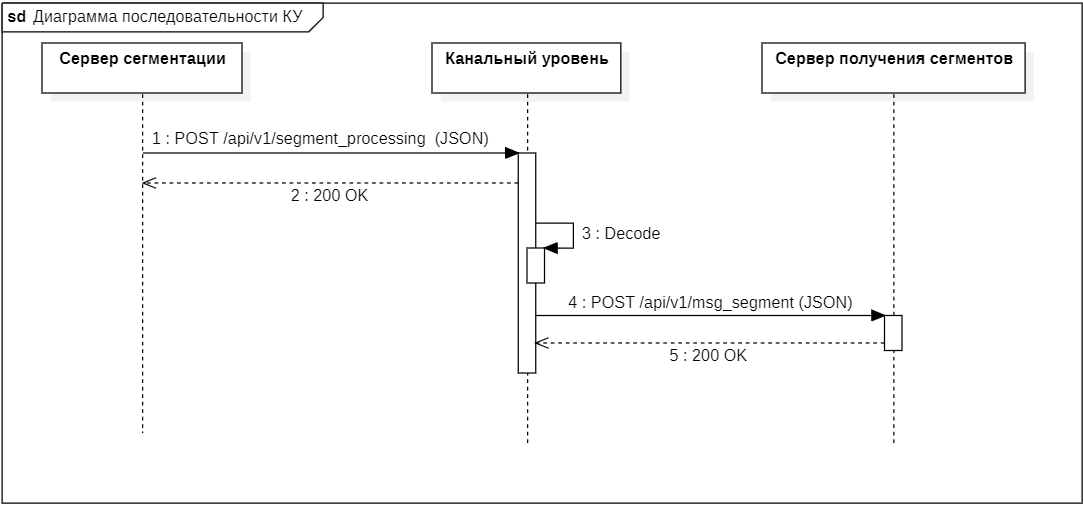


Рисунок 6 – Диаграмма последовательности Канального уровня

Методы, используемые на канальном уровне (Таблица 3):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** |
| 1 | POST | Получение сообщения от транспортного уровня и его последующая обработка | /api/v1/segment\_processing | {  “data”: []byte,  “segment\_id”: int,  “number\_of\_segments”: int,  “date\_send”: string,  “login”: string,  “message\_id”: string, } |

В начале происходит процесс получения сегмента от транспортного уровня в формате json. Для этого выполняется запрос от сервера сегментации, в котором передаются данные сегмента канальному уровню. Далее полученный сегмент кодируется в битовый формат [7, 4]-кодом Хемминга, вносится ошибка в 1 случайный бит сегмента с вероятностью 10%. Также с вероятностью 2% полученный сегмент теряется. Затем сегмент декодируется и после исправления отправляется обратно на транспортный уровень. Если сегмент потерялся, то обратно возвращается код с ошибкой на стороне сервера.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой была разработана система, выполняющая функции мессенджера, то есть предоставляющая возможность обмениваться файлами и сообщениями в реальном времени максимально удобно и надежно. В конечном итоге были достигнуты следующие результаты:

* Разработан дизайн приложения, копирующий интерфейс чата S7 Airlines.
* Спроектирована система и логика взаимодействия сервисов между собой.
* Создан интерфейс на React.
* Создан WebSocket-сервер на Express.js
* Создан веб-сервис на Django 4.2.8.
* Создан сервис транспортного уровня на Golang 1.22.
* Проведено тестирование приложения сервисов.
* Проведено тестирование всей системы как единое целое.
* Подготовлен набор документации, включающий РПЗ, ТЗ, ПМИ, РП, РСА и набор диаграмм.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ВЦИОМ [Электронный ресурс] // Всероссийский центр изучения общественного мнения (ВЦИОМ). URL: https://wciom.ru/analytical-reviews/analiticheskii-obzor/socialnye-seti-i-messendzhery-vovlechennost-i-predpochtenija?ysclid=lvqvbsvvzk727828736 (дата обращения: 27.04.2024)
2. Superjob [Электронный ресурс] // Исследовательский центр портала Superjob.ru. URL: https://www.superjob.ru/research/articles/113923/kompanij/?ysclid=lvqwzm72ye626260592 (дата обращения: 27.04.2024)
3. Официальная документация React: [Электронный ресурс]. // URL: https://legacy.reactjs.org/docs/getting-started.html (Дата обращения: 28.04.2024)
4. WebSocket - Современный учебник JavaScript: [Электронный ресурс]. // URL:  [https://learn.javascript.ru/websocket/](https://learn.javascript.ru/websocket) (Дата обращения: 20.04.2024)
5. Официальная документация библиотеки net/http языка Golang

