



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 СИСТЕМЫ УПРАВЛЕНИЯ \_\_\_\_\_

КАФЕДРА \_\_\_\_\_ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ \_\_\_\_\_

## РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Создание распределенной информационной  
системы в соответствии с вариантом

Студент ИУ5-63Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

В.А. Кузнецов  
(И.О.Фамилия)

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата)

А.И. Канев  
(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)

---

ФАКУЛЬТЕТ \_\_\_\_\_ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ \_\_\_\_\_

КАФЕДРА \_\_\_\_\_ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ \_\_\_\_\_

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОЙ РАБОТЕ*

*НА ТЕМУ:*

*Создание распределенной информационной  
системы в соответствии с вариантом*

---

---

---

---

Студент ИУ5-63Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

**Н.Д. Рассказов**  
(И.О.Фамилия)

Руководитель курсовой работы  
\_\_\_\_\_

(Подпись, дата)

**А.И. Канев**  
(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)

---

ФАКУЛЬТЕТ \_\_\_\_\_ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ \_\_\_\_\_

КАФЕДРА \_\_\_\_\_ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ \_\_\_\_\_

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОЙ РАБОТЕ*

*НА ТЕМУ:*

*Создание распределенной информационной*  
*системы в соответствии с вариантом*

---

---

---

---

Студент ИУ5-63Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Д.К. Пермяков  
(И.О.Фамилия)

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата)

А.И. Канев  
(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.

## З А Д А Н И Е на выполнение курсовой работы

по дисциплине Сетевые технологии в АСОИУ

Студент группы ИУ5-63Б Кузнецов В.А., Рассказов. Н.Д., Пермяков Д.К.  
(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

Задание Разработать автоматизированную распределенную систему для обмена сообщениями в  
реальном времени

### ***Оформление курсовой работы:***

Расчетно-пояснительная записка на 41 листе формата А4.

Дата выдачи задания « 09 » февраля 2024 г.

### **Руководитель курсовой работы**

А.И. Канев  
(И.О.Фамилия)

### **Студент**

В.А. Кузнецов  
(И.О.Фамилия)

### **Студент**

Н.Д. Рассказов  
(И.О.Фамилия)

### **Студент**

Д.К. Пермяков  
(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  
Студент группы ИУ5-63Б Кузнецов В.А., Рассказов Н.Д., Пермяков Д.К.  
(Фамилия, имя, отчество)  
Тема курсовой работы Распределенная информационная система обмена  
сообщениями в реальном времени

№ 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	<u>25.02.24</u> Планируемая дата	19.02.24		
2.	Создание макета Figma, диаграммы последовательности и формирование swagger	<u>31.03.24</u> Планируемая дата	31.03.24		
3.	Оформление РПЗ, ПМИ, РСА, РП	<u>28.04.24</u> Планируемая дата	29.04.24		
4.	Тестирование распределенной системы и подготовка проекта к демонстрации	<u>15.05.24</u> Планируемая дата	15.05.24		
5.	Защита курсовой работы	<u>20.05.24</u> Планируемая дата	20.05.24		

Студент \_\_\_\_\_  
(подпись, дата)

Руководитель работы \_\_\_\_\_  
(подпись, дата)

Студент \_\_\_\_\_  
(подпись, дата)

Студент \_\_\_\_\_  
(подпись, дата)

## **Оглавление**

1. Введение .....	7
2. Сервис транспортного уровня .....	9
3. Сервис канального уровня.....	12
4. Сервис прикладного уровня .....	15
5. Заключение.....	19
6. Список использованных источников.....	20
7. Приложения.....	<b>Ошибка! Закладка не определена.</b>

## **Введение**

В текущем политическом и социальном контексте России разработка собственного мессенджера имеет важное значение. По данным исследования Института развития интернета, проведенного в октябре 2019 года, 53% россиян готовы полностью перейти с зарубежных интернет-сервисов на отечественные [1]. Это указывает на значительную потребность в создании надежных и доверенных внутренних альтернатив. Также стоит отметить, что многие российские пользователи все чаще получают информацию через социальные сети и интернет, а телевидение теряет свою популярность как основной источник новостей. Согласно другому исследованию Левада-Центра, 42% россиян используют социальные сети для получения ежедневных новостей [2]. Это подчеркивает важность наличия надежных и удобных цифровых платформ для коммуникаций и обмена информацией. Эти данные подчеркивают актуальность и важность разработки мессенджера в России, который мог бы удовлетворить запросы местного населения на безопасность, приватность и доверие к сервисам, обеспечивая при этом свободу общения без опасений по поводу цензуры или вмешательства.

Целью является разработка системы обмена текстовыми сообщениями, которая будет функционировать в режиме реального времени. Эта система будет строиться на трёх ключевых уровнях, обеспечивающих её эффективность и надёжность. На прикладном уровне целью является разработка пользовательского интерфейса, который будет интуитивно понятен и удобен для конечного пользователя. Транспортный уровень предусматривает разработку механизмов для передачи данных между пользователями через интернет и проверки целостности данных. Канальный уровень занимается непосредственной обработкой данных на физическом уровне, включая кодирование и декодирование сигналов, а также обеспечение их доставки по каналу связи с помехами. Работа на этом уровне включает в себя решение проблем, связанных с надёжностью передачи данных. Таким образом, создаваемая система должна обеспечивать не только быстрый и удобный обмен

сообщениями, но и высокий уровень надежности, что делает её актуальной в условиях современного цифрового мира.

Данный мессенджер может быть использован отечественными организациями, которым необходимо для различных задач вести общие рабочие переписки для возможного уточнения деталей задания и улучшения опыта совместной работы.

Нефункциональные требования к разрабатываемой системе:

1. Интерфейс системы и текст ошибок должны быть русифицируемы.
2. Клиентская часть должна быть разработана под IOS.
3. Серверная часть должна поддерживать кроссплатформенность.

В ходе работы необходимо выполнить следующие задачи:

1. Спроектировать логику взаимодействия между компонентами.
2. Разработать дизайн приложения, копирующий сайт российской компании.
3. Создать WebSocket-сервер на языке Swift 5.9.
4. Создать веб-сервис на языке Golang 1.21.
5. Создать мобильное приложение на SwiftUI.
6. Развернуть систему на нескольких компьютерах и провести тестирование.
7. Подготовить набор документации, включающий ТЗ, ПМИ, РП, РСА и диаграммы.

## Сервис транспортного уровня

Транспортный уровень в архитектуре мессенджера выполняет задачу по передаче сообщений между пользователями. Он также играет роль связующего звена между прикладным и канальным уровнями.

