|  |  |
| --- | --- |
|  | **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 \_\_\_\_\_ГОЛОВНОЙ УЧЕБНО-ИССЛЕДОВАТЕЛЬСКИЙ И МЕТОДИЧЕСКИЙ ЦЕНТР ПРОФЕССИОНАЛЬНОЙ РЕАБИЛИТАЦИИ ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Распределенная информационная система\_\_\_\_\_\_ “Платформа для обмена текстовыми\_\_\_\_\_\_\_\_\_\_ сообщениями”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5Ц-82Б\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_ Дзауров И.А.\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Студент \_\_ИУ5Ц-81Б\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_ Печуркин Д.С.\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Студент \_\_ИУ5Ц-81Б\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_ Смирнов Д.И.\_\_**

(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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студенты группы \_ИУ5Ц-81Б\_\_

\_\_\_\_\_\_\_\_Дзауров Ибрагим Алиханович, Печуркин Дмитрий Сергеевич, Смирнов Даниил Иванович\_\_

(Фамилия, имя, отчество)

Тема курсовой работы \_\_\_\_Создание распределенной информационной системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Дзауров И.А.\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Печуркин Д.С.\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Смирнов Д.И.\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

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

# **СОДЕРЖАНИЕ**

[**СОДЕРЖАНИЕ** 3](#_Toc166287219)

[**ВВЕДЕНИЕ** 4](#_Toc166287220)

[**1.** **Описание прикладного сервиса** 5](#_Toc166287221)

[**2.** **Описание транспортного уровня** 9](#_Toc166287222)

[**3.** **Описание канального сервиса** 12](#_Toc166287223)

[**ЗАКЛЮЧЕНИЕ** 14](#_Toc166287224)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 15](#_Toc166287225)

[**ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 16](#_Toc166287226)

[**ПРИЛОЖЕНИЕ Б ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ** 22](#_Toc166287227)

[**ПРИЛОЖЕНИЕ В РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ** 26](#_Toc166287228)

[**ПРИЛОЖЕНИЕ Г РУКОВОДСТВО СИСТЕМНОГО АДМИНИСТРАТОРА** 30](#_Toc166287229)

# **ВВЕДЕНИЕ**

Основной целью курсовой работы является разработка системы обмена текстовыми сообщениями/файлами в реальном времени, состоящей из трех уровней: прикладного, транспортного и канального, каждый из которых реализован отдельным веб-сервисом.

Функционал данного приложения включает в себя следующее:

* Подключение к чату с помощью имени пользователя (открытие WebSocket соединения), которое будет передаваться с каждым сообщением.
* Окно чата с возможностью отправки текстовых сообщений.
* Очистка истории чата и выход из аккаунта (закрытие WebSocket соединения), после чего можно войти под другим пользователем.
* Данная работа была разбита на несколько составляющих частей:
* Прикладной уровень необходимо разработать интерфейс на базе React + Redux Toolkit + Axios + MUI с функционалом окна чата, позволяющего отправлять текстовые сообщения, а также реализовать WebSocket сервер для обмена сообщениями в режиме реального времени.
* Транспортный уровень необходимо реализовать HTTP сервер на Django для обмена сообщениями между прикладным и канальным уровнем. Реализовать очередь с помощью Apache Kafka, которая позволит эффективно собирать сегменты в единое сообщение и осуществлять их передачу между уровнями, а также определять факт наличия ошибки.
* Канальный уровень необходимо разработать алгоритм на базе кода Хэмминга (7,4), который будет эмулировать работу данного уровня: получение сегмента с транспортного уровня, кодирование сегмента, внесение в сегмент одной ошибки с вероятностью 10%, обнаружение и исправление ошибки, декодирование и отправка обратно на транспортный уровень с учетом потери кадра с заданной вероятности в размере 2%.

# **Описание прикладного сервиса**

Макет приложения был создан в онлайн-редакторе для проектирования интерфейсов Figma [8]. Все элементы, использованные в макете, являются компонентами библиотеки MUI. MUI предоставляет простую, настраиваемую и доступную библиотеку компонентов React. MUI предлагает полный набор инструментов пользовательского интерфейса, которые помогают быстрее внедрять новые функции.

Фронтенд выполнен на языке JavaScript [6]. При его разработке учитывались рекомендации по простоте, удобству и функциональности интерфейса. При разработке приложения использовалась библиотека React [4] — библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (компоненты), которые сохраняют состояние и генерируют элементы пользовательского интерфейса. Для сборки и локальной разработки использовалась утилита vite - интерфейса командной строки (command line interface, CLI), который позволяет создавать предварительно настроенные React-проекты. Предварительная настройка включает в себя, но не исчерпывается, настройкой esbuild.

Так как JavaScript не является типизированным языком, в проекте также использовался язык программирования TypeScript. Преимущества использования TypeScript состоят в том, что строгая типизация значительно сокращает количество возможных ошибок, удобно интегрируется с IDE, расширяет возможности JavaScript в ООП (TypeScript позволяет использовать наследование, абстрактные классы и многое другое), компилятор TS позволяет преобразовывать код в JS-код более старых стандартов языка, что повышает поддержку браузерами, а также повышает наглядность и документируемость кода.

Для управления состоянием в проекте используется библиотека redux-toolkit.

Для отправки запросов на бэкенд используется библиотека Axios. Axios — это облегченный клиент HTTP на базе сервиса $http с Angular.js, похожего на собственный JavaScript Fetch API.