// URL: https://pkg.go.dev/net/http (Дата обращения 10.04.2024)

1. Блейхут Р. Теория и практика кодов, контролирующих ошибки. [Текст] // Пер. с англ. М.: Мир, 1986. - 576 с.

# **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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Техническое задание

Вариант к27

Студенты группы ИУ5-63Б:

Лобанов И.И.

Валова С.В.

Рыбин В.Д.

2024г.

1. **Наименование:**

Распределённая информационная система обмена текстовыми сообщениями в реальном времени.

1. **Основание для разработки:**

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

1. **Исполнители:**

Валова С.В. (прикладной уровень) – группа ИУ5-63Б

Рыбин В. Д. (канальный уровень) – группа ИУ5-63Б

Лобанов И.И. (транспортный уровень) – группа ИУ5-63Б

1. **Цель разработки:**

Разработать распределённую систему для обмена текстовыми сообщениями 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 сообщений с указанием отправителя;
         1. При подключении к чату пользователь должен ввести логин с помощью кнопки Войти, которое будет передаваться с каждым новым сообщением;
         2. После входа появляется возможность отправки сообщений по установленному WebSocket соединению, используя кнопку “Отправить”;
         3. По кнопке “Выйти” чат и логин очищаются, а ws-подключение закрывается;
         4. В случае, если сообщение пришло с признаком ошибки, сообщение не отображается, а вместо него у получателей появляется соответствующее сообщение;
         5. Дизайн приложения соответствует сайту S7 Airlines;
      2. WebSocket-сервер
         1. Хранит имена пользователей для всех ws-подключений;
         2. Позволяет устанавливать, закрывать ws-соединения, получать сообщения от клиентов и широковещательно рассылать их подключенным клиентам;
      3. Реализация HTTP-метода Receive для получения сообщения с транспортного уровня:
         1. В json каждого сообщения указывается отправитель, признак ошибки и текст сообщения;
         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 Message для получения сообщения и его сегментирования:
         1. Разбиение сообщения на сегменты по 100 байт и их поочередная отправка на канальный уровень;
         2. Каждый сегмент содержит: время отправки (в качестве идентификатора сообщения), общую длину сообщения, номер данного сегмента в сообщении, полезную нагрузку;
      2. Реализация HTTP-метода Send 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 10% в один случайный бит каждого сформированного кадра;
         2. Сервис должен терять передаваемый кадр с вероятностью 2%;
      2. Реализация HTTP-метода segment\_processing для кодирования и декодирования полученного от транспортного уровня сегмента:
         1. Полученный от транспортного уровня json сегмента кодируется [7,4]-кодом Хемминга 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 (14.2.1 и выше)
        2. Golang (версия 1.22)
        3. Kafka (3.3 и выше)
   3. Канальный уровень:
      * 1. ПК с ОС Windows (10 и выше)
        2. Python (3.10 и выше)
        3. Django (4.2.8 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.

# **Приложение 2. 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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Программа и методика испытаний

Вариант 27

Студенты группы ИУ5-63Б:

Валова С.В.

Рыбин В.Д.

Лобанов И.И.

2024г.

1. **Объект испытаний:**

Объектом испытаний является распределённая информационная система

1. **Цель испытаний**

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

1. **Требования к объекту испытаний**

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

1. **Требования к программной документации**

Перед проведением испытаний предъявляются следующие документы:

1. Техническое задание;

2. Программа и методика испытаний;

3. Руководство пользователя.

4. Руководство администратора.

5. Особые требования на испытание системы не предъявляются.

1. **Средства и порядок испытаний**

Методы испытаний и их последовательность для распределенной информационной системы приведены в следующей таблице 1.