Сервера транспортного уровня написаны на Golang и фреймворке Gin [3]. Временное хранение сегментов происходит в очереди в Kafka [4]. Взаимодействие с другими уровнями сервиса происходит при помощи REST API. Связь сервисов показана на диаграмме развёртывания:

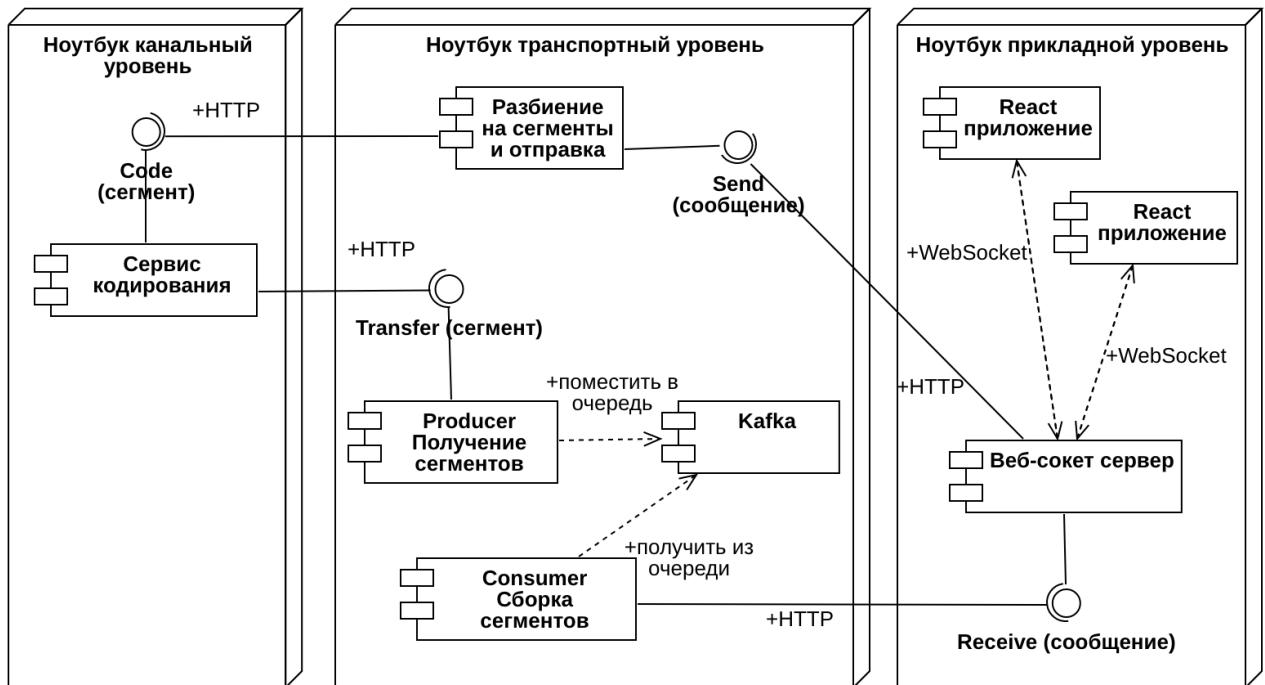


Рисунок 1 – Диаграмма развёртывания

Для эффективности передачи данных на транспортном уровне применяется контроль перегрузки (congestion control). Контроль перегрузки — это механизм управления 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. Основными задачами контроля перегрузки являются выявление наступающей перегрузки и адаптация скорости передачи данных для поддержания стабильной и эффективной работы сети. В нашем проекте используется сегментация сообщений перед отправкой на канальный уровень. При получении с канального уровня сегменты

помещаются в Kafka. Kafka представляет собой распределенный брокер сообщений, обеспечивающий отказоустойчивые конвейеры, работающие по принципу «публикация/подписка» и позволяющие обрабатывать потоки событий.

Когда пользователь отправляет сообщение, оно в виде JSON объекта передаётся с прикладного уровня на сервер сегментации. Здесь оно разбивается на сегменты по 120 байт, к которым добавляются время отправки, количество сегментов и номер данного сегмента. Затем эти пакеты-сегменты отправляются на канальный уровень, откуда они попадают на сервер получения сегментов. Его задача — формирование очереди в Kafka для дальнейшей обработки. Последний сервер транспортного уровня занимается сборкой сегментов в целые сообщения. Раз в 2 секунды он проверяет очередь, если получены все сегменты одного сообщения, то они объединяются и сообщение отправляется на прикладной уровень. Если же какие-то сегменты были утеряны при передаче через канальный уровень, на прикладной уровень отправляется сообщение об ошибке.

Взаимодействие серверов транспортного уровня между собой и с другими частями системы отражено на диаграмме последовательности (Рисунок 2).

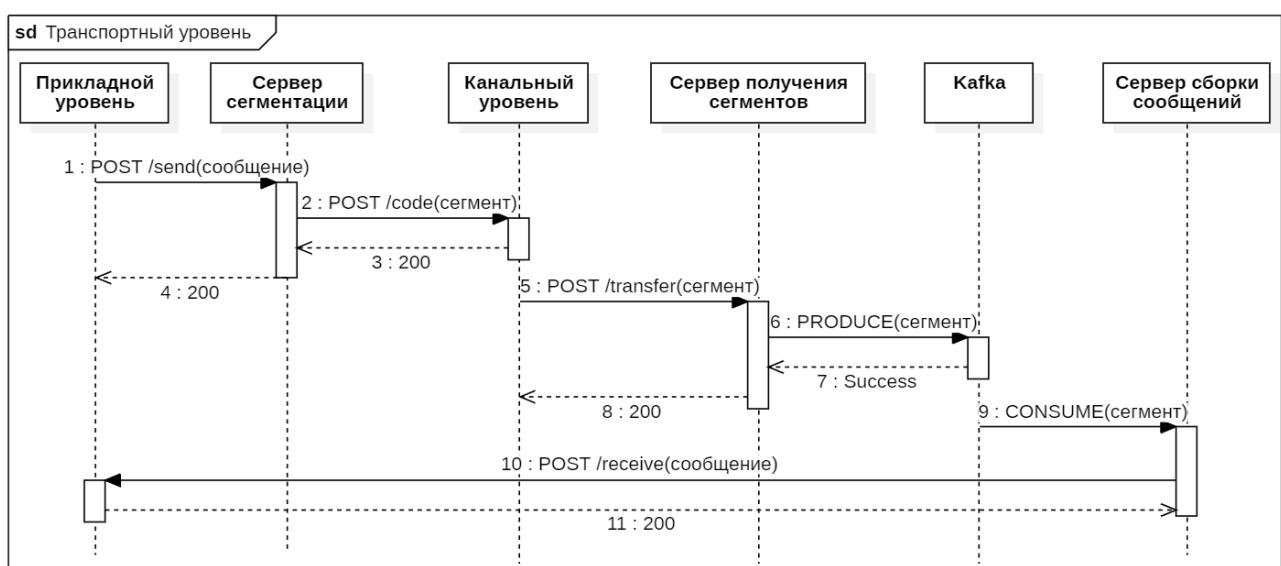


Рисунок 2 – диаграмма последовательности транспортного уровня

Методы, реализуемые транспортным уровнем, представлены в таблице 1:

**Таблица 1 – Методы транспортного уровня**

Метод	Входные параметры	Описание
POST Send	uuid string message string userName string errorCode int	Отправка сообщения с прикладного уровня на транспортный
POST Transfer	segment { data string time int number int count int } error boolean	Отправка декодированного сегмента с канального уровня на транспортный

## **Сервис канального уровня**

Канальный уровень в архитектуре мессенджера выполняет задачу по эмуляции канала связи с потерями. Он с определенной вероятностью вносит ошибку в один случайный бит каждого сформированного кадра. Когда данные приходят на канальный уровень, они могут либо потеряться полностью и затем вернуться, либо сразу вернуться на следующий сервер транспортного уровня.

