

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

*К защите допустить:*

Заведующий кафедрой ПОИТ

\_\_\_\_\_ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к дипломному проекту  
на тему

**ПЛАТФОРМА ДЛЯ ПРОВЕДЕНИЯ ОНЛАЙН-ЗАНЯТИЙ И  
ВСТРЕЧ НА СТЕКЕ LARAVEL И REACT.JS**

БГУИР ДП 1-40 01 01 037 ПЗ

Студент

Е. А. Головченко

Руководитель

Д. В. Аврамец

Консультанты:

*от кафедры ПОИТ*

Д. В. Аврамец

*по экономической части*

А. А. Горюшкин

Нормоконтролер

П. Н. Красковский

Рецензент

Минск 2024

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра Программное обеспечение информационных технологий  
(наименование кафедры)

УТВЕРЖДАЮ  
Заведующий кафедрой  
Лапицкая Н. В.  
«    » 2024 г.

**ЗАДАНИЕ**  
**на дипломный проект (дипломную работу)**

Обучающемуся Головченко Егору Антоновичу  
Курс 4 Учебная группа 051003 Специальность 1-40 01 01

1 Тема дипломного проекта (дипломной работы):

Платформа для проведения онлайн-занятий и встреч на стеке Laravel и React.Js

Утверждена приказом ректора от « 7 » марта 2024 г. № 512-с

2 Исходные данные к дипломному проекту (дипломной работе):

Операционная система – Windows 11. Языки программирования – PHP, JavaScript.

Перечень выполняемых функций: регистрация пользователей, наличие личного кабинета для зарегистрированных пользователей, редактирование профиля зарегистрированных пользователей, создание и модерация видеозвонков, просмотр списка запланированных видеозвонков, создание запланированных видеозвонков, вход в комнату видеозвонка по ссылке либо номеру комнаты, отключение собственной камеры либо микрофона во время звонка, доска для общей работы участников видеозвонка, демонстрация экрана для всех участников видеозвонка. Назначение разработки: продукт для проведения занятий в онлайн-формате, обычных и деловых встреч.

3 Содержание расчетно-пояснительной записи (перечень подлежащих разработке вопросов):

Введение

1 Анализ аналогов и литературных источников в сфере приложений для видеоконференций и видеозвонков

2 Моделирование предметной области и разработка функциональных требований

3 Проектирование программного средства

4 Создание программного средства

5 Тестирование программного средства

6 Руководство по установке и использованию

7 Технико-экономическое обоснование разработки и внедрения ПС

Заключение

Список использованных источников

Приложение А (обязательное) Исходный код серверной части

Приложение Б (обязательное) Исходный код клиентской части

Приложение В (обязательное) Исходный код веб-сокет сервера

4 Перечень графического материала (с точным указанием обязательных чертежей и графиков):

Платформа для проведения онлайн-занятий и встреч на стеке Laravel и React.Js. Схема программы – формат А1, лист 1.

Подключение нового пользователя к комнате. Схема алгоритма – формат А1, лист 1.

Демонстрация экрана другим участникам видеовстречи. Схема алгоритма – формат А1, лист 1.

Диаграмма вариантов использования. Плакат – формат А1, лист 1.

Графическое представление базы данных. Плакат – формат А1, лист 1.

Диаграмма классов серверной части программного средства. Плакат – формат А1, лист 1.

Диаграмма развертывания. Плакат – формат А1, лист 1.

5 Консультанты по дипломному проекту (дипломной работе):

Консультант от кафедры: Аврамец Д. В.

Консультант по экономической части: Горюшкин А. А.

Нормоконтролёр: Красковский П. Н.

6 Примерный календарный график выполнения дипломного проекта (дипломной работы):

Анализ предметной области, разработка технического задания – 25.03–01.04

Разработка функциональных требований, проектирование архитектуры ПС – 02.04–13.04

Разработка схемы программы, алгоритмов, данных: 14.04–21.04

Разработка программного средства: 22.04–05.05

Тестирование и отладка: 06.05–13.05

Оформление пояснительной записки и графического материала: 14.05–31.05

Дата выдачи задания 25 марта 2024 г.

Срок сдачи законченного дипломного проекта (дипломной работы) 31 мая 2024 г.

Руководитель дипломного проекта  
(дипломной работы)

(подпись)

(фамилия, инициалы)

Подпись обучающегося \_\_\_\_\_

Дата 25 марта 2024 г.

## **РЕФЕРАТ**

**ПЛАТФОРМА ДЛЯ ПРОВЕДЕНИЯ ОНЛАЙН-ЗАНЯТИЙ И ВСТРЕЧ НА СТЕКЕ LARAVEL И REACT.JS: дипломный проект / Е.А. Головченко – Минск: БГУИР, 2024, пз – п.з. – 98 с., чертежей (плакатов) – 7 л. формата А1.**

Объектом разработки является платформа для проведения различного рода онлайн-занятий и встреч.

Целью работы является предоставление инструментов для проведения видеовстреч и взаимодействия во время них с соответствием с потребностями пользователя.

Проведено изучение литературных источников предметной области, а также анализ существующих программных средств. Была разработана спецификация функциональных требований для разрабатываемого программного средства. Основными функциями являются создание комнаты для проведения видеовстречи управление запланированными звонками. В качестве дополнительных возможностей необходимо рассмотреть демонстрацию своего экрана другим участниками звонка во время проведения видеовстречи, управление доской для совместного использования, возможность модерации звонка для создателя беседы, переключения камеры и микрофона, подключение по ссылке либо номеру комнаты. Программное средство также имеет разделение пользовательских ролей на зарегистрированного и незарегистрированного пользователя. Роль пользователя определяет, какие разделы системы будут ему доступны. Безопасность пользовательских данных обеспечивается с помощью ограничения прав доступа к пользовательским данным с использованием системы пользовательских сессий, основанной на токенах.

На этапе проектирования определена архитектура программного средства, а также разработаны основные алгоритмы, обеспечивающие работу требуемой функциональности, изучены особенности работы с различными протоколами и стандартами для организации передачи потоковых данных между браузерами.

Выполнено тестирование разработанного программного средства, которое подтвердило его работоспособность.

Приведено технико-экономическое обоснование эффективности разработки и внедрения программного продукта.

На основе функциональности программного средства составлено руководство по использованию.

Дипломный проект является завершённым, поставленная задача решена в полной мере. Разработанное программное средство в дальнейшем может быть усовершенствовано добавлением новых функциональных возможностей и оптимизацией имеющихся.

Дипломная работа прошла проверку в системе антиплагиата. Уникальность составляет 97,97%.