Для обмена сообщениями в режиме реального времени используется WebSocket [7], который поддерживает непрерывное соединение между сервером и клиентом, мгновенно передавая новые данные при их обновлении. Websocket это протокол прикладного уровня, предоставляющий двунаправленный обмен данными между клиентом и сервером через одно логическое соединение TCP.

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

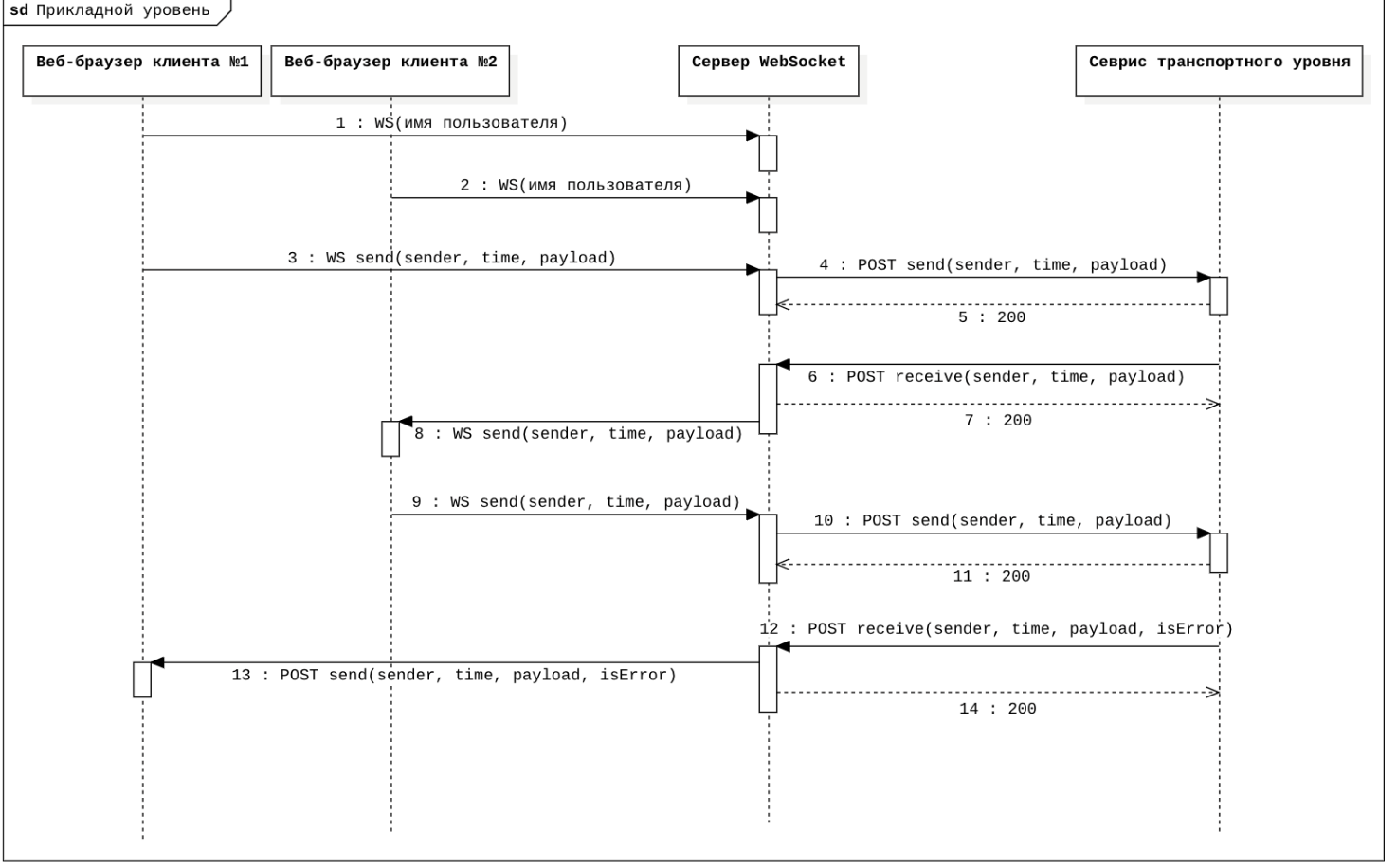


Рисунок 1 – Диаграмма последовательности прикладного уровня

При запуске веб-приложения посетитель попадает на страницу авторизации. Для подключения к чату 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. Если сообщение пришло с признаком ошибки - текст не отображается, а в чате отображается значок ошибки.

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/ | {  "sender":string",  "time": string  "payload": string,  "isError": boolean  } |

Ниже приведена первоначальная страница неавторизованного пользователя (Рисунок 2). Для присоединения к чату пользователю необходимо ввести в поле ввода имя и нажать на кнопку «Войти». В этот момент происходит открытие WebSocket соединения и перенаправление на страницу с чатом.