Сервера канального уровня написаны на Golang и фреймворке Gin [3]. Взаимодействие с другими уровнями сервиса происходит при помощи REST API.

Когда пользователь отправляет сообщение, оно в виде сегмента размера 120 байт через транспортный уровень передается с прикладного уровня на сервер кодирования. Здесь сегмент кодируется циклическим [7,4]-кодом для получения кадра. Затем с вероятностью 10% в один случайный бит каждого сформированного кадра вносится ошибка. После внесения ошибки в кадр он декодируется с исправлением ошибки и передается далее в виде сегмента на транспортный уровень.

Циклический код — это тип линейного блочного кода, широко используемый для обнаружения и исправления ошибок в цифровых данных. Основное назначение циклического кода заключается в обеспечении надежности передачи информации через шумные каналы связи, где возможны искажения данных. В циклических кодах все возможные кодовые слова образуют циклическую группу, что означает, что циклический сдвиг любого кодового слова приводит к другому кодовому слову. Это свойство облегчает процесс кодирования и декодирования, делая его более эффективным. Одним из наиболее известных примеров циклических кодов является код БЧХ (Бозе-Чоудхури-Хоквингема), который способен исправлять несколько ошибок и часто применяется в современных системах связи и хранения данных. Циклические коды включают генераторный полином, который используется для кодирования исходных данных путем деления сообщения на этот полином с получением остатка, который затем добавляется к исходному сообщению.

Этот остаток служит проверочным кодом, который помогает выявить и исправить ошибки при декодировании. Основным преимуществом циклических кодов является их простота и эффективность в реализации на аппаратном уровне, что делает их популярными для применения в сетях передачи данных и устройствах хранения информации.

Работа сервера канального уровня и взаимодействие с другими частями системы отражено на диаграмме последовательности (Рисунок 3).

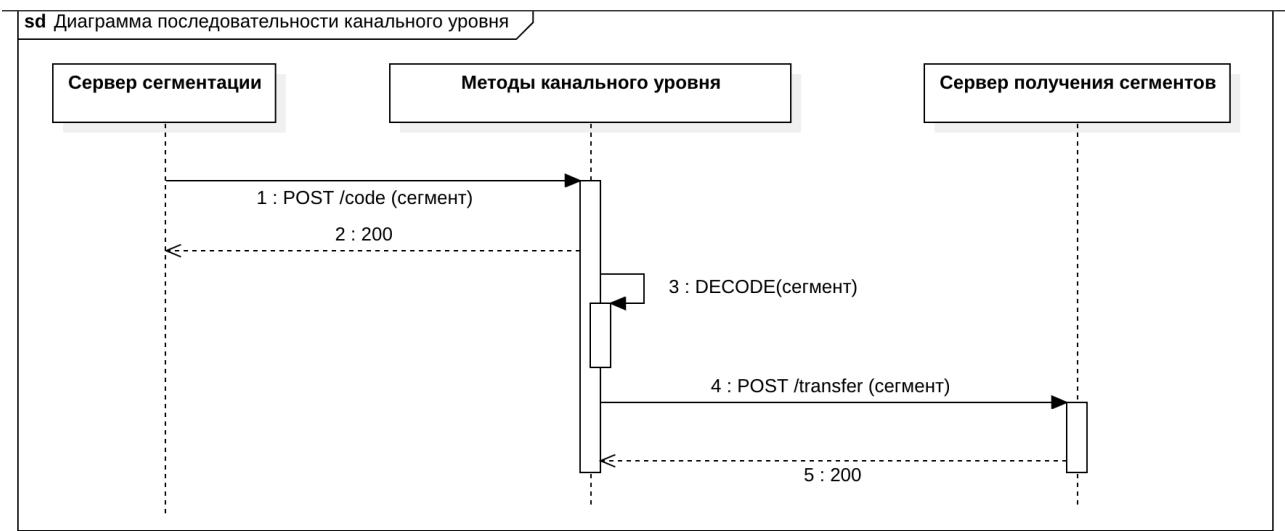


Рисунок 3 – диаграмма последовательности транспортного уровня

Метод, используемый на канальном уровне (Таблица 2):

## **Таблица 2 – Методы канального уровня**

Метод	Входные параметры	Описание
POST Code	Data string Time integer Number integer Count integer	Полученный от транспортного уровня json сегмента кодируется циклическим [7, 4] кодом в битовый формат. Вносится ошибка в 1 бит сегмента. Далее сегмент декодируется и либо

Метод	Входные параметры	Описание
		исправляется ошибка, либо сегмент теряется. Затем сегмент передается на транспортный уровень.

## **Сервис прикладного уровня**

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 сообщение игнорируется, а чате отображается сообщение об ошибке.

WebSocket — это протокол связи, который обеспечивает непрерывное и двустороннее соединение между клиентом и сервером через одно TCP-соединение. Он позволяет обмениваться данными в режиме реального времени, что делает его идеальным выбором для чат-приложений. В чатах WebSocket используется для обеспечения мгновенной доставки сообщений между пользователями. При установлении соединения между клиентом и сервером через WebSocket, клиент и сервер могут отправлять и принимать сообщения в любое время без необходимости постоянного обновления страницы или выполнения дополнительных запросов. При использовании WebSocket в чат-приложениях, каждое сообщение, отправленное одним пользователем, мгновенно передается всем остальным участникам чата. Это позволяет пользователям видеть сообщения в режиме реального времени без необходимости обновления страницы или ожидания новых данных от сервера.

Благодаря непрерывному соединению WebSocket, чат-приложения могут предоставлять такие функции, как моментальная доставка сообщений, индикация онлайн-статуса других пользователей, а также уведомления о новых сообщениях. Это делает пользовательский опыт более интерактивным и удобным для всех участников чата. Кроме того, WebSocket обеспечивает эффективное использование ресурсов сервера, поскольку он позволяет установить одно постоянное соединение для передачи всех сообщений, вместо создания отдельного HTTP-запроса для каждого нового сообщения. Это снижает нагрузку на сервер и уменьшает задержки при передаче сообщений.

В целом, использование WebSocket в чат-приложениях обеспечивает быструю и надежную передачу сообщений в реальном времени, что делает его основным инструментом для создания современных интерактивных чат-систем.