## СОДЕРЖАНИЕ

Введение.....	6
1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству .....	8
1.1 Анализ литературных источников .....	8
1.2 Анализ существующих аналогов.....	12
1.3 Формирование требований к проектируемому программному средству.....	20
2 Анализ требований к программному средству и разработка спецификации функциональных требований.....	25
2.1 Описание функциональности программного средства.....	25
2.2 Спецификация функциональных требований.....	26
3 Проектирование программного средства .....	29
3.1 Разработка программной архитектуры.....	29
3.2 Разработка модели базы данных .....	31
3.3 Разработка алгоритмов программного средства.....	32
4 Разработка программного средства.....	39
4.1 Взаимодействие с базой данных .....	39
4.2 Реализация пользовательского интерфейса .....	41
4.3 Разработка сервиса аутентификации .....	42
4.4 Разработка websocket сервиса .....	45
4.5 Разработка сервиса запланированных встреч .....	46
5 Тестирование программного средства.....	47
6 Руководство по установке и использованию .....	54
6.1 Установка программного средства .....	54
6.2 Руководство пользователя.....	54
7 Технико-экономическо обоснование разработки и использования платформы для проведения онлайн-занятий и встреч на стеке Laravel и React.Js.....	62
Заключение .....	69
Список использованных источников .....	70
Приложение А (обязательное) Исходный код серверной части .....	72
Приложение Б (обязательное) Исходный код клиентской части.....	78
Приложение В (обязательное) Исходный код веб-сокет сервера .....	94

## **ВВЕДЕНИЕ**

В последние годы прослеживается тенденция на рост работы в удалённом и гибридном формате. Современные компании всё больше переводят своих сотрудников на подобный формат трудовой деятельности. В особенности это касается области информационных технологий, где проявляется наиболее явный переход сотрудников на гибридный либо удалённый формат работы. Вместе с этим возрастает спрос на онлайн-образование. Различными площадками для обучения пользуются лучшие мировые университеты и популярность таких платформ будет только расти. Во многом этому поспособствовала пандемия, которая обосновала переход почти всех сфер жизни человека в онлайн-формат и создала отличные условия для развития различного рода технологий. Одним из наиболее важных и актуальных направлений в этом контексте является технология для проведения видеоконференций, благодаря которой возникает всё больше возможностей для общения и взаимодействия на расстоянии.

Видеоконференции стали неотъемлемой частью бизнеса, образования и даже личного общения, позволяя людям со всего мира встречаться, общаться, обмениваться файлами и своими наработками без необходимости физического присутствия. Однако, несмотря на широкое распространение, многие существующие платформы для проведения видеоконференций имеют ряд недостатков, среди которых проблемы с безопасностью, недостаточная функциональность либо сложность в использовании. В особенности отмечу, что современные платформы обладают либо недостаточной функциональностью во время проведения онлайн-занятия или встречи, при этом достаточно просты в использовании, либо обладают достаточно полным, но при этом сильно пересложнённым функционалом, что сказывается на высокой стоимости данных платформ и дополнительному обучению персонала.

Целью данной дипломной работы является разработка платформы для проведения онлайн-занятий и встреч, которая будет простой в использовании, иметь весь необходимый функционал для проведения качественных и эффективных встреч между людьми в виртуальном пространстве, при этом безопасной для пользователей. В ходе работы будет проведен анализ существующих решений, определены требования к новой платформе, разработана архитектура системы и реализован прототип.

## ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

*Фреймворк* – программное обеспечение, облегчающее разработку и объединение различных компонентов программного средства.

*Браузер* – прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов.

*Веб-приложение* – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер.

*Сессия* – время от подсоединения пользователя к серверу до обрыва связи или отсоединения.

*Протокол* – набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами.

IP – маршрутизируемый протокол сетевого уровня стека TCP/IP.

RFC – документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и стандарты, широко применяемые во всемирной сети.

TCP – протокол для создания надёжного виртуального полнодуплексного соединения между процессами.

UDP – протокол для обеспечения доставки дейтограмм без подтверждения их получения.

NAT – это механизм в сетях TCP/IP, позволяющий преобразовывать IP-адреса транзитных пакетов.

HTTP – протокол прикладного уровня передачи произвольных данных.

REST – набор общепринятых концепций по построению веб ориентированных приложений.

API – описание способов, включающих в себя набор классов, процедур, функций, структур или констант, которыми одно программное средство может взаимодействовать с другим средством.

JSON – стандартизованный текстовый формат для представления структурированных данных.

URL – стандартизованный и общепринятый формат указания локации расположения веб-ресурса в сети Интернет.

ORM – технология программирования, связывающая базы данных с концепциями объектно-ориентированных языков программирования.

SQL – декларативный язык программирования, применяемый для создания, управления и модификации данных в реляционной базе данных.

# **1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**

## **1.1 Анализ литературных источников**

Видеоконференция – это вид телекоммуникаций между двумя и более абонентами, который позволяет им видеть и слышать друг друга независимо от разделяющего их расстояния [1]. Для организации видеоконференций используется специализированная технология – видеоконференцсвязь. Общение в режиме видеоконференций также называют сеансом видеоконференцсвязи.

Видеоконференцсвязь (ВКС) – телекоммуникационная технология, обеспечивающая организацию видеоконференций между двумя и более абонентами по сети передачи данных. Во время сеанса видеоконференцсвязи обеспечивается интерактивный обмен звуком и изображением. Также абоненты могут транслировать телеметрические данные, компьютерные данные, демонстрировать документы и объекты с использованием дополнительных видеокамер.

Создание платформы для проведения видеоконференцсвязи становится возможным благодаря открытому стандарту WebRTC.

WebRTC (Web Real-Time Communication – коммуникация в режиме реального времени) – это API (Application Programming Interface – программный интерфейс приложения) и протокол [2]. Протокол WebRTC – это набор правил, позволяющий двум агентам WebRTC (браузерам) вести двунаправленную безопасную коммуникацию в реальном времени. WebRTC API позволяет разработчикам использовать протокол WebRTC. WebRTC API в настоящее время определен только для JavaScript. Протокол WebRTC поддерживается рабочей группой rtcweb IETF [3]. WebRTC API задокументирован в W3C как webrtc [4]. В процессе установки соединения с помощью WebRTC можно выделить этапы сигнализации, установки соединения, безопасности, коммуникации. Переходы между этапами происходят последовательно. Обязательным условием для начала следующего этапа является успешное завершение предыдущего. Каждый этап состоит из большого количества других протоколов. Для создания WebRTC было объединено множество существующих технологий. В этом смысле WebRTC – это комбинация и конфигурация хорошо известных технологий, появившихся в начале 2000-х годов.

