УДК 004.9

РАЗРАБОТКА МОДУЛЯ ВИДЕОКОНФЕРЕНЦИЙ ДЛЯ СИСТЕМЫ КОНТРОЛЯ И ОЦЕНКИ ЗНАНИЙ СТУДЕНТОВ LMSDOT

Свиткин Артур Эдуардович[1], Дедович Татьяна Григорьевна[2], Смирнов Даниил Павлович[3]

[1]Студент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: svitkin_ae@mail.ru.

[2]Кандидат физико-математических наук, доцент;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
Старший научный сотрудник;
Объединенный институт ядерных исследований;
Россия, 141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: tdedovich@jinr.ru.

[3]Аспирант;
Государственный университет «Дубна»;
Россия, 141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: smirnov@lmsdot.com

В статье проанализированы различные LMS-системы и их подход к проведению дистанционных занятий, сформированы требования к разрабатываемому модулю видеоконференций. Проанализированы различные архитектуры видеоконференций. Предложено архитектурное решение, которое впоследствии было реализовано и внедрено в Lmsdot. В рамках работы была изменена клиентская часть, создан сигнальный сервер, внедрён Pusher сервер, добавлен медиасервер mediasoup и использован его API для организации сигнального сервера. Проведено тестирование модуля видеоконференций. Модуль встроен в систему Lmsdot и показал стабильную и качественную работу.

Ключевые слова: электронное обучение, *LMS* системы, образование, видеоконференции, архитектуры видеоконференций, *WebRTC*, *mediasoup*.

Для цитирования:
Свиткин А. Э., Дедович Т. Г., Смирнов Д. П. Разработка модуля видеоконференций для системы контроля и оценки знаний студентов Lmsdot // Системный анализ в науке и образовании: сетевое научное издание. 2024. № 4. С. 68-78. EDN: IQNZTT. URL: https://sanse.ru/index.php/sanse/article/view/638.

DEVELOPMENT OF A VIDEO CONFERENCE MODULE FOR THE SYSTEM FOR MONITORING AND ASSESSING STUDENTS' KNOWLEDGE LMSDOT

Svitkin Arthur E.[1], Dedovich Tatyana G.[2], Smirnov Daniil P.[3]

[1]Student;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: svitkin_ae@mail.ru.

[2]PhD in Physical and Mathematical Sciences, associate professor;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
Senior Researcher;
Joint Institute for Nuclear Research;
6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

Статья находится в открытом доступе и распространяется в соответствии с лицензией Creative Commons «Attribution» («Атрибуция») 4.0 Всемирная (CC BY 4.0) https://creativecommons.org/licenses/by/4.0/deed.ru

e-mail: tdedovich@jinr.ru.
[3]PhD student;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: smirnov@lmsdot.com.

The article analyzes various LMS systems and their approach to conducting distance learning, and formulates the requirements for the module being developed. Various videoconferencing architectures have been analyzed. An architectural solution was proposed, which was subsequently implemented and implemented in Lmsdot. As part of the work, the client part was changed, a signaling server has been created, Pusher server implemented. The mediasoup media server has been added and its API has been used to organize a signaling server. The video conferencing module has been tested, the results of which confirm its readiness for use in the Lmsdot system; the module will provide stable and high-quality work during the learning process.

Keywords: e-learning, LMS systems, education, video conferencing, video conferencing architectures, WebRTC, mediasoup.

For citation:
Svitkin A. E., Dedovich T. G., Smirnov D. P. Development of a video conference module for the system for monitoring and assessing students' knowledge Lmsdot, *System analysis in science and education*, 2024;(4):68-78 (in Russ). EDN: IQNZTT. Available from: https://sanse.ru/index.php/sanse/article/view/638.

Введение

Современные тенденции в образовании предъявляют все более высокие требования к системам электронного обучения, обеспечивая эффективное и комфортное взаимодействие между преподавателями и студентами. Одним из важных аспектов этого взаимодействия является проведение дистанционных лекционных и семинарских занятий, которые стали неотъемлемой частью образовательного процесса.

В университете «Дубна», преподавателями Дедович Т.Г., Смирновым Д.П., Воробьёвым В.В., Беляковым В.А., Зинченко Д.А., во время организации учебного процесса по дисциплинам: «Дискретная математика», «Математическая логика и теория алгоритмов» и «Теория автоматов и формальных языков», «Теория языков программирования и методов трансляции», используется система контроля и оценки знаний студентов *Lmsdot* [1].

На данный момент система *Lmsdot* имеет текстовый чат, позволяющий преподавателям и студентам обмениваться сообщениями. Для повышения удобства коммуникаций между преподавателями и студентами в систему необходимо внедрить модуль видеоконференций. Модуль должен обеспечивать создание аудио- и видеовстреч, передачу аудио- и видеоинформации в режиме реального времени, а также поддерживать функции «демонстрация экрана» и «поднятие руки». Преподавателю должна быть предоставлена возможность управления пользователями. Важно обеспечить высокую надежность и безопасность передачи данных, удобство использования интерфейса, способность системы масштабироваться при увеличении числа пользователей и корректную работу даже при большом количестве одновременных подключений. В данной статье будет рассмотрен процесс создания модуля видеоконференций.