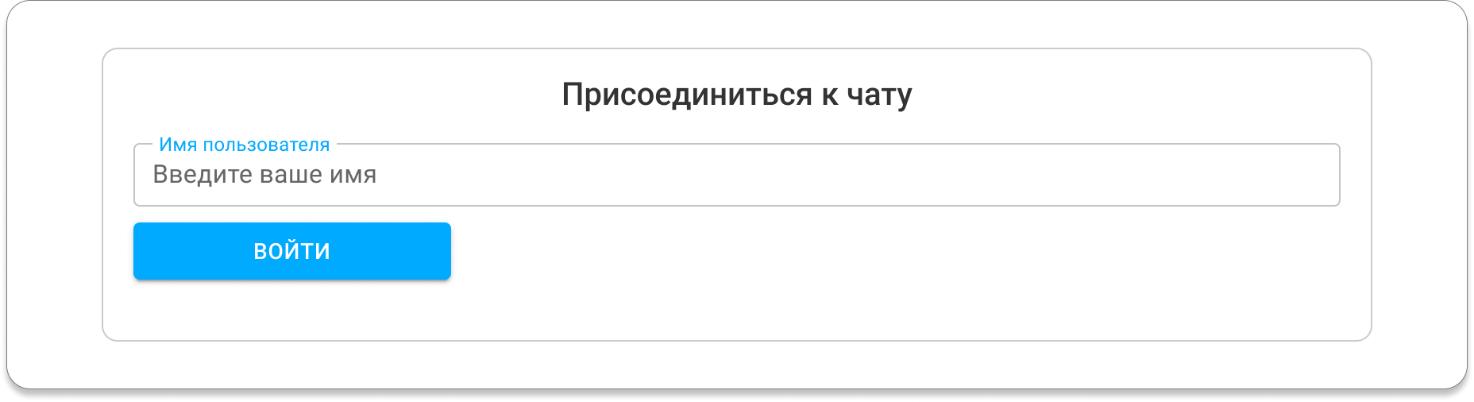


Рисунок 2 *–* Страница авторизации

После авторизации пользователю доступна страница с чатом (Рисунок 3) для обмена текстовыми сообщениями в режиме реального времени.

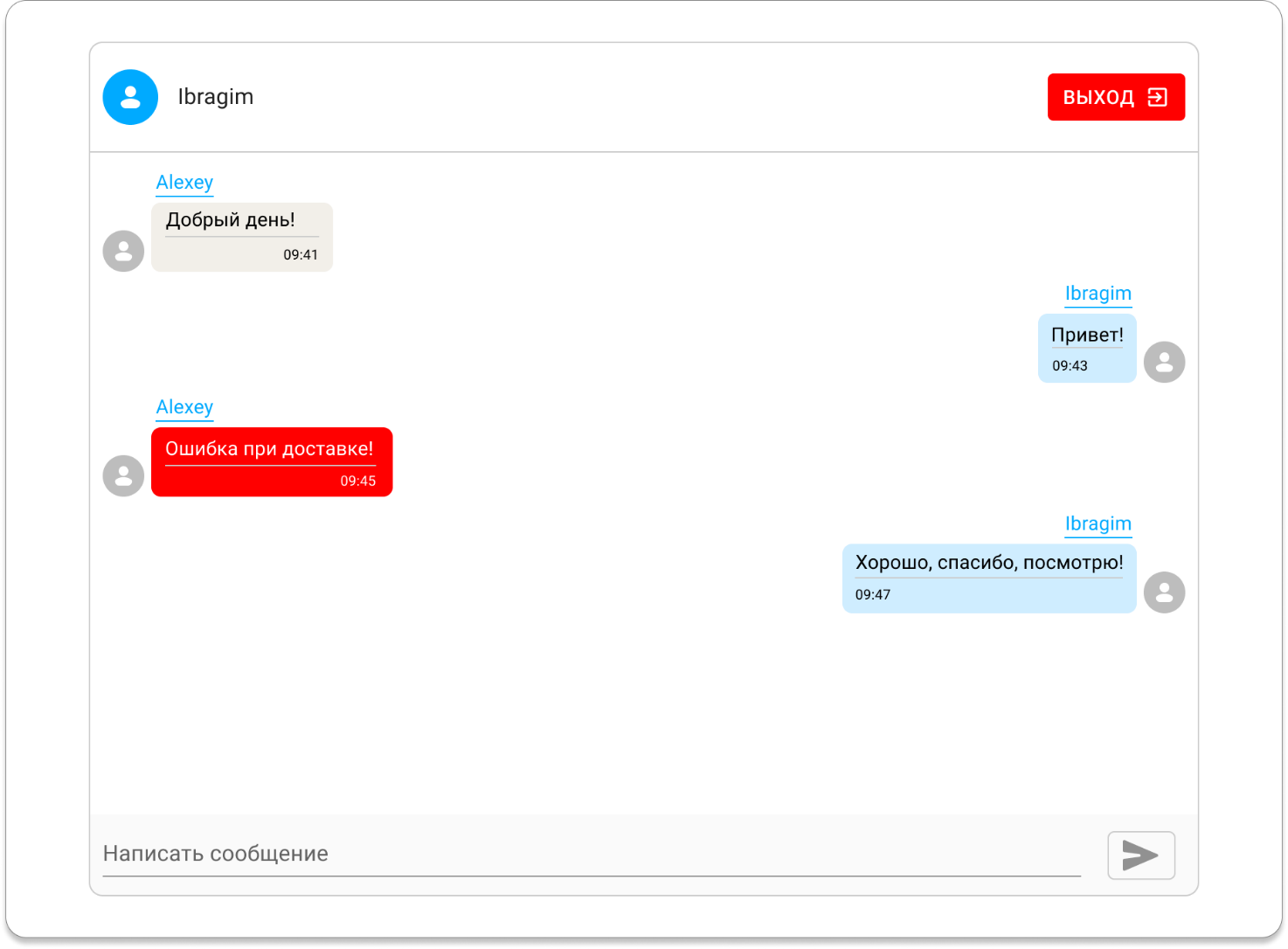


Рисунок 3 *–* Страница обмена текстовыми сообщениями

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

# **Описание транспортного уровня**

Для более глубокого понимания архитектуры системы и ее развертывания представлена диаграмма развертывания. На этой диаграмме отображены уровни, на которых размещены компоненты системы, а также их взаимосвязи и распределение по узлам сети.