Первым этапом является сигнализация. При запуске WebRTC агент не знает, с кем и по поводу чего будет происходить коммуникация. И именно сигнализация дает нам прозрачность. Это первый этап и его назначение – подготовка вызова для того, чтобы два агента WebRTC могли начать коммуникацию. Сигнализация осуществляется с помощью существующего протокола SDP (Session Description Protocol – протокол описания сессии). SDP – это аббревиатура от Session Description Protocol – протокол описания сеанса связи

[5]. Session Description Protocol определяет параметры обмена мультимедийными данными (как правило, потоковыми) между двумя конечными точками и опубликован IETF в документе RFC 4566. Session Description Protocol обычно инкапсулируется в другой протокол – наиболее широко он используется с протоколом SIP в приложениях IP-телефонии. Простыми словами, Session Description Protocol – это список возможностей конечной точки, где перечислены ее параметры и технологии приема данных. Типичный список возможностей включает, какой IP-адрес должен принять входящий медиапоток, какой номер порта должен принять медиапоток, тип медиа, который ожидает конечная точка, протокол, по которому конечная точка ожидает обмен информацией и метод кодирования, с которым может работать конечная точка. Этим списком декларирование не ограничивается и может включать и другие параметры. В установлении сессии участвуют две конечные точки, которые обмениваются списком спецификаций и возможностей, в соответствии со стандартом SDP. Session Description Protocol сам по себе не передает никаких мультимедиа-данных, а просто служит для согласованием общего, приемлемого для обеих сторон, набора параметров обмена. А сами медиапотоки передаются по другим протоколам. Сигнализация, как правило, происходит вне WebRTC. WebRTC, обычно, не используется для передачи сигнальных сообщений. Для передачи SDP между подключенными парами могут использоваться такие технологии, как конечные точки REST, веб-сокеты.

REST, или репрезентативная передача состояний – это архитектурный стиль для обеспечения стандартов между компьютерными системами в Интернете, облегчающий системам взаимодействие друг с другом [6]. Системы, совместимые с REST, часто называемые RESTful системами, характеризуются тем, что они не имеют состояния и разделяют задачи клиента и сервера. В архитектурном стиле REST реализация клиента и сервера могут выполняться независимо, без ведома друг друга. Это означает, что код на стороне клиента может быть изменен в любое время, не влияя на работу сервера, а код на стороне сервера может быть изменен, не влияя на работу клиента. Пока каждая сторона знает, в каком формате сообщения отправлять другой стороне, они могут быть модульными и отдельными. Отделяя проблемы пользовательского интерфейса от проблем хранения данных, мы повышаем гибкость интерфейса на разных платформах и улучшаем масштабируемость за счет упрощения серверных компонентов. Кроме того, разделение позволяет каждому компоненту развиваться независимо. Используя интерфейс REST, разные клиенты обращаются к одним и тем же конечным точкам REST, выполняют одни и те же действия и получают одинаковые ответы.

Протокол WebSocket, описанный в спецификации RFC 6455 [7], обеспечивает возможность обмена данными между браузером и сервером через постоянное соединение. Данные передаются по нему в обоих направлениях в виде пакетов, без разрыва соединения и дополнительных HTTP-запросов. WebSocket особенно хорош для сервисов, которые нуждаются в постоянном обмене данными, например онлайн игры, торговые площадки, работающие в

реальном времени.

В процессе сигнализации агенты WebRTC получают достаточно информации для того, чтобы попытаться выполнить подключение. Для этого используется другой протокол под названием ICE [8]. Протокол последовательно собирает данные о возможности создания локального соединения между устройствами (пирами) по TCP или UDP. Если такой возможности нет, он проверяет возможность использования STUN во внешней сети, если и такой возможности не обнаруживается, ICE проверяет возможность использовать TURN во внешней сети (в том числе и проверяет доступность подключения к нескольким ближайшим TURN-серверам). После окончания сбора информации о структуре окружающей клиента сети, клиентом, создающим сессию отправляется сообщение на TURN-сервер, содержащее полученную архитектуру ближайшей сети, а так же кидает запрос на создание сессии для пира, хранящий пачку IP-адресов, которые знает клиент. Пир соответственно отвечает множественными STUN-сообщениями по каждому из IP. Рано или поздно, хотя бы одно из сообщений прилетит инициатору сессии, проскочив сквозь NAT. Как только клиент ответит на все такие сообщения, у пира появится карта сети по которой будет выбираться самый легкий для прохождения маршрут и выполниться проверка доступности между пиром и клиентом. Процесс работы протокола ICE представлен на рисунке 1.1.

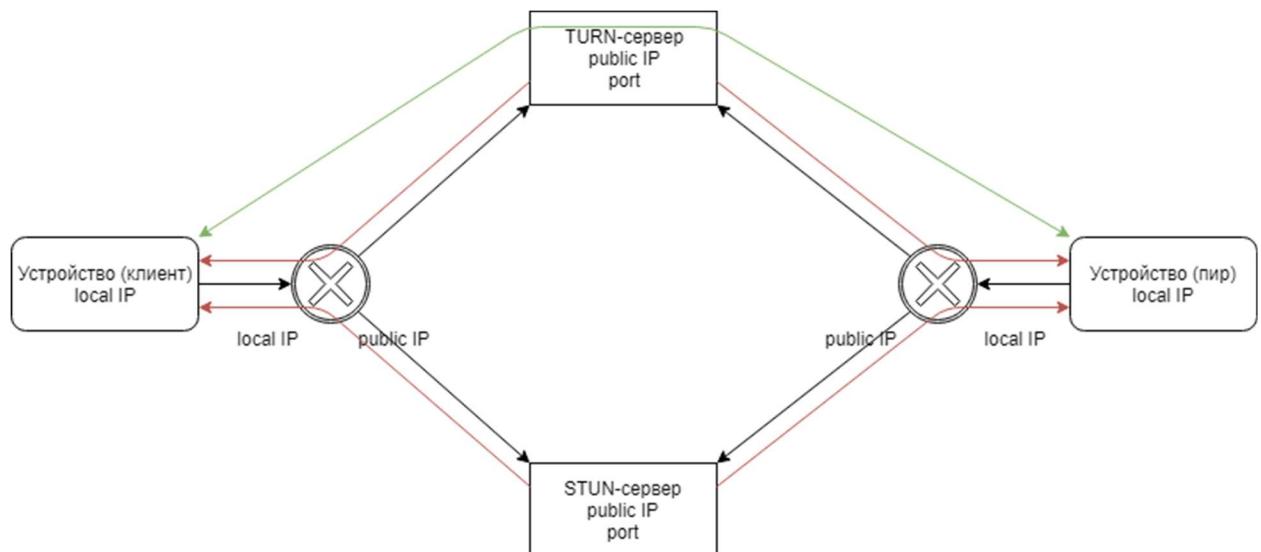


Рисунок 1.1 – Процесс работы протокола ICE

Session Traversal Utilities for NAT (STUN) – это протокол для нахождения и определения публичного адреса клиента и любых ограничений в маршрутизаторе, которые препятствуют прямому соединению с узлом [9]. Клиент отправляет запрос к STUN серверу в интернете, который отвечает публичным адресом клиента и, доступен ли, или нет, клиент за NAT маршрутизатором. Процесс работы протокола STUN представлен на рисунке 1.2.



Рисунок 1.2 – Процесс работы протокола STUN

Traversal Using Relays around NAT (TURN) предназначен для обхода ограничения симметричного NAT путём открытия соединения с TURN сервером и ретрансляции всей информации через этот сервер. Создаётся соединение с TURN сервером и сообщите всем узлам слать пакеты этому серверу, которые затем будут переправлены вам. Очевидно, что они приходят с некоторыми накладными расходами, поэтому это используется, только если нет других альтернатив. Схема работы протокола TURN представлена на рисунке 1.3.