1. Анализ сервисов видеоконференций для LMS-систем

В работе проведен анализ сервисов видеоконференций, которые используются *LMS*-системами *Moodle* [2], *iSpring Learn* [3] и *Google Classroom* [4]. В системе *Moodle* для организации видеоконференций можно подключать сервисы *Jitsi* [5] и *Zoom* [6]. В системах *iSpring Learn* и *Google Classroom* используются сервисы *Zoom* и *Google Meet* [7], соответственно.

Сравнительный анализ был проведен по следующему набору характеристик: создание пространств для аудио- и видеовстреч, передача аудио- и видеоинформации потоково, поддержание количества участников встречи больше 100 человек, время проведения встречи больше 60 минут, функции «демонстрация экрана» и «поднятие руки», управление пользователями. Результаты

сравнительного анализа модуля коммуникации различных *LMS* систем представлены в таблице 1. Отметки «1» или «0» в таблице означают, что система обладает или не обладает данной характеристикой, соответственно.

Табл. 1. Характеристики бесплатных версий сервисов видеоконференций

	Jitsi (бесплатная)	*Zoom* (бесплатная)	*Google Meet* (бесплатная)
Создание пространств для аудио- и видеовстреч	1	1	1
Передача аудио- и видеоинформации потоково	1	1	1
Количество участников встречи >100	0	0	0
Время проведения встречи >60 минут	1	0	0
Функция «демонстрация экрана»	1	1	1
Функция «поднятие руки»	1	1	1
Управление пользователями	1	1	0

Из таблицы видно, что функции бесплатных версий сервисов видеоконференций ограничены. В них не поддерживается количество участников больше 100, *Zoom* не позволяет проводить встречи дольше одного часа, а *Google Meet* не позволяет гибко управлять пользователями (преподаватель не может выключить микрофон у участника конференции).

Отметим, что для разрабатываемого *web*-приложения важно иметь финансовую независимость, сохранять конфиденциальность передаваемых данных, не зависеть от запретов правообладателей на использование программного обеспечения и иметь полный функционал. Поэтому было принято решение разработать собственный модуль видеоконференций для системы *Lmsdot*.

2. Обзор медиасерверов

В начале необходимо выбрать медиасервер, так как он выполняет основные функции:

- кодирует аудио и видео потоки, поступающие от участников, в соответствии с требуемыми форматами и параметрами, а затем декодирует потоки для передачи другим участникам;
- маршрутизирует потоки, обеспечивает адаптивное потоковое вещания в зависимости от пропускной способности сети и устройств, а также контролирует задержки и качество передачи данных;
- синхронизирует аудио и видео потоки от разных участников, для обеспечения согласованного воспроизведения звука и изображения;
- устанавливает и завершает соединений между участниками;
- управляет звуком, изображением, переключением между различными источниками и режимами передачи данных.

В работе рассматривались медиасерверы *Janus Gateway* [8], *Kurento* [9], *Jitsi* [10] и *mediasoup* [11, 12]. Каждый из них имеет свои преимущества и подходит для различных сценариев использования в зависимости от конкретных требований, таких как масштабируемость, гибкость, простота внедрения и расширяемость. Первые три медиасервера ограничивают гибкость при реализации, предоставляя высокоуровневый *API*. *Mediasoup* и его клиентские библиотеки предоставляют низкоуровневый *API*. Они предназначены для реализации различных сценариев без каких-либо ограничений. Поэтому он был выбран для разработки приложения.

Mediasoup имеет открытый исходный код, специально разработанный для *WebRTC*, основанный на сетевой архитектуре *SFU* (англ. *Selective Forwarding Unit*, рус. модуль выборочной пересылки). Нижний уровень реализован на *C++*, а внешний – инкапсулирован в *Node.js*. Он предоставляет *API* для управления передачей видео и аудио в реальном времени. Архитектурные особенности *mediasoup* представлены на рис. 1.