WS сервер прикладного уровня написан на Swift[5] и фреймворке VAPOR[6]. Взаимодействие с транспортным уровнем происходит при помощи REST API. Взаимодействие мобильного приложения с WS сервером происходит через WebSocket протокол. Мобильное приложение написано при использовании фреймворка SwiftUI[7]. За основу дизайна выбрано мобильное приложение telegram.

Взаимодействие прикладного уровня с остальными компонентами системы отражено на диаграмме последовательности (Рисунок 4).

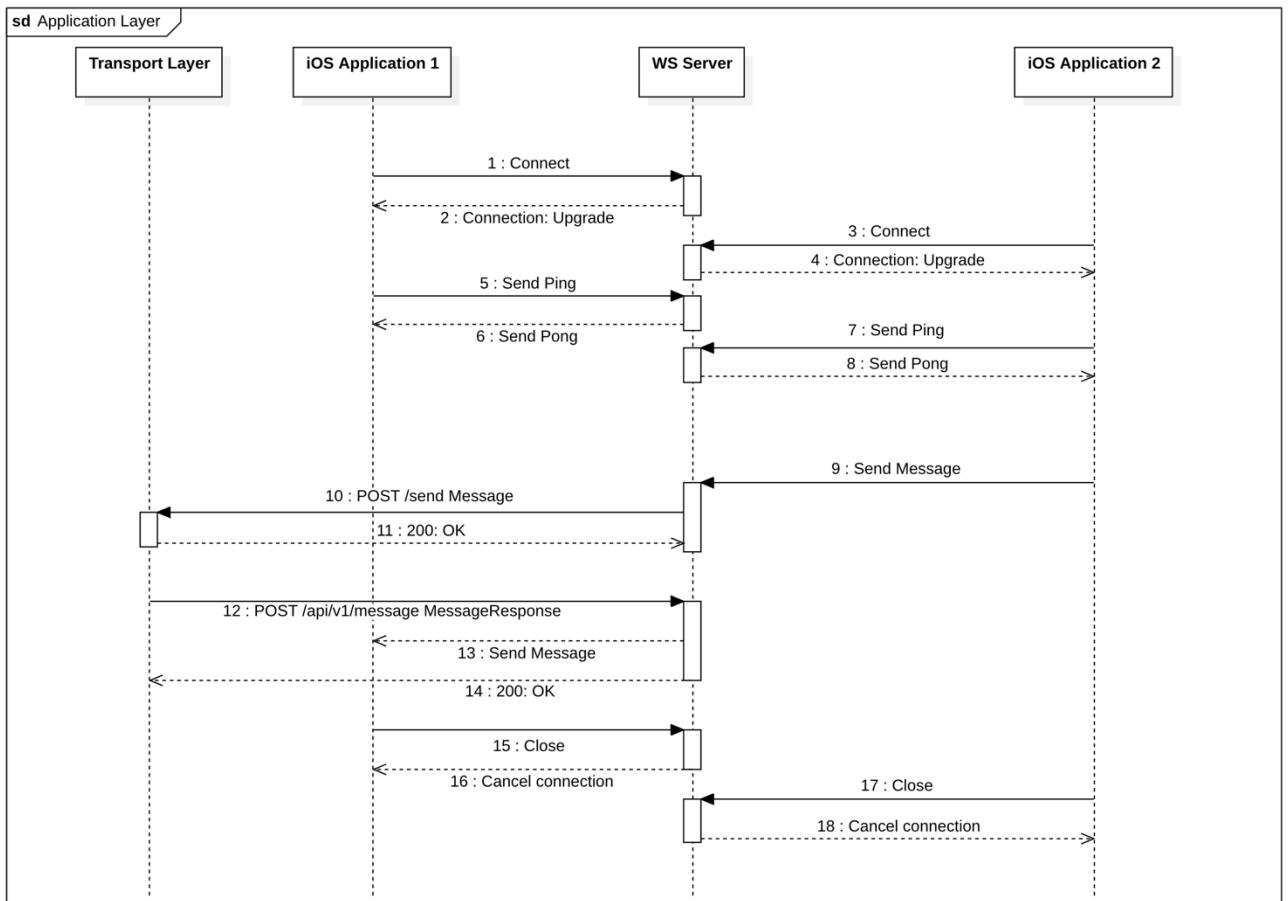
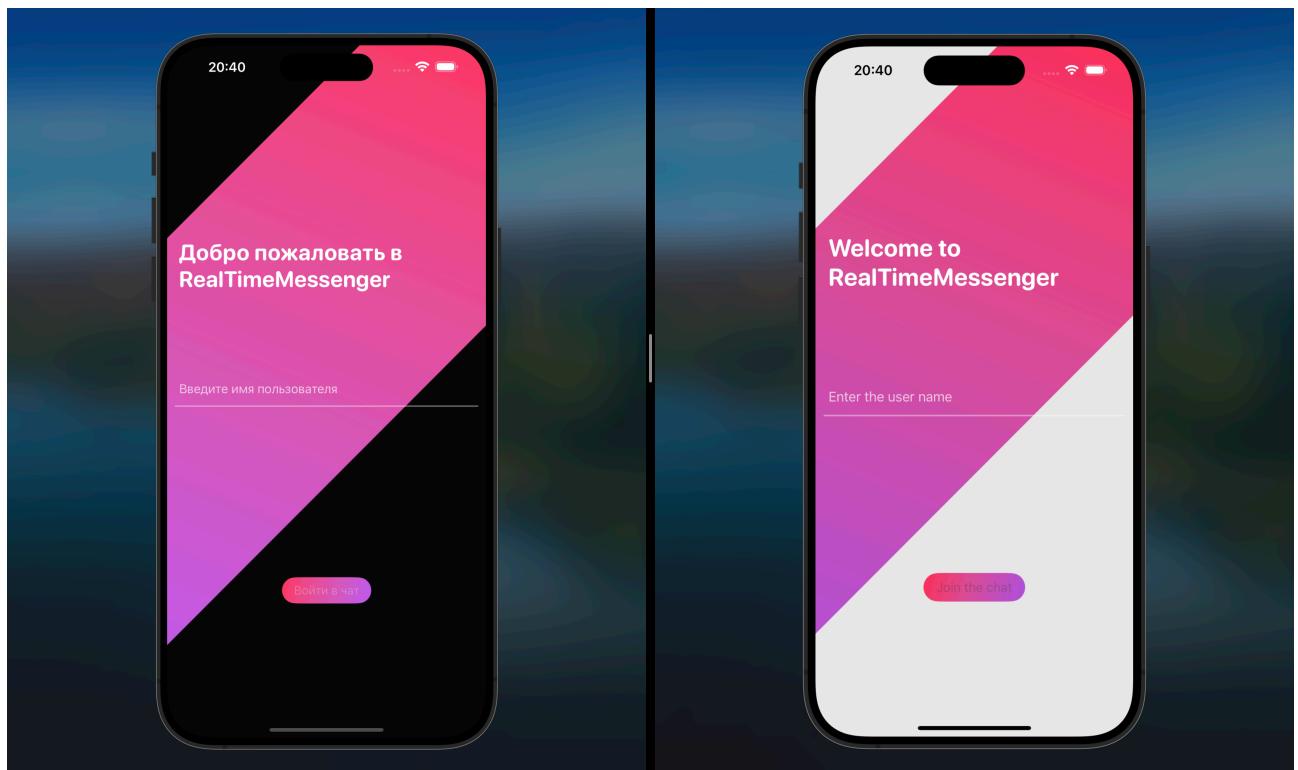