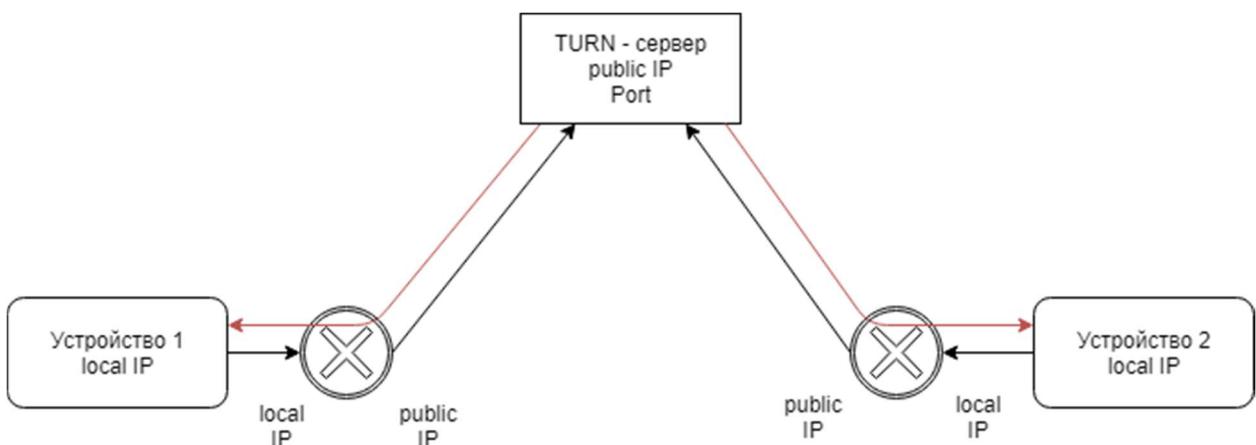


Рисунок 1.3 – Процесс работы протокола TURN

После успешного подключения ICE WebRTC приступает к установке зашифрованного транспортного канала. Он используется для передачи аудио, видео и других данных. Это делается с помощью двух протоколов, также разработанных задолго до появления WebRTC. Первым является протокол DTLS (Datagram Transport Layer Security) – это протокол, разработанный для обеспечения безопасности связи [10]. Протокол DTLS основан на потоковом протоколе Transport Layer Security (TLS) и обеспечивает безопасное взаимодействие для клиент-серверных приложений, предотвращая фальсификации, подслушивание и подделку сообщений. Вторым протоколом является SRTP (Secure Real-time Transport Protocol) – определяет профиль протокола RTP и предназначен для шифрования, установления подлинности сообщения, целостности, защиты от подмены данных RTP в однонаправленных и multicast передачах медиа и приложениях [11]. Сначала WebRTC выполняет рукопожатие DTLS с помощью соединения ICE. Для передачи аудио/видео используется другой

протокол под названием RTP (Real-time Transport Protocol) – протокол передачи данных в реальном времени [12]. Пакеты, передаваемые по RTP, защищаются с помощью SRTP. Сессия SRTP начинается с извлечения ключей из установленной сессии DTLS.

После установки безопасного двунаправленного соединения между двумя агентами WebRTC начинается этап коммуникации. Для этого используются протоколы RTP и SCTP (Stream Control Transmission Protocol) – протокол передачи с управлением потоком [13]. RTP используется для передачи медиа, зашифрованного с помощью SRTP, а SCTP нужен для отправки и приема сообщений по каналу связи.

Таким образом WebRTC объединяет множество специализированных протоколов в единую систему, каждая часть которой выполняется непосредственно друг за другом. Система протоколов, которую объединяет WebRTC, представлены на рисунке 1.4.

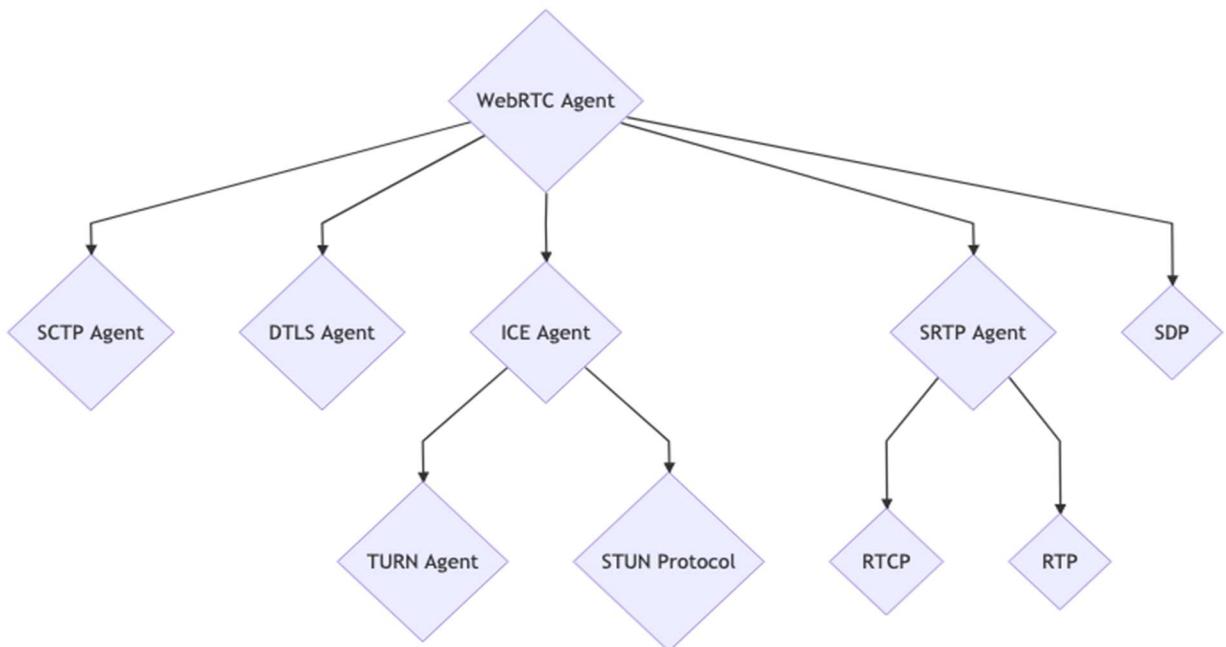


Рисунок 1.4 – Протоколы, используемые WebRTC

## 1.2 Анализ существующих аналогов

Проведенный анализ литературных источников продемонстрировал понятие, что такое видеоконференция и видеоконференцсвязь, а так же на основе каких технологий и протоколов работают платформы для проведения данных видеоконференций.

На данный момент существует довольно обширный выбор платформ для

проведения видеовстреч. Для клиента важными показателями являются простота использования, необходимый и достаточный функционал, а также безопасность при проведении онлайн-встреч. В соответствии с анализом литературных источников и перечисленными показателями проведён анализ программных средств, позволяющих выполнять одну или несколько из перечисленных требований: простота использования, обширный функционал, безопасность.