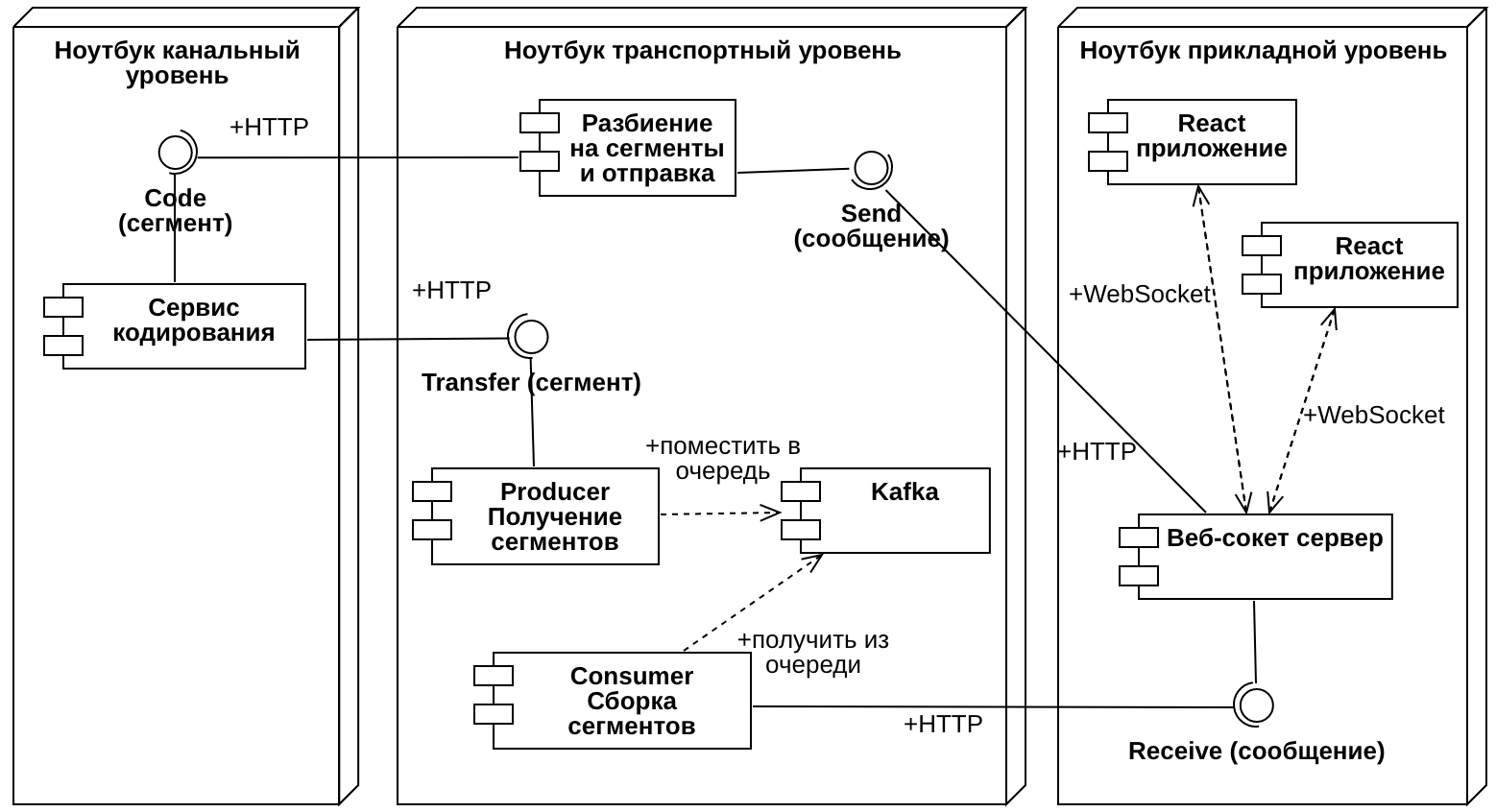


Рисунок 4 – Диаграмма развертывания

Транспортный уровень в системе играет ключевую роль в обеспечении эффективной и надежной передачи сообщений между канальным и прикладным уровнями. Его задачей является разбиение сообщений на сегменты, их отправка на канальный уровень, а также сборка сегментов в целостные сообщения для передачи на прикладной уровень.

HTTP-метод Send предоставляет интерфейс для отправки сообщений от отправителей на транспортный уровень. Этот метод получает данные сообщения, включая имя отправителя, текстовую полезную нагрузку и время отправки. После получения сообщения оно разбивается на сегменты по 200 байт. Каждый сегмент содержит информацию об отправителе, времени отправки, общем количестве сегментов и номере сегмента в сообщении. Затем сегменты отправляются на канальный уровень через метод Code, который обеспечивает их доставку и сохранность. HTTP-метод Transfer, в свою очередь, принимает сегменты сообщений с канального уровня. Полученные сегменты помещаются в очередь для последующей сборки в целостные сообщения прикладного уровня. Используется брокер Apache Kafka [2], который сохраняет сегменты в теме для последующей передачи и обработки. Это обеспечивает отказоустойчивость и надежность передачи данных. Раз в 2 секунды сегменты собираются в единое сообщение. В случае, если часть сегментов не была принята (например, из-за потери связи или ошибок в передаче), итоговое сообщение передается на прикладной уровень с признаком ошибки.