Таблица 1 - Методы испытаний и их последовательность

| № | Действие | Результат | № п. ТЗ |
| --- | --- | --- | --- |
| 1 | Авторизация:  Ввод логина. | Авторизация и открытие окна с чатом. | 5.1.1.1 |
| 2 | Ввод и отправка сообщений | Отправка сообщений по установленному WebSocket соединению | 5.1.1.2 |
| 3 | Закрытие чата. | Чат и логин очищаются, а ws-подключение закрывается | 5.1.1.3 |
| 4 | Сообщение с признаком ошибки. | У получателей отображается сообщение «Ошибка при доставке» | 5.1.1.4 |
| 5 | Дизайн приложения соответствует сайту «S7 Airlines» |  | 5.1.1.5 |
| 6 | HTTP-метод Receive для получения сообщения с транспортного уровня. | Полученное по HTTP сообщение отправляется широковещательной WebSocket рассылкой всем подключенным ws-клиентам, кроме тех, у кого логин совпадает с именем отправителя. | 5.1.3 |
| 7 | HTTP-метод Message для получения сообщения с прикладного уровня и его разбиения на сегменты и отправки на канальный уровень. | Полученное сообщение разбилось на сегменты по 100 байт, и отправилось на канальный уровень | 5.2.1 |
| 8 | Метод Segment для получения сегмента с канального уровня и его отправки в очередь Kafka | Полученный сегмент отправлен в очередь в Kafka | 5.2.2.1 |
| 9 | HTTP-метод Send для сборки сегментов в сообщение и отправки прикладному уровню | При повторных запросах в очередь в Kafka (максимум 3 запроса) сообщение полностью собрано и отправлено на прикладной уровень | 5.2.3 |
| 10 | Отправка POST запроса серверу code | Отсутствие ошибки | 5.3.1 |
| 11 | Отправка POST запроса серверу code | Ошибка в бите одного кадра | 5.3.1.1 |
| 12 | Отправка POST запроса серверу code | Полностью потерянное сообщение | 5.3.1.2 |

# **Приложение 3. 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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Руководство пользователя

Вариант 27

Студенты группы ИУ5-63Б:

Валова С.В.

Рыбин В.Д.

Лобанов И.И.

2024 г.

1. **Введение**
   1. **Область применения**

Требования настоящего документа применяются при:

* предварительных комплексных испытаниях;
* опытной эксплуатации;
* приемочных испытаниях;
* промышленной эксплуатации.
  1. **Краткое описание возможностей**

Распределенная система «Платформа для обмена текстовыми сообщениями» представлена в виде веб-приложения, которое поддерживает мгновенный обмен текстовыми сообщениями между пользователями в режиме реального времени.

Она представляет собой окно чата с возможностью ввода текстового сообщения. При подключении необходимо ввести имя пользователя (открытие WebSocket соединения), которое будет передаваться с каждым сообщением. При обновлении страницы история чата не сохраняется. С помощью кнопки выйти можно отчистить чат и логин (закрытие WebSocket соединения), чтобы потом сменить пользователя.

1. **Назначение и условия применения РС «Платформа для обмена текстовыми сообщениями»**
   1. **Назначение РС «Платформа для обмена текстовыми сообщениями»**

Распределенная система «Платформа для обмена текстовыми сообщениями» предназначена для обеспечения мгновенного обмена сообщениями между пользователями в режиме реального времени.

Для использования РС «Платформа для обмена текстовыми сообщениями» необходимо выполнение следующих условий:

* 1. **Системные требования**

Для работы клиента необходимо использовать Yandex Browser или Google Chrome.

1. **Условия выполнения программы**

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

1. **Выполнение программы**
   1. Запуск программы
      1. На компьютере необходимо запустить браузер и ввести в адресную строкуhttp://localhost:3000.
2. **Описание операций**
   1. **Авторизация**

Доступно для: неавторизованные пользователи.

Операция №1: открыть веб страницу

Для авторизации необходимо ввести имя пользователя и нажать на кнопку «Войти» (рис. 7).