Рисунок 4 – диаграмма последовательности прикладного уровня

Пользователь вводит своё имя в текстовое поле, тем самым создавая WebSocket соединение с WS сервером. При успешном соединении пользователь отправляется на экран чата, где, используя поле ввода, пользователь может ввести текст сообщения, который передаётся по WebSocket соединению на WS сервер. WS сервер же преобразует объект типа string в json объект и по HTTP протоколу передаёт объект транспортному уровню. Когда WS сервер получает данные от транспортного по HTTP протоколу, WS сервер отправляет данные на сторону клиента.

**Таблица 3 – Методы прикладного уровня**

Метод	Входные параметры	Описание
POST handleMessageFromExternal- Service	data string? error string?	Прием декодированного сегмента с транспортного уровня на прикладной



**Рисунок 5 – Скриншот работы программы (авторизация)**

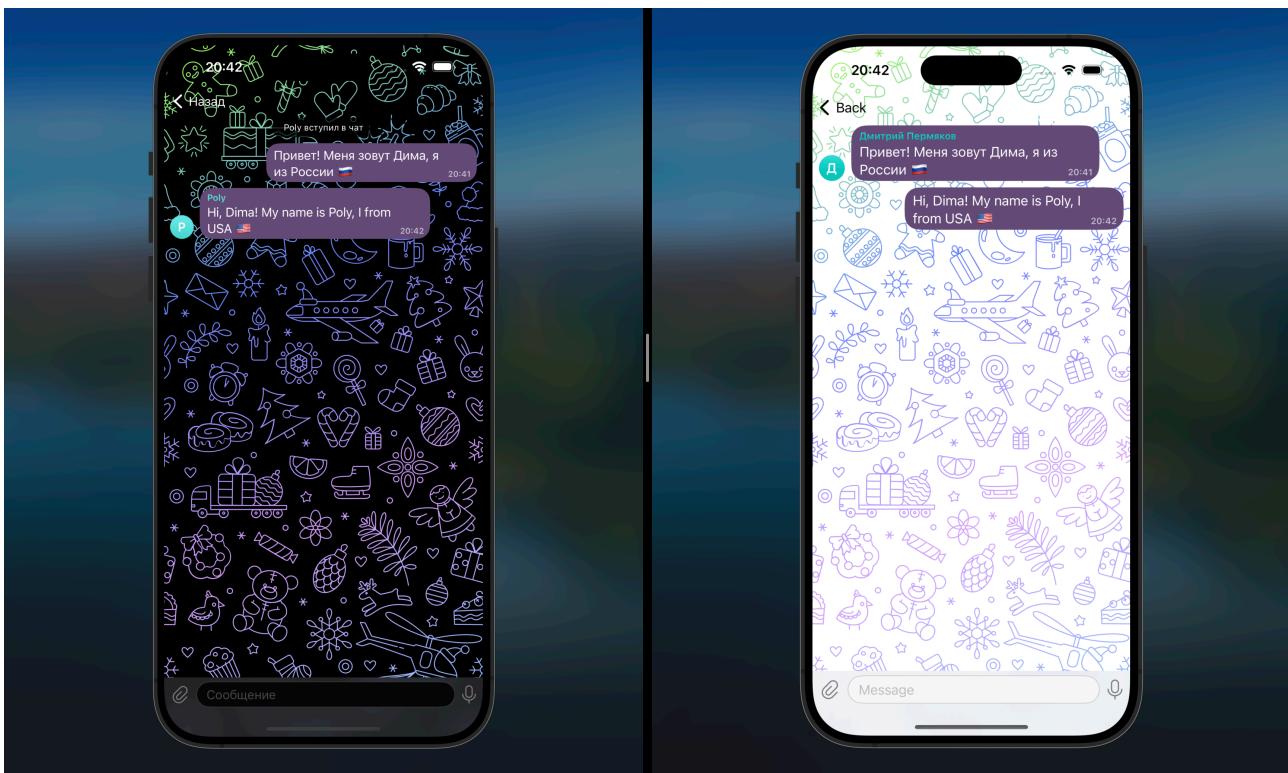


Рисунок 6 – Скриншот работы программы

## **Заключение**

В ходе курсовой работы был разработан мессенджер, обеспечивающий возможность обмена текстовыми сообщениями в реальном времени в удобном для пользователя интерфейсе. В проекте были выполнены следующие задачи:

1. Спроектирована логика взаимодействия между компонентами
2. Разработан дизайн копирующий интерфейс Telegram.
3. Созданы сервисы на Golang и Swift.
4. Создано мобильное приложения для IOS на SwiftUI.
5. Проведено тестирование отдельных компонентов.
6. Проведено интеграционное тестирование всей системы в сборе.
7. Подготовлен набор документации, включающий ТЗ, ПМИ, РП, РСА и диаграммы

## **Список использованных источников**

1. Институт развития интернета [Электронный ресурс] // Газета.ru.  
URL:  
<https://www.gazeta.ru/tech/2019/10/29/12782816/iri.shtml?ysclid=lvk57t3xhi190253574&updated> (дата обращения: 24.04.2024).
2. Левада-центр [Электронный ресурс] // Аналитический центр Юрия Левады. URL: <https://www.levada.ru/2021/02/23/sotsialnye-seti-v-rossii/> (дата обращения: 24.04.2024).
3. Руководство по Gin Web Framework [Электронный ресурс] // GoLang.  
URL: <https://github.com/gin-gonic/gin> (дата обращения: 13.04.2024).
4. Руководство по Sarama библиотека для работы с Apache Kafka.  
[Электронный ресурс] // GoLang. URL:  
<https://pkg.go.dev/github.com/Shopify/sarama> (дата обращения: 16.04.2024).
5. Swift Электронный ресурс // URL: <https://swiftbook.org/docs/tur-po-swift/?ysclid=lvk4fz83f7227689523> (дата обращения: 02.02.2024)
6. Vapor Swift Электронный ресурс // URL: <https://vapor.codes> (дата обращения: 12.02.2024)
7. SwiftUI Электронный ресурс // URL:  
<https://developer.apple.com/xcode/swiftui/> (дата обращения: 12.02.2024)

## **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 к17**

**Исполнители:**

**Пермяков Д.К. гр. ИУ5-63Б**

**Кузнецов В.А. гр. ИУ5-63Б**

**Рассказов Н.Д. гр. ИУ5-63Б**

**Преподаватель: Канев А.И.**

**2024г.**

## **1. Наименование**

Online мессенджер «RealTimeMessenger»

## **2. Основание для разработки**

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

## **3. Исполнители**

Исполнителями являются студенты МГТУ им. Н.Э. Баумана группы ИУ5-63Б:

1. Пермяков Д.К. (Прикладной уровень)
2. Кузнецов В.А. (Транспортный уровень)
3. Рассказов Н.Д. (Канальный уровень)

## **4. Цель разработки**

Реализовать систему обмена текстовыми сообщениями в реальном времени, состоящую из трех уровней: прикладной, транспортный и канальный.