В результате были найдены и проанализированы различные программные средства, краткая характеристика которых представлены ниже.

### 1.1.1 Программное средство Zoom

Zoom – это облачный сервис видеоконференций, разработанный и поддерживаемый компанией Zoom Video Communications. Он предназначен для работы на расстоянии и предлагает широкий спектр услуг, включая видео- и аудиоконференции, совместную работу, чат и вебинары. Интерфейс приложения представлен на рисунке 1.2.

Приложение было создано в 2013 году для корпоративных пользователей. Настоящий бум платформы Zoom случился в условиях пандемии. Тогда Zoom активно использовали не только сотрудники для работы, но и обычные люди для личного общения с близкими [14].

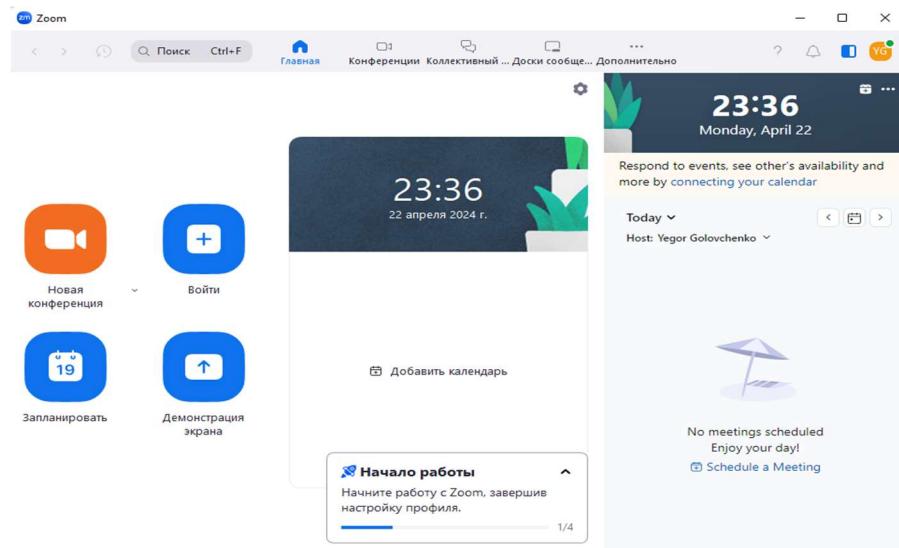


Рисунок 1.2 – Интерфейс Zoom

Среди основных функций данного программного средства можно выделить следующие:

- трансляция видео и аудио в высоком качестве;
- поддержка видеоконференций с числом участников до 10000;
- предоставление инструментов для совместной работы, таких как обмен сообщениями, обмен файлами и совместное использование экрана;
- возможность демонстрации своего экрана;

- возможность изменить цветовую тему встречи, загрузить уникальные обои и закрепить наиболее используемые действия на панели инструментов;
- возможность записывать свои встречи или мероприятия;
- возможность интеграции с другими приложениями.

Данный набор функций делают его удобным инструментом для проведения видеоконференций. Можно выделить следующие недостатки:

- ограничение по времени звонка 40 минут в бесплатной версии;
- сложность в использовании;
- слабая безопасность приложения;
- блокировка оплаты лицензий.

### 1.1.2 Программное средство Microsoft Teams

Microsoft Teams – это платформа для сотрудничества, общения и повышения продуктивности, которая включает функции, такие как чат, видео, обмен файлами и звонки. Он является частью экосистемы Microsoft 365 и тесно интегрирован с другими продуктами Microsoft, а также распространяется по корпоративной подписке [15]. Интерфейс приложения представлен на рисунке 1.3.

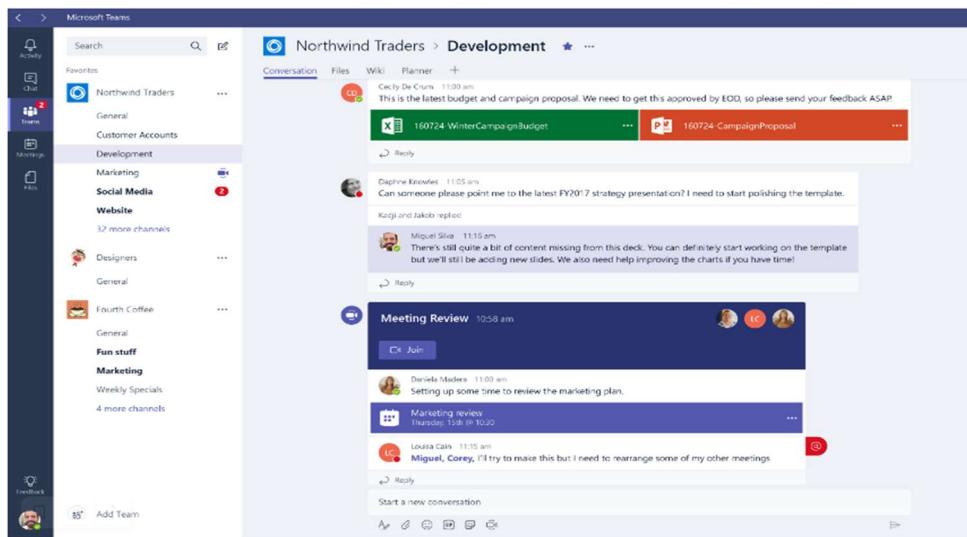


Рисунок 1.3 – Интерфейс Microsoft Teams

Бесплатная версия Microsoft Teams дает возможность запускать видеоконференции до 100 собеседников. У пользователей есть возможность создавать группы, обмениваться файлами, показывать экраны и совместную работу над документами в режиме реального времени. Среди основных функций данного программного средства необходимо выделить следующие:

- поддержка видеоконференций с числом участников до 100;
- предоставление инструментов для совместной работы, таких как обмен сообщениями, обмен файлами и совместное использование экрана;

– интеграция с другими приложениями Microsoft, такими как Office, OneNote и Outlook;

– обеспечение шифрования данных для встреч, чатов, звонков и файлов.

Из недостатков можно выделить следующие:

– нестабильная работа, долгая и некорректная загрузка файлов, которая может вывести из строя всю систему устройства;

– нет поддержки файлов DOC, XLS и PPT;

– не самый простой функционал.

### 1.1.3 Программное средство Google Meet

Google Meet – это безопасная и стабильная платформа помогает общаться с коллегами и успешно решать рабочие задачи каждый день. Google Meet раньше знали под именем Google Hangouts. Речь идет о Google Hangout Meet – версии, ориентированной на нужды бизнеса. Программа способна поддерживать до 100 участников для обычных пользователей и 250 для пользователей G Suite. Meet прекрасно справляется с рабочими вызовами, семинарами и даже небольшими беседами с друзьями. Интерфейс приложения представлен на рисунке 1.4.