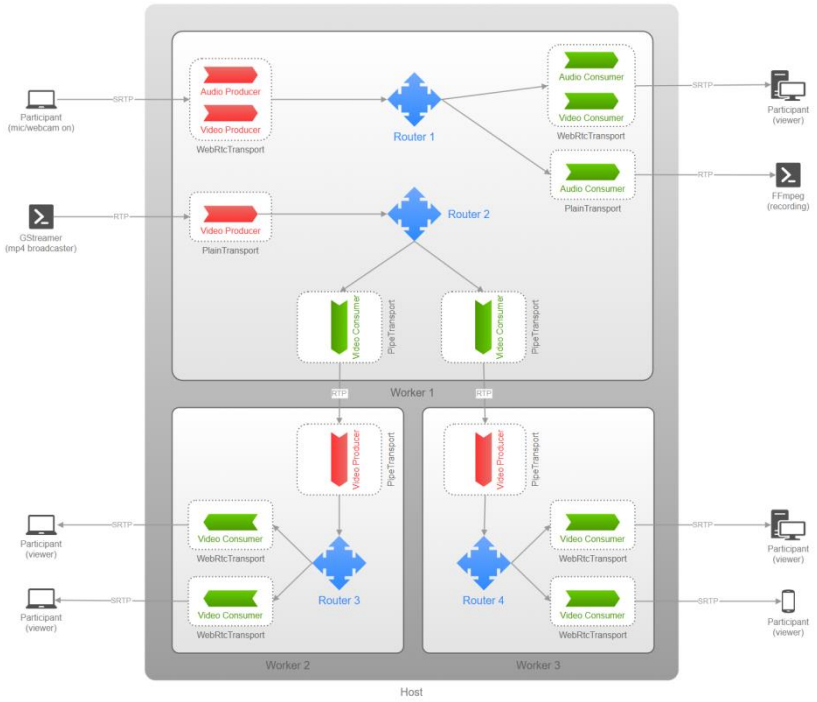

Рис. 1. Архитектура mediasoup

Протокол передачи имеет несколько уровней. Воркеры (большие белые прямоугольники) – это подпроцессы *C++* с одним потоком центрального процессора. Они могут включать в себя несколько маршрутизаторов (синие ромбы), которые действуют по принципу выборочной пересылки и отвечают за передачу мультимедиа между потребителями (зелёные стрелки) и отправителями (красные стрелки). Каналы *WebRTC* (белые прямоугольники, включающие в себя зеленые или красные стрелки) содержат потребителей и отправителей.

3. Технология передачи потоковых данных WebRTC

Рассмотрим технологию с открытым исходным кодом *WebRTC* (англ. *Web Real-Time Communication*, рус. коммуникация в режиме реального времени), предназначенную для организации передачи потоковых данных между браузерами или другими поддерживающими его приложениями по технологии *peer-to-peer* [13]. *WebRTC* – охватывает множество существующих технологий единого назначения и объединяет их в упрощенный, широко применимый пакет, структура которого показана на рис. 2. Описание каждого элемента и их функциональность опишем ниже.

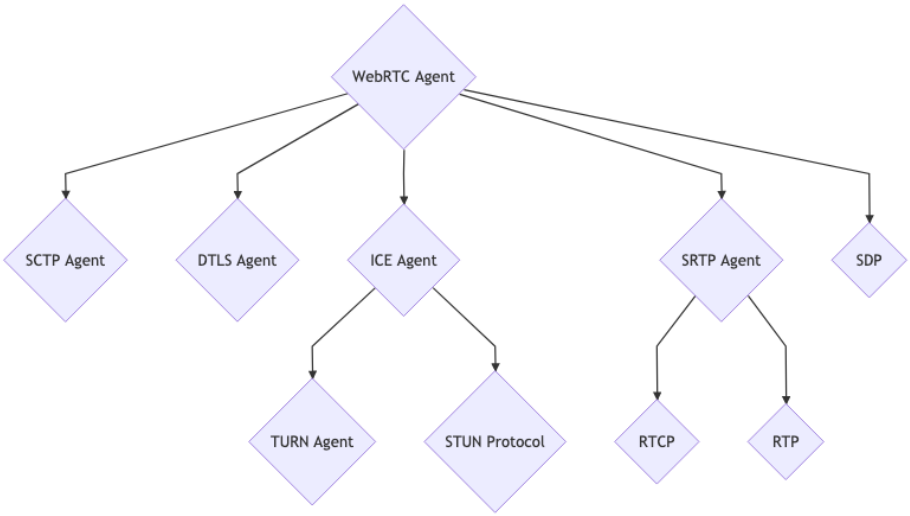

Рис. 2. Структура WebRTC

Технология *WebRTC* – состоит из четырёх последовательных шагов: сигнализация, подключение, безопасность и общение.

Сигнализация обеспечивает инициализацию подключения (обмен сигналами). Для этого используется текстовый протокол *SDP* (англ. *Session Description Protocol*, рус. протокол описания сессии). Каждое *SDP* сообщение состоит из пар ключ/значение: *IP*-адреса и порты агентов, количество используемых аудио- и видеодорожек, аудио- и видеокодеки агентов, уникальный идентификатор, секретный ключ, сертификат безопасности. Передача сигналов обычно происходит отдельно и не использует *WebRTC*. Любая архитектура, подходящая для отправки сообщений, может ретранслировать *SDP* между подключающимися одноранговыми узлами.