Сервисы транспортного уровня реализованы на языке Python [1] с использованием фреймворка Django [3]. Django обеспечивает простоту разработки, масштабируемость и гибкость при создании веб-приложений. Это позволяет удобно управлять HTTP-запросами и обрабатывать данные, поступающие на транспортный уровень.

Для наглядного представления взаимодействия между компонентами системы и последовательности выполнения операций на рисунке 5 представлена диаграмма последовательности. Диаграмма иллюстрирует шаги, которые выполняются при отправке и передаче сегментов сообщения между различными уровнями системы.

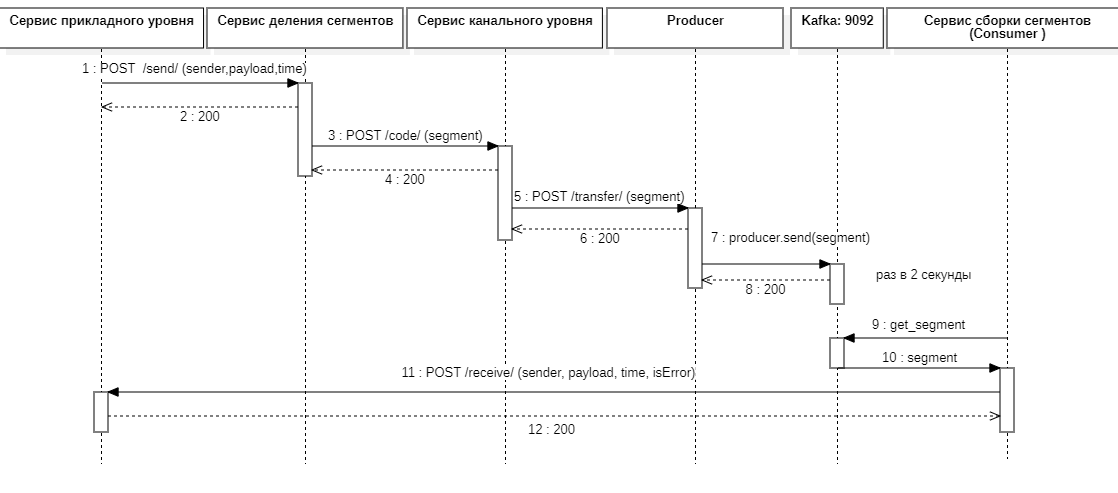


Рисунок 5 – Диаграмма последовательности

Методы, используемые на транспортном уровне (Таблица 2):

**Таблица 2 – Методы транспортного уровня**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** |
| 1 | POST | Получение сообщения с транспортного уровня | /send/ | {  "sender":string",  "time": string  "payload": string,  "isError": boolean  } |
| 2 | POST | Получение сообщения с канального уровня | /transfer/ | {  sender: username,  payload:string,  time:string,  total\_segments:  integer,  segment\_number:  integer  } |

# **Описание канального сервиса**

Канальный сервис эмулирует работу канального уровня, который представляет собой критически важный уровень передачи данных в архитектуре модели OSI, обеспечивающий надежность и целостность информации в сети.

Технологии, используемые на канальном уровне, включают в себя язык программирования Python [1] в сочетании со встроенной библиотекой http.server [5]. Эти инструменты предоставляют средства для создания и управления HTTP-серверами, а также обеспечивают удобство разработки и эффективную обработку HTTP-запросов, способствуя тем самым гарантированию надежности и целостности передаваемой информации.

Основным элементом на канальном уровне является POST-метод code. Он выполняет кодирование, внесение ошибки, обнаружение и исправление ошибки и декодирование данных, полученных от транспортного уровня. Затем происходит передача сегментов данных на транспортный уровень с помощью POST-метода transfer, реализованного со стороны транспортного уровня. Реализация метода code основана на принципах алгоритма Хэмминга [7,4], который обеспечивает обнаружение и исправление ошибок и надежность передачи данных.

Для более наглядного представления взаимодействия между компонентами системы и последовательности выполнения операций при отправке и передаче сегментов сообщений между канальным и транспортным уровнями, на рисунке 6 приведена диаграмма последовательности. Эта диаграмма иллюстрирует каждый шаг процесса, начиная от отправки данных на одном уровне до их приема и обработки на другом, позволяя лучше понять взаимодействие между уровнями системы и порядок выполнения операций.