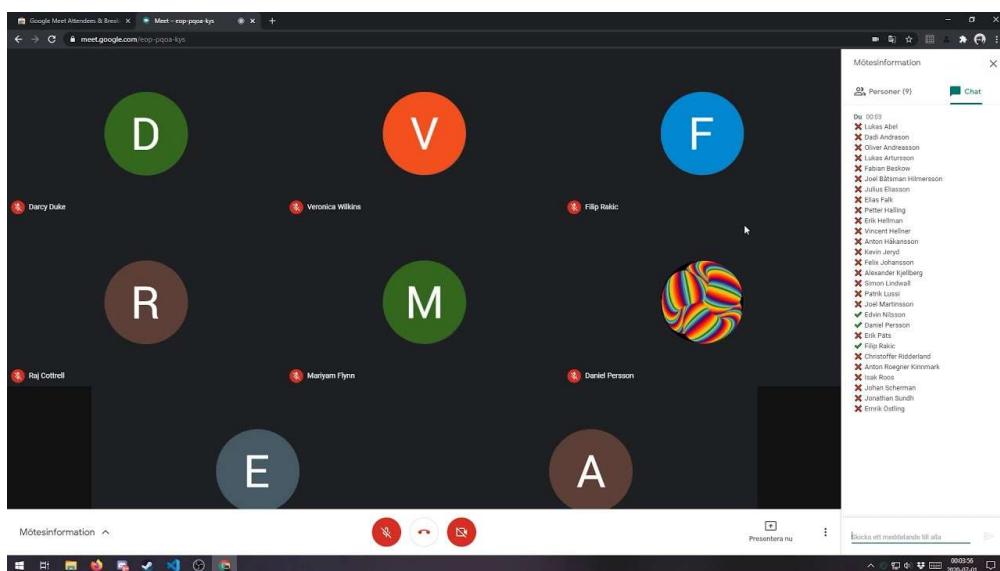


Рисунок 1.4 – Интерфейс Google Meet

Видео встречи продолжительностью до 60 минут предоставляются бесплатно. При желании увеличить длительность сессии, необходимо приобрести платный план. Все дополнительные возможности, включая прямую трансляцию, международные номера для набора, запись встреч и административное управление, предлагаются по определенной цене. Одним из основных плюсов Google Meet является использование AI для автоматической коррекции освещения. Эта функция позволяет участникам встречи присоединяться к видеозвонку без беспокойства о видимости [16].

Ниже приведен перечень основных функций программного средства, представленных на сегодняшний день:

- поддержка видеоконференций с числом участников до 250;
- предоставление инструментов для совместной работы, таких как обмен сообщениями, обмен файлами и совместное использование экрана;
- видео подписи и субтитры во время общения и выступлений;
- функция размытия фона;
- простота в использовании;
- интеграция с Google Calendar и Google Jamboard, поддержка Gmail;
- высокая безопасность.

У программного средства Google Meet есть и недостатки:

- google Meet позволяет использовать всего один экран за раз;
- если ведущий демонстрирует презентацию Keynote или PowerPoint, иногда экран остается пустым для всех участников;
- ограниченность функционала.

#### **1.1.4 Сервис Jitsi Meet**

Jitsi – это полностью бесплатная платформа для проведения видеоконференций с открытым кодом [17]. Сервис безопасен, использует сквозное шифрование. На сайте сервиса указано, что одновременно во встрече могут участвовать до 100 человек без ограничений по времени. Для того чтобы организовать встречу и принять в ней участие, дополнительная регистрация не нужна. Для начала видеозвонка достаточно кликнуть на кнопку в браузере. Сервис сам генерирует название встречи, чтобы ссылка была максимально безопасной. Если же вы вводите название встречи самостоятельно, то теоретически к встрече может подключиться любой человек, просто удачно подобрав ссылку. Участники по умолчанию свободно присоединяются к сеансу видеосвязи. Однако в настройках безопасности можно включить режим лобби (вход только после одобрения) или установить пароль. На компьютере Jitsi запускается в веб-браузерах Chrome, Opera, Яндекс. Браузер и Firefox. Safari пока не поддерживается. Интерфейс приложения представлен на рисунке 1.5.

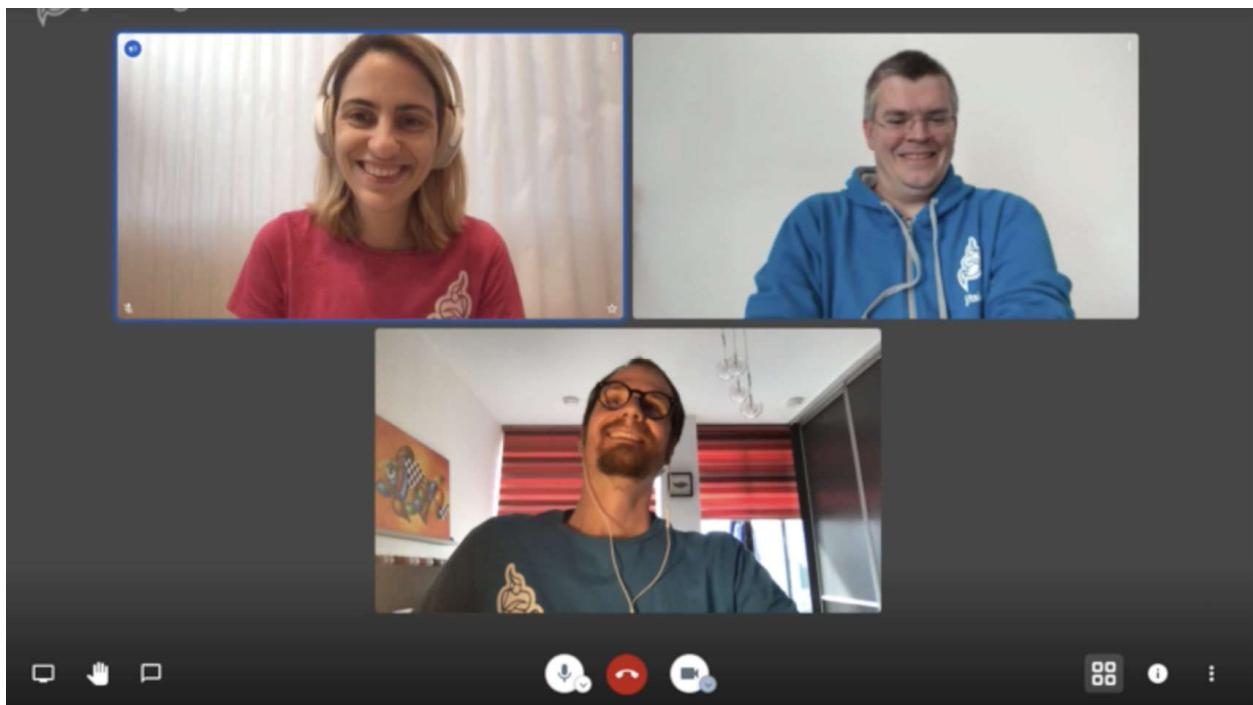


Рисунок 1.5 – Интерфейс Jitsi Meet

Ниже приведен перечень основных функций программного средства Jitsi Meet:

- возможность демонстрации экрана;
  - чат со смайлами и возможность отправить личное сообщение участнику;
  - настройка трансляции в YouTube;
  - возможность записи с сохранением в Dropbox;
  - простота использования
  - интеграция с Google календарем для планирования встречи.
- Недостатки программного средства Jitsi Meet:
- отсутствие роли модератора;
  - ограниченность функционала.