Подключение *WebRTC* использует протокол *ICE* (англ. *Interactive Connectivity Establishment*, рус. установка интерактивного подключения). Он позволяет устанавливать прямое соединение между двумя агентами без центрального сервера и использует *NAT* (англ. *Network Address Translation*) для присваивания устройству публичного *IP*-адреса. Этот механизм в сетях *TCP/IP* изменяет *IP*-адрес в заголовке пакета, проходящего через устройство маршрутизации трафика. Запросы транслируются от частного *IP*-адреса устройства к публичному *IP*-адресу маршрутизатора (с уникальным портом). Для обеспечения преобразования сетевых адресов используются технологии *STUN* и *TURN*. Протокол *STUN* (англ. *Session Traversal Utilities for NAT*, рус. утилиты обхода сеансов для *NAT*) позволяет клиенту, находящемуся за сервером трансляции адресов, определить свой внешний *IP*-адрес, способ трансляции адреса и порта во внешней сети. Некоторые маршрутизаторы («Симметричный *NAT*») имеют ограничения на то, кто может подключаться к устройствам в сети, например, принимают соединения только от узлов, к которым ранее подключались. В этой ситуации используют *TURN* (англ. *Traversal Using Relays around NAT*, рус. обход с использованием реле вокруг *NAT*). Клиент создаёт соединение с *TURN* сервером и сообщает всем узлам команду – слать пакеты этому серверу, которые затем будут переправлены ему.

Безопасность обеспечивается с помощью двух протоколов *DTLS* (англ. *Datagram Transport Layer Security*, рус. протокол датаграмм безопасности транспортного уровня) и *SRTP* (англ. *Secure Real-Time Transport Protocol*, рус. безопасный транспортный протокол реального времени). *DTLS* обеспечивает защищённость соединений для протоколов, использующих датаграммы. Он предотвращает перехват, прослушивание, вмешательство, не нарушая защиты целостности данных или подделку содержимого сообщения. Сначала он подключается, завершая установку *DTLS* над соединением, установленным *ICE*. *WebRTC* проверяет, что сертификат, переданный по *DTLS*, соответствует сертификату, переданному при сигнализации. Это *DTLS*-соединение затем используется для получения и отправки сообщений. Далее *WebRTC* использует протокол *RTP* (англ. *Real-time Transport Protocol*, рус. транспортный протокол реального времени), защищенный с помощью *SRTP*, для передачи аудио- и видеоданных.

Общение основывается на двух уже существующих протокола: *RTP* и *SCTP* (англ. *Stream Control Transmission Protocol*, рус. протокол передачи с управлением потоком). *RTP* используется для обмена медиаданными (аудио- и видеопотоки), зашифрованными с помощью *SRTP*. *RTP* – протокол прикладного уровня, предоставляющий необходимые инструменты для реализации потоковой передачи данных в реальном времени. Он позволяет разработчикам справляться с задержками, потерей пакетов и перегрузками по своему усмотрению. *SCTP* используется для отправки и получения сообщений, зашифрованных с помощью *DTLS*. *SCTP* – протокол транспортного уровня, обеспечивающий передачу данных. Он позволяет обеспечить необходимую задержку для систем реального времени.

4. Анализ архитектур видеоконференций

Любая система видеоконференцсвязи (ВКС) состоит из:

- Терминалов ВКС – устройств, поддерживающих передачу видео и аудио;
- Сервера ВКС, который необходим для проведения групповых видеоконференций;
- Инфраструктуры – каналов связи, а также устройств для передачи данных, трансляции и записи видеоконференции;
- Периферийного оборудования – спикерфонов, микшеров, микрофонов, *PTZ*-камер.

В таблицах 2 и 3 представлены характеристики различных архитектур видеоконференций для четырех участников на терминале и сервере, полученные компанией «Труконф» [14]. Сравнивались следующие архитектуры:

- *MCU* (*Multipoint Control Unit*) – объединение всех потоков в один;
- *SFU* (*Selective Forwarding Unit*) – передача потоков от каждого участника;
- *Simulcast* – передача множественных потоков разных разрешений;

Табл. 2. Характеристики архитектур видеоконференций для четырех участников на терминале

На терминале для 4-х участников	*MCU*	*SFU*	*Simulcast*
Количество исходящих потоков	1	1	3
Количество входящих потоков	1	3	3
Исходящий канал, Мб/с	1,0	1,0	1,5
Входящий канал, Мб/с	1,0	3,0	1,0
Нагрузка на ЦП	20%	60%	80%

Табл. 3. Характеристики архитектур видеоконференций для четырех участников на сервере

На сервере для 4-х участников	*MCU*	*SFU*	*Simulcast*
Количество исходящих потоков	4	12	12
Количество входящих потоков	1	4	12
Исходящий канал, Мб/с	4,0	12,0	4,0
Входящий канал, Мб/с	4,0	4,0	6,0
Нагрузка на ЦП	100%	0%	0%

Из таблицы видно, что архитектура *MCU* плохо показывает себя при большом количестве пользователей, так как в этом случае нагрузка на центральный процессор сервера очень высока. Архитектура *Simulcast* имеет обратную проблему, при возрастании количества участников перегружается терминал. При использовании архитектуры *SFU* не нужен широкий исходящий канал для клиента, и она нетребовательна к ресурсам сервера.

По итогам анализа была выбрана архитектура *SFU* как наиболее устойчивая к большому количеству одновременных подключений и устоявшаяся в *IT* сообществе.