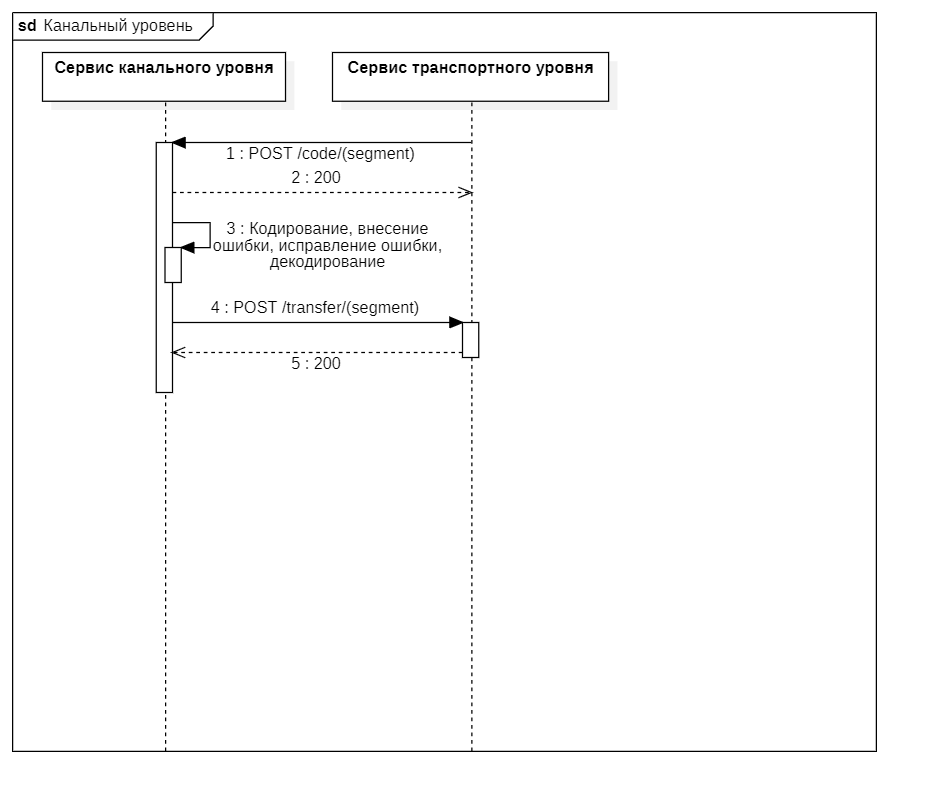


Рисунок 6 – Диаграмма последовательности

Методы, используемые на канальном уровне (Таблица 3):

**Таблица 3 – Методы канального уровня**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **Url** | **Входные данные** |
| 1 | POST | Получение сообщения с прикладного уровня | /send/ | {  sender: username,  payload:string,  time:string  } |

# 

# **ЗАКЛЮЧЕНИЕ**

В заключении курсовой работы по разработке системы обмена текстовыми сообщениями в реальном времени, можно выделить следующие ключевые достижения:

* В процессе работы была эмулирована работа трех уровней модели OSI: канального, транспортного и прикладного.
* Была разработана документация, описывающая структуру и функционал системы, а также создано руководство по развёртыванию и использованию.
* Созданная система является примером важности каждого уровня модели OSI и их взаимодействия в процессе передачи данных.
* Продемонстрировано, как информация проходит через различные уровни обработки и транспортировки данных, подчеркивая значимость корректной работы каждого уровня для обеспечения надежной передачи данных в современных сетевых приложениях.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Официальная документация Python: [Электронный ресурс]. // URL: https://www.python.org/doc/ (Дата обращения: 28.04.2024)
2. Официальная документация Apache Kafka: [Электронный ресурс]. // URL: https://kafka.apache.org/documentation/(Дата обращения: 28.04.2024)
3. Официальная документация Django: [Электронный ресурс]. // URL: https://docs.djangoproject.com/en/4.2/ (Дата обращения: 28.04.2024)
4. Официальная документация React: [Электронный ресурс]. // URL: https://legacy.reactjs.org/docs/getting-started.html (Дата обращения: 28.04.2024)
5. Официальная документация Python по настройке HTTP-сервера: [Электронный ресурс]. // URL: <https://docs.python.org/3/library/http.server.html> (Дата обращения: 20.04.2024)
6. Современный учебник JavaScript: [Электронный ресурс]. // URL: <https://learn.javascript.ru/> (Дата обращения: 20.04.2024)
7. WebSocket - Современный учебник JavaScript: [Электронный ресурс]. // URL:  [https://learn.javascript.ru/websocket/](https://learn.javascript.ru/websocket) (Дата обращения: 20.04.2024)
8. Инструкция по базовым возможностям Figma: [Электронный ресурс]. // URL: <https://practicum.yandex.ru/blog/chto-takoe-figma-dlya-dizainera/> (Дата обращения: 20.04.2024)

# **ПРИЛОЖЕНИЕ А 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 3

Студенты:

Дзауров И.А., группа ИУ5Ц-82Б

Смирнов Д.И., группа ИУ5Ц-81Б

Печуркин Д.С., группа ИУ5Ц-81Б

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:**

Дзауров И. А. (прикладной уровень) – группа ИУ5Ц-82Б

Смирнов Д. И. (транспортный уровень) – группа ИУ5Ц-81Б

Печуркин Д. С. (канальный уровень) – группа ИУ5Ц-81Б

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 и времени отправки;
         1. Чтобы войти в чат, пользователь должен указать свое имя, которое будет автоматически передаваться вместе с каждым новым текстовым сообщением и нажать кнопку “Войти”;
         2. После входа появляется возможность ввода и 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 …
      2. WebSocket-сервер
         1. Хранит имена пользователей для всех ws-подключений;
         2. Позволяет устанавливать, закрывать ws-соединения, получать сообщения от клиентов и широковещательно рассылать сообщения 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 200 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 10% 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 [7,4]-кодом Хэмминга 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 Windows(10 и выше)
        2. Django (4.2 и выше)
        3. Kafka (2.12 и выше)
        4. Python (3.0 и выше)
   3. Канальный уровень:
      * 1. ПК с ОС Windows(10 и выше)
        2. Python (3.0 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 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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Программа и методика испытаний