### 1.1.5 Программное средство «Яндекс Телемост»

«Яндекс Телемост» – это сервис для видеоконференций, разработанный компанией «Яндекс» [18]. Он предлагает множество функций, которые делают виртуальные встречи легкими и доступными. В Телемосте можно устраивать видеовечеринки, собирать конференции с друзьями и встречаться с семьей. Достаточно просто создать встречу и отправить ссылку участникам. Встречи в Телемосте не ограничены по длительности и могут объединять до 40 участников. Интерфейс приложения представлен на рисунке 1.6.

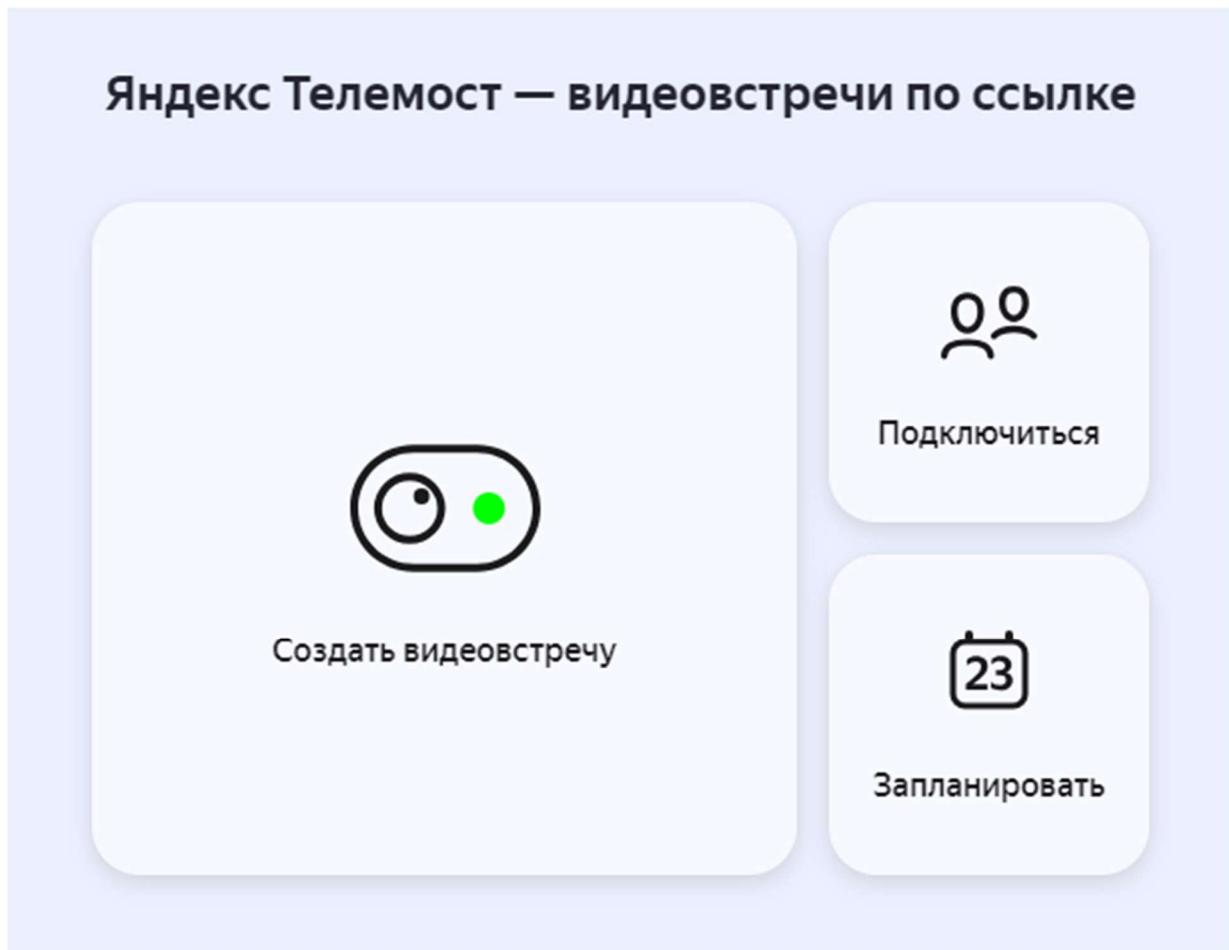


Рисунок 1.6 – Интерфейс «Яндекс телемост»

Ниже приведен перечень основных функций программного средства, представленных на сегодняшний день:

- предоставление инструментов для совместной работы, таких как обмен сообщениями, обмен файлами и совместное использование экрана;
- видео подписи и субтитры во время общения и выступлений;
- чат во время видеоконференции;
- простота в использовании;
- регистрация только для создателей видеовстречи;
- неограниченность по времени встречи;
- интеграция с Яндекс ID, Яндекс Календарём;
- высокая степень безопасности.

У программного средства «Яндекс» есть следующие недостатки:

- поддержка видеоконференций с числом участников до 40;
- ограниченность функционала.

### 1.1.6 Программное средство «МТС Линк»

«МТС Линк» – российская экосистема сервисов для бизнес-коммуникаций [19]. Инструменты «МТС Линк» помогают работать обычным и

удаленным командам: проводить конференции, вебинары, курсы, настраивать совместную работу с помощью майнд-карт и мессенджера. Какие сервисы включены в экосистему «МТС Линк»: Встречи, Вебинары, Курсы, Чаты, Доски. Проводить вебинары, взаимодействовать с коллегами можно в браузере, в приложении для десктопа и смартфона. Компания разработала «МТС Линк» on premise, это решение позволяет установить платформу на собственные серверы компании. Таким образом будут доступны 3 цифровых продукта: «Линк Встречи», «Линк Вебинары», «Линк Курсы». К «МТС Линк» можно подключить другие сервисы: «МойОфис», «Битрикс24», «Яндекс Метрику». Пользователи могут выбрать нужные интеграции в специальном магазине – это модуль в личном кабинете «МТС Линк», доступный для корпоративных клиентов и только на некоторых платных тарифах. Интерфейс приложения представлен на рисунке 1.7.