5. Требования к модулю видеоконференций

Система должна предоставлять следующие функциональные возможности:

- создавать пространства для аудио- и видеовстреч;
- передавать аудио- и видеоинформацию потоково;
- использовать функцию «демонстрация экрана» и «поднятие руки»;
- преподавателю управлять пользователями (отключать микрофон и удалять участников).

Перечислим нефункциональные требования к модулю видеоконференций:

- высокая надежность и безопасность передачи данных;
- простота и понятность интерфейса;
- масштабируемость системы при увеличении числа пользователей;
- корректная работа модуля при ~200 одновременных подключений.

6. Модернизация структуры Lmsdot

В ходе модернизации приложения *Lmsdot* была разработана архитектура решения, представленная на рис. 3.

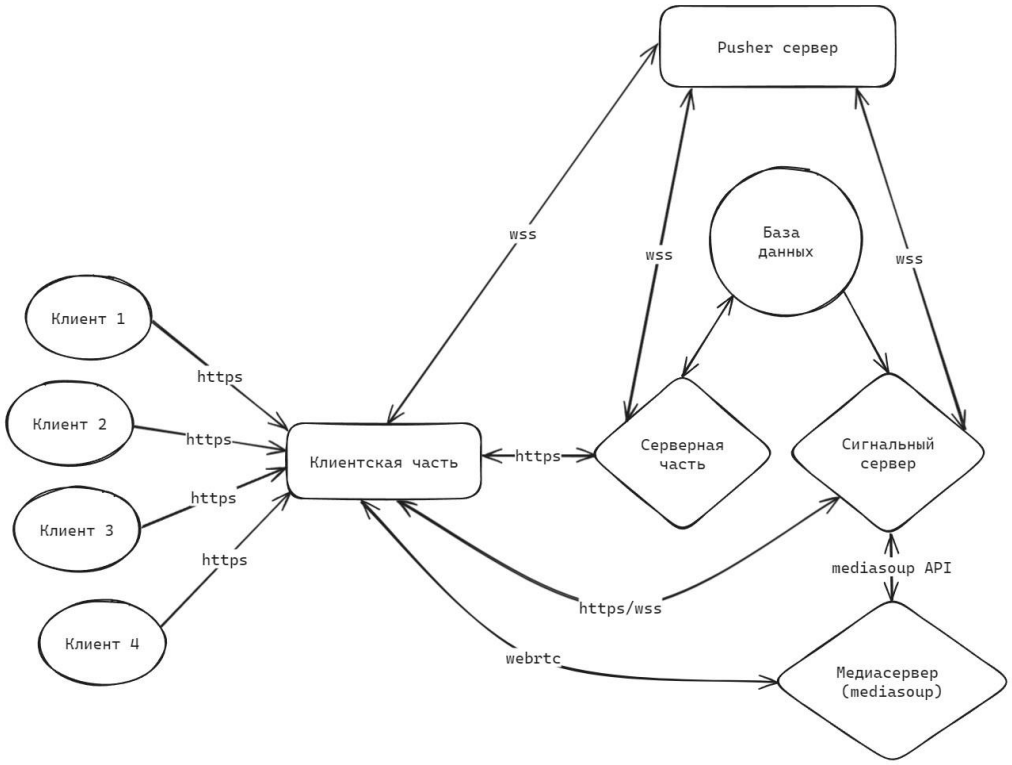

Рис. 3. Архитектура системы Lmsdot с модулем видеоконференций

В работе изменена клиентская часть для отображения модуля видеоконференций. Создан сигнальный сервер для обеспечения сигнализации агентов *WebRTC* и отслеживания их состояния. Внедрён *Pusher* сервер [15] для автоматического обновления данных между клиентской и серверными частями. Добавлен медиасервер *mediasoup* и использован его *API* для организации сигнального сервера.

Pusher сервер обеспечивает двустороннее соединение между различными компонентами системы, позволяя им обмениваться данными в реальном времени. Он используется для отправки уведомлений, обновлений и другой информации между клиентской и серверной частями системы, обеспечивая мгновенную обратную связь и интерактивность.

Сигнальный сервер обеспечивает сигнализацию агентов *WebRTC* и отслеживает их состояние. Он играет важную роль в установлении и поддержании соединения между клиентами для передачи данных в реальном времени по протоколу *WebRTC*, обеспечивая стабильность и надежность передачи медиаданных. Сигнальный сервер берёт на себя функции:

1. подключение к каналу передачи медиаданных;
2. инициализации получателей медиаданных;
3. прерывание потока получения медиаданных;
4. создание отправителя медиаданных;
5. прекращение отправки медиаданных;
6. получение информации о созданном пространстве встречи;
7. присоединение к пространству встречи;
8. покидание пространства встречи.

7. Клиентская часть

Для описания взаимодействия между пользователями системы и самой системой была выбрана диаграмма вариантов использования, представленная на рис. 4.