## **5. Функциональные требования**

### **5.1. Прикладной уровень.**

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

5.1.1.1. Для подключения к чату пользователь должен ввести имя, которое будет передаваться с каждым новым сообщением;

5.1.1.2. После входа появляется возможность отправки сообщений по установленному WebSocket соединению с помощью поля ввода

сообщения и иконки отправить, которая появляется, когда поле ввода не пустое;

5.1.1.3. При отправке сообщения, у отправителя отображается сообщение с иконкой, что сообщение в процессе отправки;

5.1.1.4. В случае, если сообщение пришло с признаком ошибки, у отправителя иконка процесса меняется на красную иконку ошибки, другим пользователям сообщение не приходит;

5.1.1.5. В случае, если сообщение пришло без признака ошибки, у отправителя иконка процесса у сообщения меняется на иконку с двумя галочками. Другие пользователи получают это сообщение широковещательно.

5.1.1.6. Сообщения отправителя отображаются справа, другие сообщения пользователь видит слева;

5.1.1.7. Когда пользователь подключается к WebSocket соединению, в чате широковещательно отображается сообщение “<имя пользователя> вступил в чат”. В случае выхода, отображается сообщение “<имя пользователя> покинул чат”;

5.1.1.8. Дизайн приложения соответствует приложению Telegram;

5.1.1.9. Интерфейс приложения охватывает возможность локализации текста на английский и русский языки;

## 5.1.2. Веб-сокет 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 подключенными клиентам;

## 5.2. Транспортный уровень.

### 5.2.1. Реализация HTTP-метода Send для сегментирования сообщений:

5.2.1.1. Разбиение сообщения на сегменты по 120 байт и их поочередная отправка на канальный уровень через метод Code;

5.2.1.2. Каждый сегмент содержит:

5.2.1.2.1. Время отправки (в качестве идентификатора сообщения).

5.2.1.2.2. Общую длину сообщения.

5.2.1.2.3. Номер данного сегмента в сообщении.

5.2.1.2.4. Полезную нагрузку.

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

5.2.2.1. Форматирование очереди для полученных сегментов, которые раз в 2 секунды собираются в сообщения прикладного уровня;

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 10% в один случайный бит каждого сформированного кадра;

5.3.1.2. Сервис должен терять передаваемый кадр с вероятностью 1.5%;

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

5.3.2.1. Полученный от транспортного уровня json сегмента кодируется циклическим [7,4]-кодом 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. ОС: Linux Ubuntu 22.04.03+/Windows 10 22H2+/MacOS 14+

6.1.1.2. Swift (5.9 и выше)

6.1.1.3. Vapor 4.89.0

6.1.1.4. Docker (24.0.5 и выше)

6.1.1.5. Докер образы

6.1.2. Клиентская часть

6.1.2.1. iPhone (iOS 17 и выше)

6.1.2.2. Swift 5.9

6.2. Транспортный уровень

6.2.1. Ноутбук с ОС Windows (10 и выше)

6.2.2. Golang (1.21.5 и выше)

6.2.3. Docker (24.0.5 и выше)

6.2.4. Докер образы

6.2.4.1. confluentinc/zookeeper 3.7.2

6.2.4.2. confluentinc/cp-kafka 5.1.2-1

6.3. Канальный уровень

6.3.1. Ноутбук с ОС MacOS (Sonoma 14 и выше)

6.3.2. Golang (1.21.5 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 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 методичка испытаний**

**Вариант к17**

**Исполнители:**

**Пермяков Д.К. гр. ИУ5-63Б**

**Кузнецов В.А. гр. ИУ5-63Б**

**Рассказов Н.Д. гр. ИУ5-63Б**

**Преподаватель: Канев А.И.**

**2024г.**

## **1. Объект испытаний:**

Объектом испытания является online мессенджер «RealTimeMessenger».

## **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 стартовый экран приложения.
2	5.1.2.1	Заполнение поля имени, нажатие на кнопку «Войти в чат».	Осуществляется переход на страницу чата.
3	5.1.2.2	Отправка сообщений. Ввод текста сообщения и нажатие на кнопку «Отправить».	Отображение текста сообщения и времени отправки сообщения.
4	5.1.2.3	Получение сообщения от другого пользователя.	Отображение имени отправителя зелёным цветом, аватарки пользователя в виде первой

			буквы его никнейма, сообщения и времени отправки сообщения.
5	5.1.2.4	Вход/выход из чата	Отображение надписи <имя> вступил/покинул чат на чёрном/белом фоне в зависимости от выбранной темы приложения.
10	5.1.2.9	Выход из чата. Нажатие на кнопку «Назад» или свайп слева направо.	Осуществляется переход на стартовую страницу.
11	5.1.2.10	Отправка сообщения с ошибкой. Нажатие на иконку «Отправить».	Отображение текста “ошибка отправки сообщения” у всех пользователей в чате.
12	5.1.3	Запоминание имени для WebSocket подключения. Ввод имени и нажатие на кнопку «Войти в чат».	Открытие WebSocket соединения, запоминание имени пользователя.
13	5.1.4	Метод handleMessageFrom-ExternalService.	В json сообщения указывается отправитель, пэйлоад сообщения, время отправки, признак ошибки. Сообщение отправляется всем участникам чата.
14	5.2.1	Отправка POST запроса на сервер сегментации с сообщением через метод Send.	Сообщение разбивается на сегменты по 120 байт, которые отправляются на канальный уровень. В

			сегменте содержатся время отправки, общая длина сообщения, номер данного сегмента, полезная нагрузка.
15	5.2.2.1	Отправка Post запроса с пакетом-сегментом на сервер получения сегментов через метод Transfer.	Полученный сегмент помещается в очередь Kafka.
16	5.2.2.2	Добавление сегментов в очередь Kafka.	Раз в 2 секунды очередь проверяется. Если были получены все сегменты сообщения, то оно передаётся на прикладной уровень, иначе отправляется ошибка.
17	5.3.2	Метод Code.	Полученный от транспортного уровня json сегмента кодируется циклическим [7, 4] кодом в битовый формат. Вносится ошибка в 1 бит сегмента. Далее сегмент декодируется и либо исправляется ошибка, либо сегмент теряется. Затем сегмент передается на транспортный уровень.