Вариант 3

Студенты:

Дзауров И.А., группа ИУ5Ц-82Б

Смирнов Д.И., группа ИУ5Ц-81Б

Печуркин Д.С., группа ИУ5Ц-81Б

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 имени пользователя. | 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 «Авито» |  | 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 Send, разбиение на сегменты | Полученное сообщение с прикладного уровня разбивается на сегменты, кодируется в бинарный вид, и отправляется на канальный уровень | 5.2.1 |
| 8 | HTTP-метода Transfer. Сборка сегментов и отправка на прикладной уровень | Каждые две секунды из очереди Kafka запрашиваются сегменты, которые объединяются в единое сообщение и отправляются на прикладной уровень с идентификатором ошибки | 5.2.2 |
| 9 | Кодирование Хэммингом [7,4] | Получение исходного сегмента, его кодирование с добавлением 3 битов четности, внесение 1 ошибки в сегмент из 7 битов, обнаружение и исправление ошибки через код Хэмминга [7,4], декодирование сегмента. При выполнении будет отображаться в консоли: исходный сегмент, его закодированная форма, информация об ошибке, позиция ошибочного бита и окончательно декодированный сегмент. Информация об исправленной ошибке не будет выводится, поскольку алгоритм Хэмминга гарантирует её исправление. | 5.3.1 |
| 10 | HTTP-метод Code: метод кодирует сегмент, полученный от транспортного уровня, вносит ошибку в сегмент, исправляет ошибку в сегменте, декодирует сегмент | Передача сегментов через HTTP-метод Transfer транспортного уровня. Вывод JSON отправленного сообщения в консоль. При потере сегмента в консоли выводится сообщение "Потеря кадра". | 5.3.2 |

# **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 «Информатика и системы управления»**

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

Дисциплина «Сетевые технологии в АСОИУ»

Руководство пользователя

Вариант 3

Студенты:

Дзауров И.А., группа ИУ5Ц-82Б

Смирнов Д.И., группа ИУ5Ц-81Б

Печуркин Д.С., группа ИУ5Ц-81Б

2024г.

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).

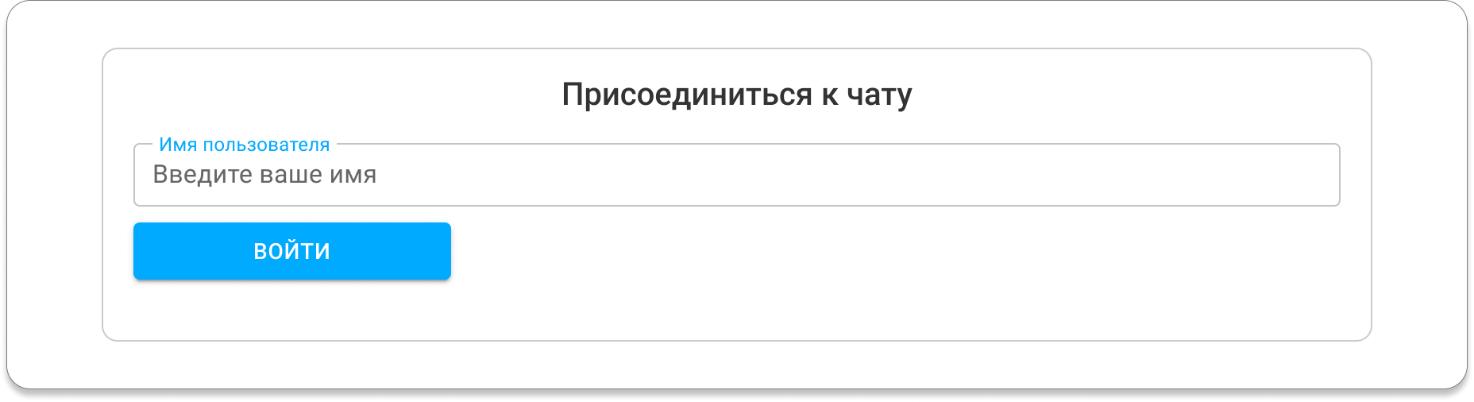


Рисунок 7 - Главный экран неавторизованного пользователя

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 клавишу «Enter».