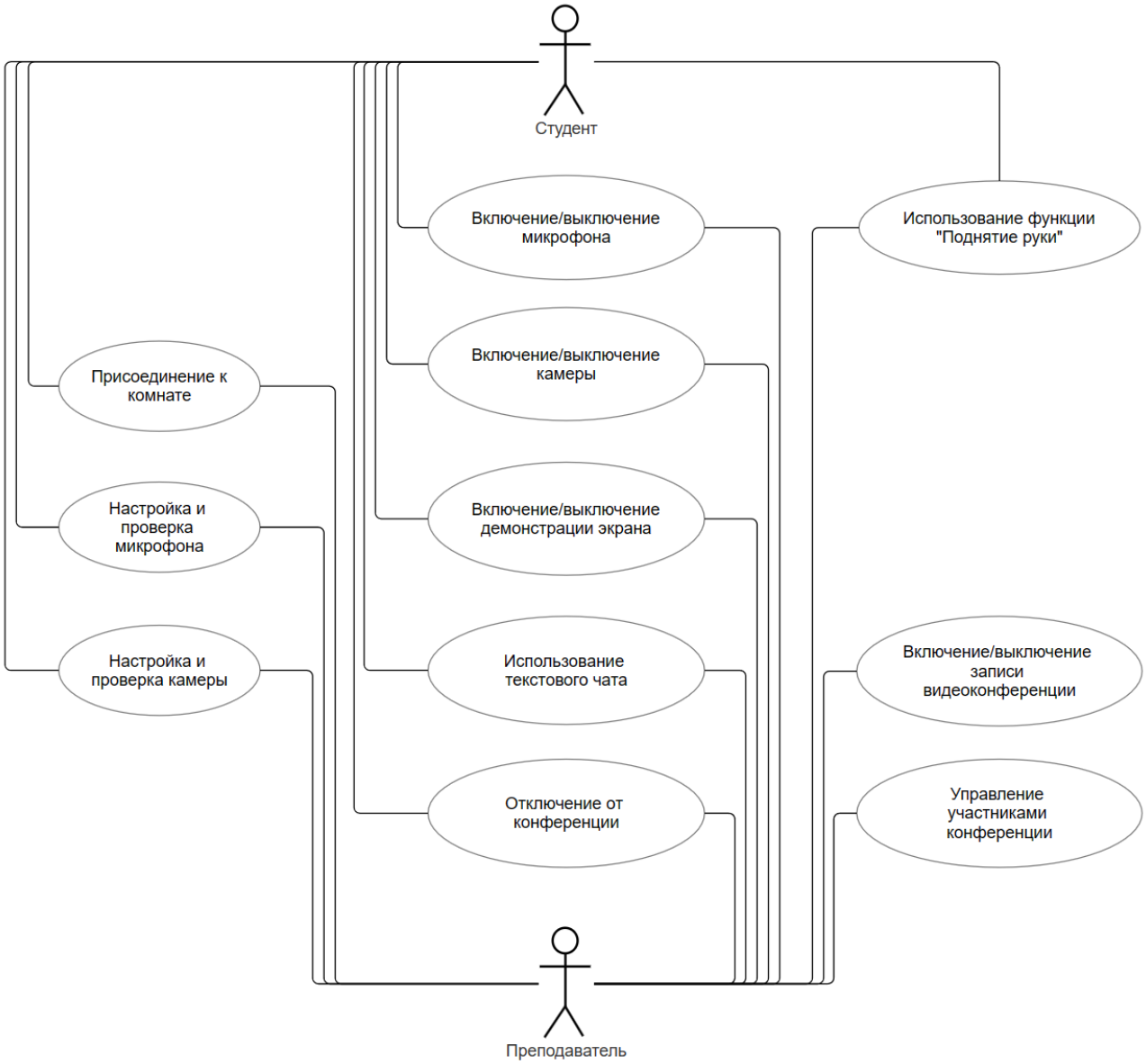

Рис. 4. Модель процессов клиентской части

На диаграмме отображены следующие общие функции для обоих акторов (преподаватель, студент): «Присоединение к комнате», «Настройка и проверка микрофона», «Настройка и проверка камеры», «Включение/выключение микрофона», «Включение/выключение камеры», «Включение/выключение демонстрации экрана», «Поднятие руки», «Использование текстового чата», «Отключение от конференции». Уникальными функциями для преподавателя являются «Включение/выключение записи видеоконференции» и «Управление участниками конференции».

Экран подключения к комнате

Макет экрана подключения к комнате представлен на рис. 5 и предоставляет следующую функциональность: «Присоединение к комнате», «Настройка и проверка микрофона», «Настройка и проверка камеры».