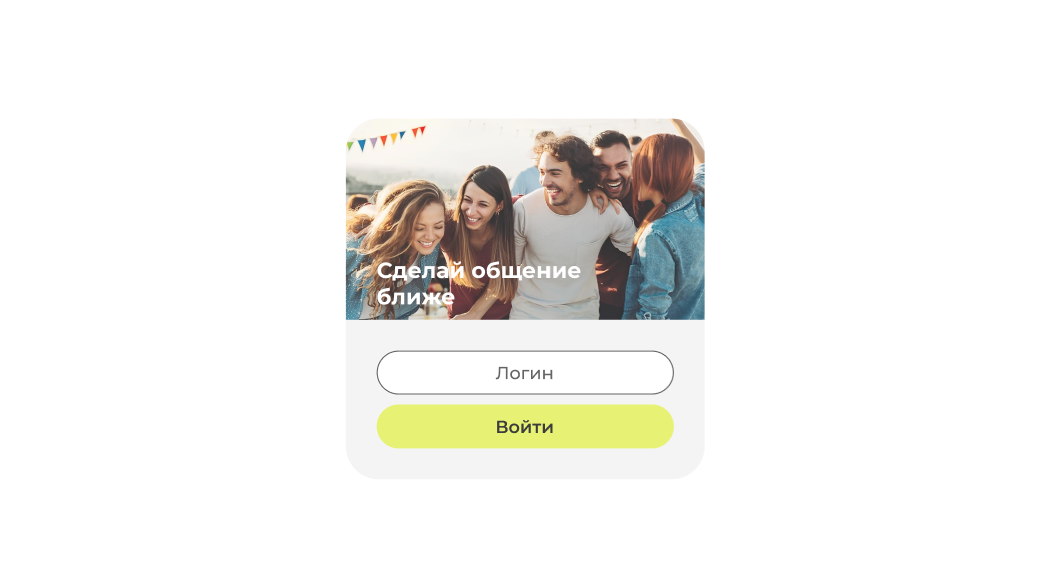


Рисунок 7 - Главный экран неавторизованного пользователя

После успешной авторизации откроется окно чата для отправки и получений текстовых сообщений в режиме реального времени. (рис. 8).

* 1. **Обмен текстовыми сообщениями**

Доступно для: все пользователи.

Операция №1: для отправки текстового сообщения необходимо набрать текст в специальное поле ввода и нажать на кнопку «Отправить».



Рисунок 8 – Страница обмена текстовыми сообщениями

* 1. **Выход из аккаунта**

Доступно для: все пользователи.

Для выхода из аккаунта необходимо нажать на кнопку «Выйти» в правом верхнем углу.

# **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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Руководство системного администратора

Вариант 27

Студенты группы ИУ5-63Б:

Валова С.В.

Рыбин В.Д.

Лобанов И.И.

2024г.

1. **Системные требования**

Для работы клиента необходим Yandex Browser или Google Chrome.

* 1. **Требования к ОС**
     1. Windows 10 или более поздней версии.
     2. MacOS 14.2.1 или более поздняя версия
     3. Ubuntu 18.04 (64-разрядная версия) или более поздней версии.
  2. **Требования к оборудованию**
     1. Центральный процессор: с частотой от 1 ГГц
     2. Оперативная память: 8 Гб и выше
     3. Пространство на жестком диске: 8 Гб
  3. **Требования к ПО**
     1. Node.js 21.2.0 или более поздняя версия
     2. Golang 1.22
     3. Python 3.11, Django 4.2.8 или более поздние версии

1. **Порядок развёртывания серверов:**
   1. **Прикладной уровень**
      1. **WebSocket-сервер**
         1. Перейти в папку ws
         2. Установить зависимости: npm install
         3. Запустить веб-сокет сервис: npm run dev. Сервис запустится на порту 9000
      2. **Frontend** 
         1. Перейти в папку front
         2. Установить зависимости: npm install
         3. Запустить фронтенд: npm run dev. Сервис запустится на порту 3000
   2. **Транспортный уровень**

2.2.1. Скачать код из репозитория <https://github.com/cantylv/coursework-bmstu-2024-networking>

2.2.2. Установить все зависимости: go mod tidy

2.2.3. Запустить веб-сервер командой: go run cmd/main/main.go. Сервер запустился на 8000 порте.

* 1. **Канальный уровень**
     1. Скачать код из репозитория https://github.com/DemiurgeRV/coursework-bmstu2024?tab=readme-ov-file
     2. Перейти в папку проекта
     3. Запустить программу