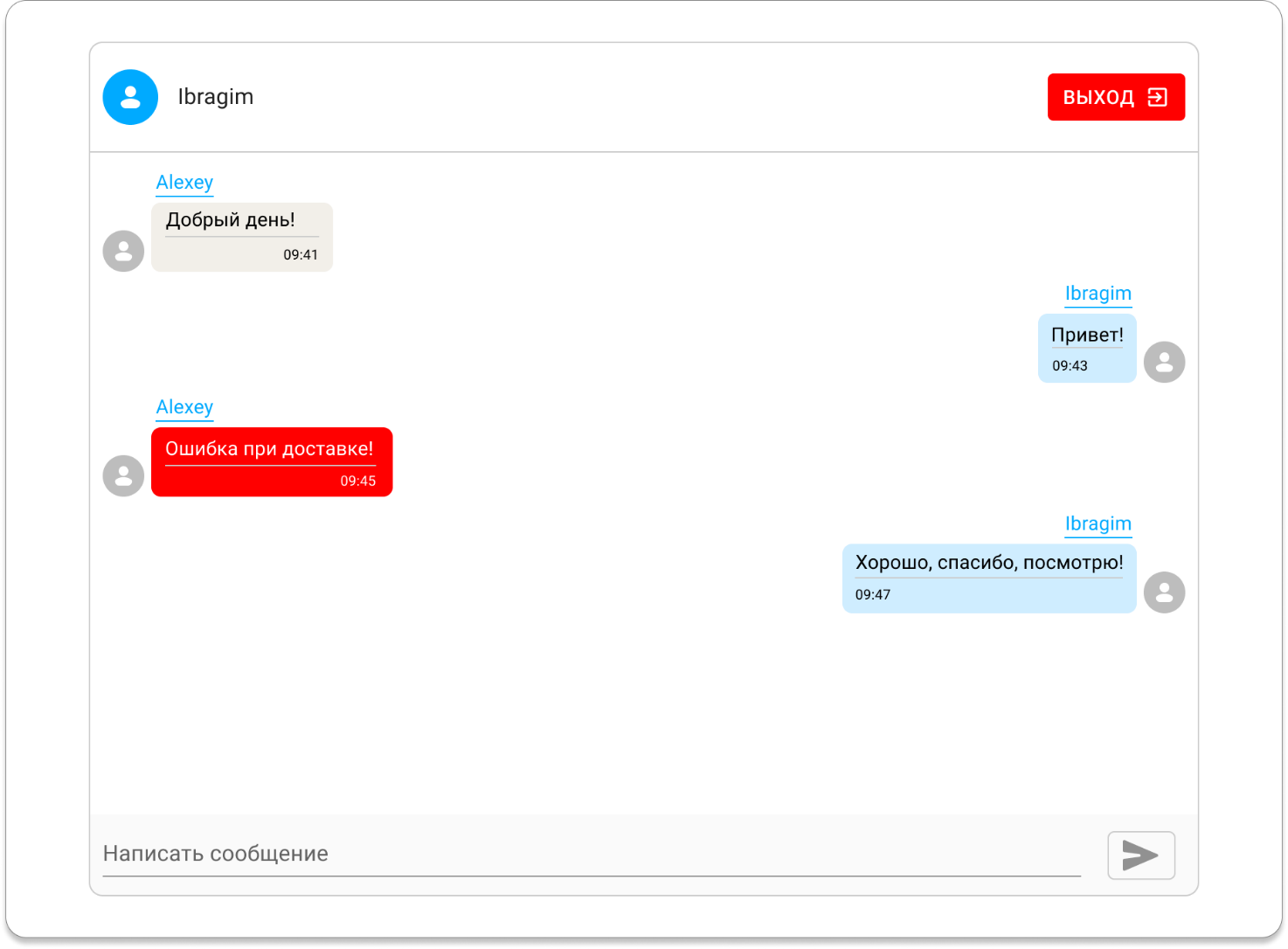


Рисунок 8 – Страница обмена текстовыми сообщениями

* 1. **Выход из аккаунта**

Доступно для: все пользователи.

Для выхода из аккаунта необходимо нажать на кнопку выхода в правом верхнем углу меню.

# **ПРИЛОЖЕНИЕ Г 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 3

Студенты:

Дзауров И.А., группа ИУ5Ц-82Б

Смирнов Д.И., группа ИУ5Ц-81Б

Печуркин Д.С., группа ИУ5Ц-81Б

2024г.

1. **Системные требования**

Для работы клиента необходим Yandex Browser или Google Chrome.

* 1. **Требования к ОС**
* Windows

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

* MacOS

macOS High Sierra 10.13 или более поздней версии.

* Linux

Ubuntu 18.04 (64-разрядная версия) 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. **Требования к ПО**
     1. Сервер для запуска микросервисов с установленным Node.js версии 14 и выше;
     2. Браузер, совместимый с веб-приложением (например, Google Chrome, Mozilla Firefox, Safari);
     3. Для работы с Django также потребуется установленный фреймворк Django и Python версии 3 и выше;

1. **Порядок развёртывания серверов:**
   1. **Сервис канального уровня**
2. Перейти в каталог с сервисом канального уровня:

* cd link\_layer

1. Запустить сервис канального уровня:

* py server.py

После этого на порту 8000 будет доступен сервис канального уровня

* 1. **Сервис транспортного уровня**

1. Запуск терминала
2. Запуск Zookeeper:

C:\kafka>.\bin\windows\zookeeper-server-start.bat C:\kafka\config\zookeeper.properties

1. Запуск Apache Kafka:

C:\kafka\_2.13-3.7.0>.\bin\windows\kafka-server-start.bat C:\kafka\_2.13-3.7.0\config\server.properties

1. Запуск Django:

* D:\Sem8\kursovaya\> .\venv\Scripts\activate;
* D:\Sem8\kursovaya\> cd .\transport\_level\;
* python manage.py runserver 192.168.3.10:8000.
  1. **Сервис прикладного уровня**
     1. **WebSocket** 
        1. Перейти в каталог с сервисом Websocket:
* cd ws-server
  + - 1. Установить зависимости:
* npm install
  + - 1. Запустить Websocket сервис:
* npm run dev

После этого на порту 3000 будет доступен сервис WebSocket.

* + 1. **Фронтенд**
       1. Перейти в каталог с фронтендом:
* cd messenger
  + - 1. Установить зависимости:
* npm install
  + - 1. Запустить сервис фронтенда:
* npm run dev

После этого на порту 5173 будет доступен сервис фронтенда.