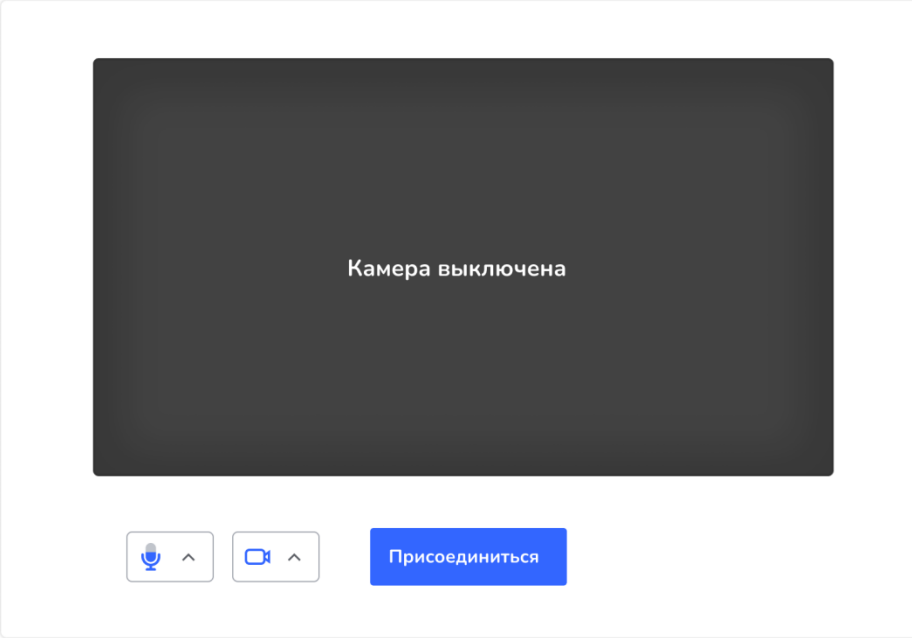

Рис. 5. Подключение к комнате

Экран видеоконференции

Макет экрана видеоконференций представлен на рис. 6. В левой части экрана показана панель активностей семинара, в неё входят: «Звонок» (активная вкладка), «Задания», «Решения», «Доска» и «Настройки».

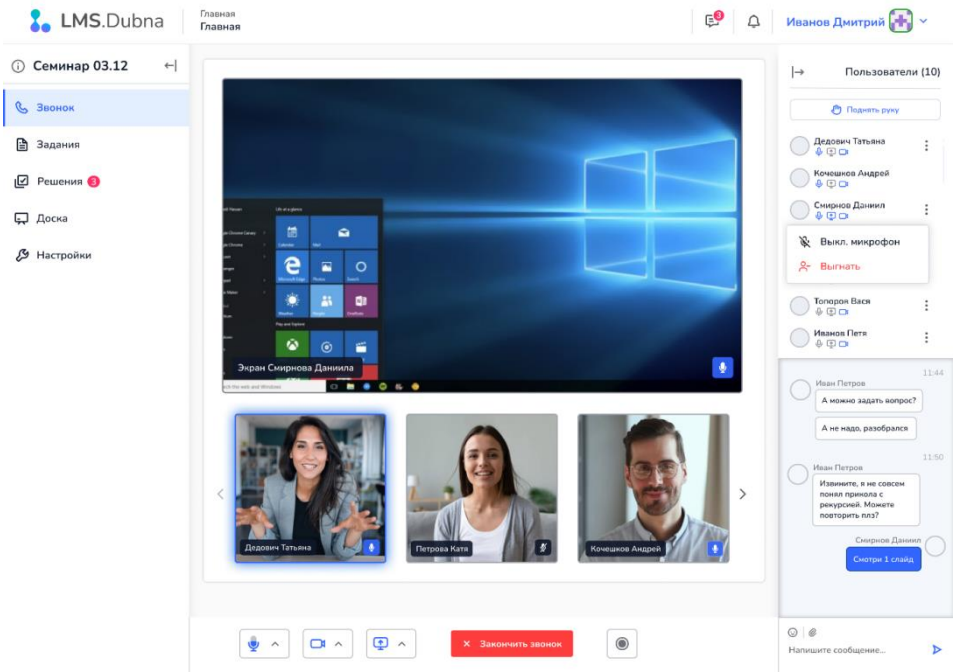

Рис. 6. Видеоконференция

По центру экрана представлены участники конференции, отметим, что основную часть занимает демонстрация экрана участника. В нижней части представлен набор действий. Перечислим действия слева направо и опишем их функциональность: «Включение/выключение микрофона», «Включение/выключение камеры», «Включение/выключение демонстрации экрана», «Отключение от конференции», «Включение/выключение записи видеоконференции». В правой части сверху отображён список участников, через него осуществляется поднятие руки пользователем. В правой

части ниже был встроен текстовый чат для оперативного обмена сообщениями во время видеоконференции.

8. Инструменты реализации

- Медиасервер – *mediasoup*;
- Технология передачи потоковых данных – *WebRTC*;
- Языки программирования – *JavaScript*, *TypeScript*;
- Серверный фреймворк – *ElysiaJS* [16] (среда исполнения *BunJS* [17]);
- Клиентский фреймворк – *React* [18].