## **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 к17**

**Исполнители:**

**Пермяков Д.К. гр. ИУ5-63Б**

**Кузнецов В.А. гр. ИУ5-63Б**

**Рассказов Н.Д. гр. ИУ5-63Б**

**Преподаватель: Канев А.И.**

**2024г.**

# **1. Введение**

## **1.1. Область применения**

Требования текстового сообщения 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 сообщениями между пользователями в реальном времени.

Распределенная система предоставляет возможность доступа к чату всем людям, которые открыли приложение и ввели свое имя в строку идентификации. При успешном соединении 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 данные 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 необходимо iOS устройство версии iOS17 и выше.

### **3. Условия выполнения программы**

Для работы программы требуется любое iOS устройство, поддерживающее версию ОС iOS17, а также стабильное интернет-соединение.

### **4. Выполнение программы**

#### **4.1. Инсталляция/дeинсталляция**

Потребуется iOS устройства поддерживающее 17 версию операционной системы, рекомендуется iPhone.

#### **4.2. Запуск программы**

На устройство необходимо скачать и открыть мобильное приложение RealTimeMessenger.

### **5. Описание операций**

#### **5.1. Идентификация в системе**

Доступно для: все пользователи iOS устройств.

Операция: открыть приложение.

Для идентификации в системе необходимо открыть приложение и ввести свое имя (рис.1).

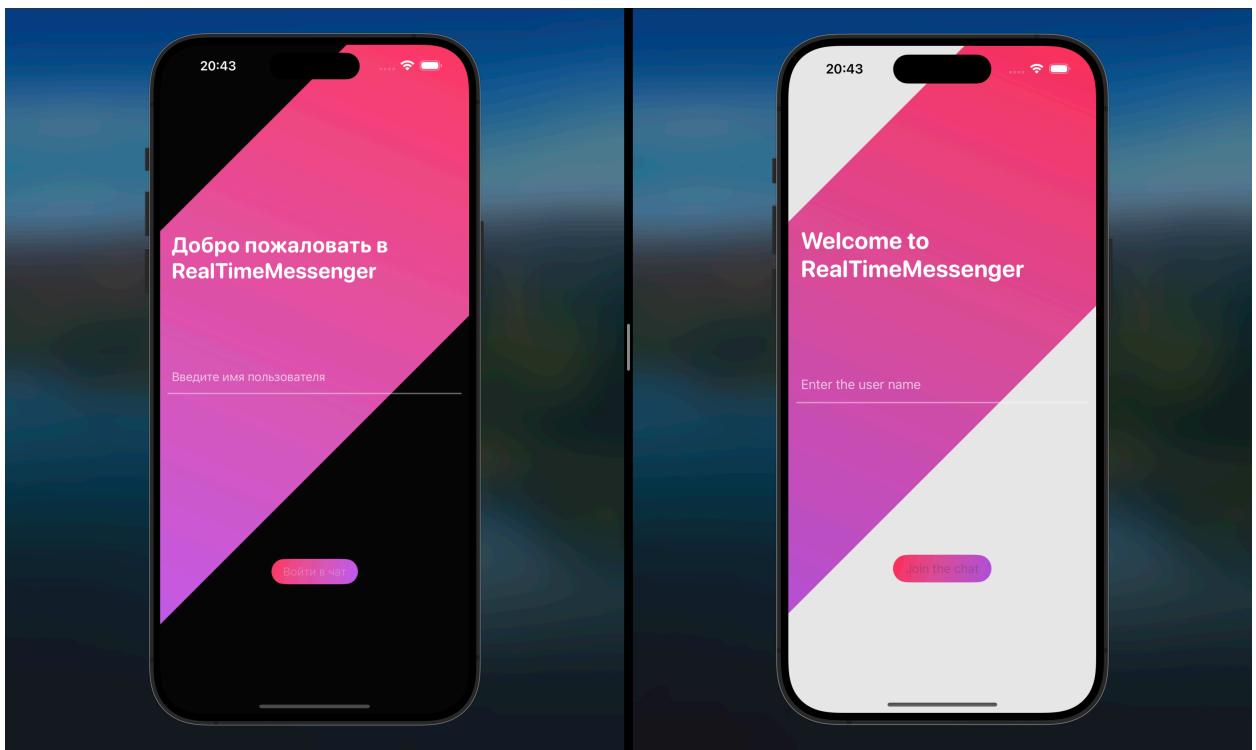


Рисунок 1 – окно идентификации

## 5.2. Просмотр и отправка сообщений

Доступно для: идентифицированных пользователей.

Операция 1: для отправки сообщения нужно вести его в специальное поле ввода сообщения и нажать на значок «Отправить» (рис.2).



Рисунок 2 – отправка сообщения

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



Рисунок 3 – экраны чата

### 5.3. Выход из чата

Доступно для: идентифицированных пользователей.

Операция: для выхода из чата необходимо нажать кнопку «Назад/Back» или сделай свайп слева направо (рис. 4) в левом верхнем углу экрана чата.

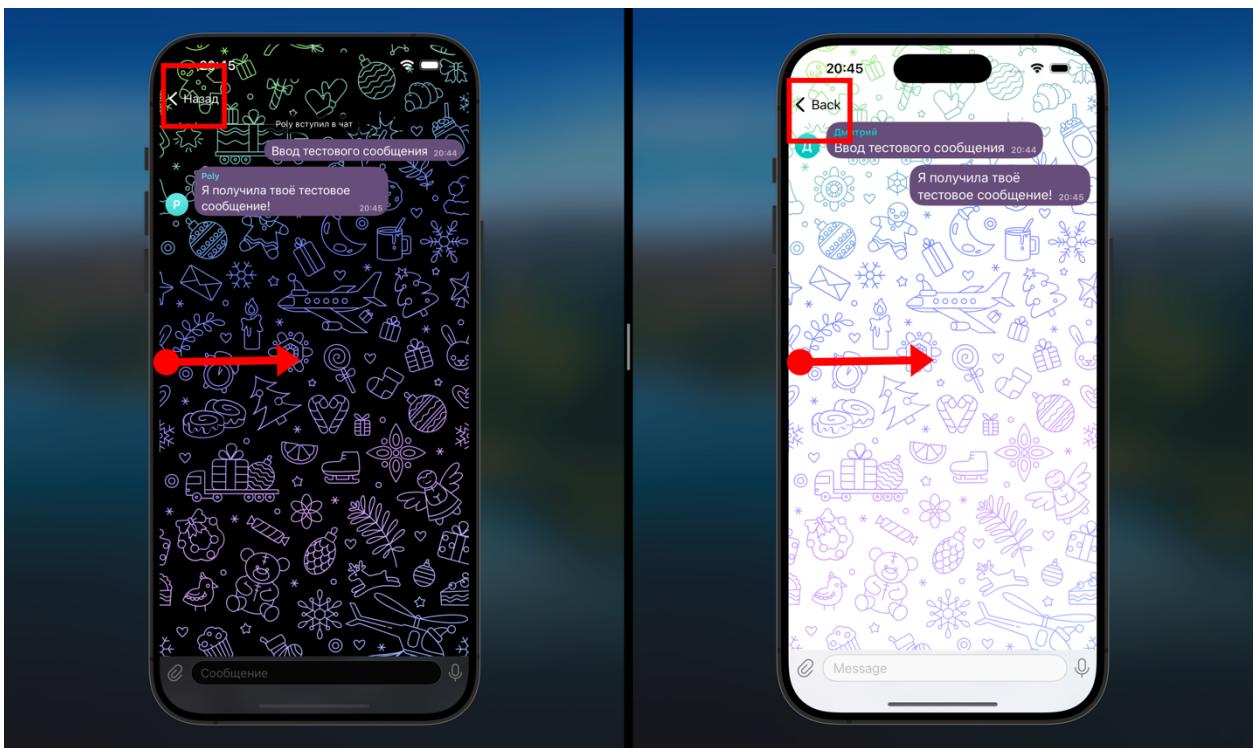


Рисунок 6 – Выход

## **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 к17**

**Исполнители:**

Пермяков Д.К. гр. ИУ5-63Б

Кузнецов В.А. гр. ИУ5-63Б

Рассказов Н.Д. гр. ИУ5-63Б

Преподаватель: Канев А.И.

**2024г.**

## **Системные требования**

### **1.1. Прикладной уровень**

#### **1.1.1. Требования к оборудованию**

- 1.1.1.1. Центральный процессор: с частотой от 1 ГГц
- 1.1.1.2. Оперативная память: 8 Гб и выше
- 1.1.1.3. Пространство на жестком диске: 1 Гб

#### **1.1.2. Требования к ОС**

- 1.1.2.1. Windows 10 или более поздняя версия
- 1.1.2.2. MacOS 12.7.4 или более поздняя версия
- 1.1.2.3. Ubuntu 22.04 или более поздняя версия

#### **1.1.3. Требования к ПО**

- 1.1.3.1. Swift 5.9 или более поздняя версия
- 1.1.3.2. Vapor 4.89.0
- 1.1.3.3. Docker 24.0.5 или более поздняя версия

### **1.2. Транспортный уровень**

#### **1.2.1. Требования к оборудованию**

- 1.2.1.1. Центральный процессор: с частотой от 1 ГГц
- 1.2.1.2. Оперативная память: 8 Гб и выше
- 1.2.1.3. Пространство на жестком диске: 1 Гб

#### **1.2.2. Требования к ОС**

- 1.2.2.1. Windows 10 или более поздняя версия
- 1.2.2.2. MacOS 12.7.4 или более поздняя версия
- 1.2.2.3. Ubuntu 22.04 или более поздняя версия

#### **1.2.3. Требования к ПО**

- 1.2.3.1. Golang 1.21 или более поздняя версия
- 1.2.3.2. Docker 24.0.5 или более поздняя версия

### **1.3. Канальный уровень**

#### **1.3.1. Требования к оборудованию**

- 1.3.1.1. Центральный процессор: с частотой от 1 ГГц
- 1.3.1.2. Оперативная память: 8 Гб и выше

1.3.1.3. Пространство на жестком диске: 1 Гб

### **1.3.2. Требования к ОС**

1.3.2.1. Windows 10 или более поздняя версия

1.3.2.2. MacOS 12.7.4 или более поздняя версия

1.3.2.3. Ubuntu 22.04 или более поздняя версия

### **1.3.3. Требования к ПО**

1.3.3.1. Golang 1.21 или более поздняя версия

## **2. Порядок развёртывания системы**

### **2.1. Прикладной уровень**

#### **2.1.1. WS Сервер**

2.1.1.1. Скачать код по ссылке <https://github.com/BMSTU-IU5-RealTimeMessenger/RealTimeMessengerAPI>

2.1.1.2. Запустить Docker

2.1.1.3. Выполнить команду

Docker-compose up -d

#### 2.1.2. Мобильное приложение

2.1.2.1. Скачать код по ссылке <https://github.com/BMSTU-IU5-RealTimeMessenger/RealTimeMessenger-iOS>

2.1.2.2. Открыть в Xcode

2.1.2.3. Выбрать необходимый девайс симулятора

2.1.2.4. Нажать кнопку Start

### **2.2. Транспортный уровень**

#### **2.2.1. Kafka**

2.2.1.1. Запустить Docker

2.2.1.2. Запустить сервис-координатор

```
docker run -d --name zookeeper -p 22181:2181 \
-e ZOOKEEPER_CLIENT_PORT=2181 \
-e ZOOKEEPER_TICK_TIME=2000 \
confluentinc/cp-zookeeper:latest
```

2.2.1.3. Запустить Kafka

```
docker run -d --name kafka --link zookeeper:zookeeper \
-p 29092:29092 -e KAFKA_BROKER_ID=1 \
-e KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 \
-e
KAFKA_ADVERTISED_LISTENERS=PLAINTEXT_HOST://localhost:29092 \
-e
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT \
-e KAFKA_INTER_BROKER_LISTENER_NAME=PLAINTEXT \
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
confluentinc/cp-kafka:latest
```

## **2.2.2. Сервер сегментации**

### **2.2.2.1. Скачать код из репозитория**

<https://github.com/BMSTU-IU5-RealTimeMessenger/RealTimeMessenger-Segmentation>

### **2.2.2.2. Перейти в папку проекта**

### **2.2.2.3. В файле «.env» в переменной «CHANNEL\_LAYER\_ADDR»**

**указать адрес сервера канального уровня**

### **2.2.2.4. Запустить программу при помощи команды:**

**go run main.go**

## **2.2.3. Сервер получения сегментов**

### **2.2.3.1. Скачать код из репозитория**

<https://github.com/BMSTU-IU5-RealTimeMessenger/RealTimeMessenger-Collection>

### **2.2.3.2. Перейти в папку проекта**

### **2.2.3.3. Запустить программу при помощи команды:**

**go run main.go**

## **2.2.4. Сервер сборки сообщений**

### **2.2.4.1. Скачать код из репозитория**

<https://github.com/BMSTU-IU5->

RealTimeMessenger/RealTimeMessenger-Assembly

2.2.4.2. Перейти в папку проекта

2.2.4.3. В файле «.env» в переменной

«APPLICATION\_LAYER\_ADDR» указать адрес сервера  
прикладного уровня

2.2.4.4. Запустить программу при помощи команды:

`go run main.go`

## 2.3. Канальный уровень

### 2.3.1. Сервер кодирования

2.3.1.1. Скачать код из репозитория

<https://github.com/BMSTU-IU5->

RealTimeMessenger/RealTimeMessenger-Channel

2.3.1.2. Перейти в папку проекта

2.3.1.3. Запустить программу