9. Проверка на практике

Внедрение модуля видеоконференций было выполнено в марте 2024 года. Преподавателями Дедович Т.Г. и Смирновым Д.П. проводилось ручное тестирование модуля в рамках курсов «Теория автоматов и формальных языков» и «Математическая логика и теория алгоритмов». При проведении ручного тестирования, производилась сверка фактических результатов и ожидаемых, описанных в тест-кейсах.

После проведения тестирования были выявлены и устранены некоторые дефекты визуального отображения участников. Результаты тестирования модуля видеоконференций подтверждают его готовность к использованию в системе *Lmsdot*, модуль обеспечит стабильную и качественную работу в процессе обучения.

Заключение

Целью данной работы является расширение возможностей блока коммуникаций в системе контроля и оценки знаний студентов *Lmsdot*, разработанной в университете «Дубна». В ходе работы были проанализированы другие *LMS*-системы и их подход к проведению дистанционных занятий, сформированы требования к разрабатываемому модулю. Подготовлена теоретическая база для понимания предметной области, а именно сетевого взаимодействия, используемых протоколов, архитектур видеоконференцсвязи и медиасерверов.

Проведён анализ системы *Lmsdot* и на его основе предложено архитектурное решение, которое впоследствии было реализовано программно и внедрено в *Lmsdot*. В рамках работы была изменена клиентская часть для отображения модуля видеоконференций. Создан сигнальный сервер для обеспечения сигнализации агентов *WebRTC* и отслеживания их состояния. Внедрён *Pusher* сервер для автоматического обновления данных, между клиентской и серверными частями. Добавлен медиасервер *mediasoup* и использован его *API* для организации сигнального сервера.

Проводилось ручное тестирование модуля в рамках курсов «Теория автоматов и формальных языков» и «Математическая логика и теория алгоритмов». Тестирование проведено по описанным тест-кейсам. Результаты тестирования подтвердили его готовность к использованию.

Разработанный модуль удобен в использовании и расширяет возможности блока коммуникаций для преподавателей и студентов. Таким образом, данная работа имеет практическую ценность для студентов и преподавателей, которые используют *Lmsdot* в своей работе.

Список источников

1. Кочешков А. Д., Смирнов Д. П., Дедович Т. Г. Разработка информационной системы контроля и оценки знаний студентов по математическим дисциплинам в университете «Дубна» // Системный анализ в науке и образовании. – 2021. – № 2. – С. 140–150. – URL: http://sanse.ru/download/442.
2. Moodle : [веб-сайт]. – URL: https://moodle.org/ (дата обращения: 01.12.2023).

3. iSpring Learn : [веб-сайт]. – ООО «Ричмедиа», 2001-2023. – URL: https://www.ispring.ru/ (дата обращения: 01.12.2023).
4. Google Classroom : [веб-платформа]. – URL: https://classroom.google.com (дата обращения: 01.12.2023).
5. Free Video Conferencing Software for Web & Mobile | Jitsi : [веб-сайт]. – URL: https://jitsi.org/ (дата обращения: 04.12.2023).
6. Zoom : [веб-сайт]. – Zoom Communications, Inc., 2023. – URL: https://zoom.us/ru (дата обращения: 06.12.2023).
7. Google Meet : [веб-платформа]. – URL: https://meet.google.com/ (дата обращения: 01.10.2023).
8. Janus Gateway : [репозиторий проекта] // GitHub : [веб-платформа]. – GitHub, Inc., 2023. – URL: https://github.com/meetecho/janus-gateway (дата обращения: 01.12.2023).
9. Kurento overview : [Fora Soft: веб-сайт]. – URL: https://forasoft.github.io/kurento-media-server-overview/ (дата обращения: 07.12.2023).
10. Jitsi : [репозитории проекта] // GitHub : [веб-платформа]. – GitHub, Inc., 2023. – URL: https://github.com/orgs/jitsi/repositories (дата обращения: 05.12.2023).
11. mediasoup : [веб-сайт]. – URL: https://mediasoup.org/ (дата обращения: 09.12.2023).
12. mediasoup : [репозиторий проекта] // GitHub : [веб-платформа]. – GitHub, Inc., 2023. – URL: https://github.com/versatica/mediasoup (дата обращения: 09.12.2023).
13. Peer-to-peer / [Университет ИТМО Wiki]. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Peer-to-peer (дата обращения: 08.12.2023).
14. TrueConf. RU : Заглавная страница // TrueConf Wiki : [веб-проект]. – URL: https://trueconf.ru/blog/wiki (дата обращения: 08.12.2023).
15. Pusher : [веб-сайт]. – Pusher Ltd., 2024. – URL: https://pusher.com (дата обращения: 09.02.2024).
16. ElysiaJS: Elysia - Ergonomic Framework for Humans. – URL: https://elysiajs.com (дата обращения: 01.02.2024).
17. Bun – A fast all-in-one JavaScript runtime. – URL: https://bun.sh (дата обращения: 20.01.2024).
18. React : [веб-сайт]. – Meta Platforms, Inc.[1], 2024. – URL: https://react.dev (дата обращения: 05.02.2024).

[1] Деятельность Meta Platform Inc. на территории РФ запрещена.