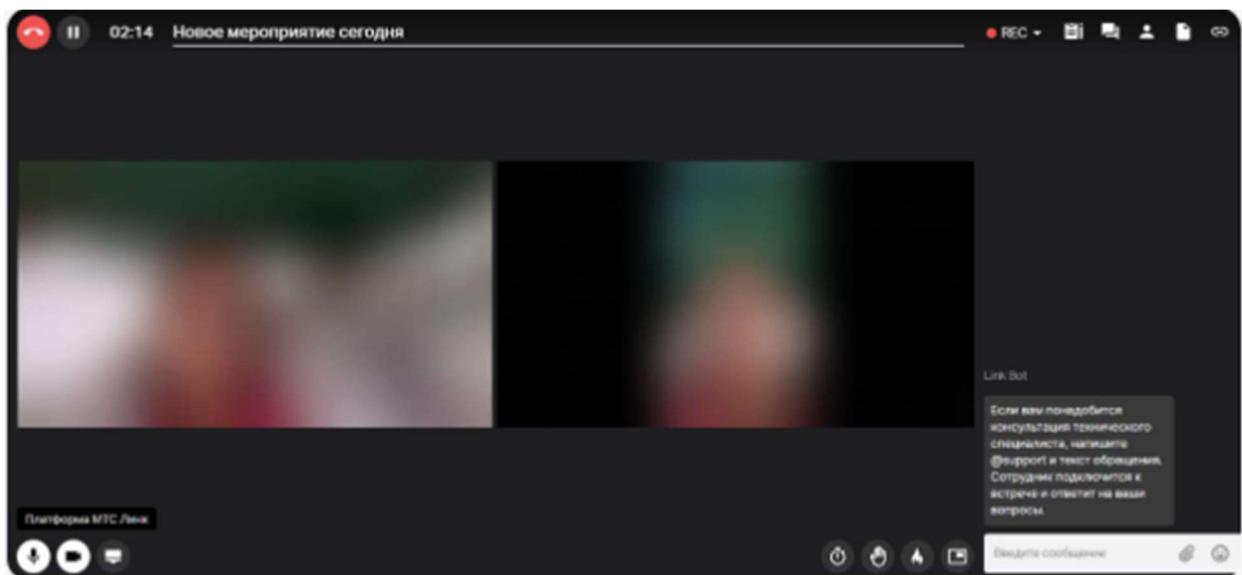


Рисунок 1.7 – Интерфейс «МТС Линк»

У платформы «МТС Линк» можно выделить следующие преимущества:

- возможность проводить собрания, обучения, работать всей командой на досках;
- наличие on-premise решение, которое позволяет установить платформу на сервер компании;
- возможность использовать эффективные дополнительные функции: доски, опросы, заметки, реакции во время звонка;
- возможность настраивать интеграции платформы со сторонними приложениями по API, выбирая их в специальном разделе;

Недостатки программного средства «МТС Линк»:

- некоторые кнопки и иконки расположены в непривычных местах. У корпоративного мессенджера часть текста в интерфейсе просто не видно;
- высокая нагрузка на устройства.

## **1.3 Формирование требований к проектируемому программному средству**

### **1.3.1 Назначение проекта**

Назначением проекта является разработка платформы для проведения онлайн-занятий и встреч. По результатам изучения предметной области, анализа литературных источников и обзора существующих систем-аналогов сделан вывод, что для решения текущей задачи необходимо выполнить следующие шаги:

- проектирование архитектуры веб-приложения;
- проектирование базы данных веб-приложения;
- разработка алгоритма для организации видеозвонка между двумя и более пользователями;
- разработка алгоритма для передачи своего экрана другим пользователям;
- разработка алгоритма для включения и выключения микрофона и видеосвязи;
- разработка алгоритма передачи файлов и общей доски для работы во время занятий;
- разработка пользовательского интерфейса платформы;
- разработка пользовательского интерфейса видеозвонка;
- тестирование платформы.

### **1.3.2 Перечень основных функций**

Веб-приложение должно поддерживать следующие основные функции:

- регистрация и аутентификация пользователей;
- управление профилем зарегистрированных пользователей;
- возможность создания видеовстречи для зарегистрированных пользователей;
- добавление пользователей на видеовстречу с помощью ссылки;
- создание расписания видеовстреч для зарегистрированных пользователей;
- просмотр списка прошедших видеозвонков для зарегистрированных пользователей;
- возможность включения и выключения микрофона и видео;
- возможность выхода из звонка;
- возможность показа своего экрана другим участникам звонка;
- возможность голосового перевода;
- использование совместной белой доски участникам звонка;
- передача файлов между участниками звонка;
- чат между участниками звонка.

### **1.3.3 Требования к входным данным**

Входные данные для программного средства должны быть представлены в виде запросов пользователей через интерфейс платформы, введенные ими при создании видеозвонка, подключении к нему, просмотра запланированных и прошлых видеозвонков, различных действий непосредственно во время звонка.

### **1.3.4 Требования к выходным данным**

Выходные данные отображаются пользователю в виде страниц платформы с запрашиваемой для пользователя информацией, такой как список запланированных и прошедших видеозвонков, а также страниц с комнатой для видеозвонка.

### **1.3.5 Требования к временным характеристикам**

Производительность программного средства должна удовлетворять следующим временным характеристикам:

- время обработки входных данных не должно превышать 1 секунды;
- время отображения результата, полученного после обработки серверной частью не должно превышать 1 секунды.

### **1.3.6 Требования к надежности**

Программное средство должно защищать персональную информацию о пользователях, обеспечивать безопасное общение между клиентской и серверной сторонами и защиту от несанкционированного доступа к данным. Также оно должно обеспечить стабильную работу приложения при высоких нагрузках и отказоустойчивость системы.

### **1.3.7 Выбор технологий программирования**

В качестве основных языков программирования были выбраны JavaScript и PHP. JavaScript (JS) – мультипарадигменный язык программирования [20]. Он поддерживает объектно-ориентированный, императивный и функциональный стили. JavaScript является реализацией спецификации ECMAScript (стандарт ECMA-262). Обычно он используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты включают:

- динамическую слабую типизацию;
- автоматическое управление памятью;
- прототипы для создания объектов;
- передачу функции как аргументы другим функциям и возвращение из функций.

Язык PHP – это популярный язык программирования общего назначения с открытым исходным кодом и подробной документацией, который специально создан для разработки веб-приложений [21]. PHP поддерживает

объектно-ориентированное программирование, работу с базами данных и файлами, шаблонизацию, благодаря чему подходит для создания MVC-приложений. MVC – это архитектурный шаблон, который разделяет приложение на три компонента: модель, представление и контроллер. Модель отвечает за работу с данными, представление отвечает за отображение данных на экране, а контроллер отвечает за обработку запросов пользователя и связывание модели и представления. MVC позволяет упростить разработку, тестирование и поддержку веб-приложений, а также повысить их гибкость и расширяемость.

Для разработки клиентской части используется библиотека для создания пользовательских интерфейсов ReactJs [22]. React был разработан Facebook и Instagram для обеспечения более эффективной разработки пользовательских интерфейсов. Он не является фреймворком, а скорее библиотекой, специализированной на визуализации. React используется не только для веб-разработки, но и в других областях, таких как мобильные приложения (с помощью React Native) и виртуальная реальность (с помощью React 360). Основные принципы и цели React:

- стремление минимизировать ошибки, возникающие при разработке пользовательских интерфейсов.;
- использование компонентов – автономных логических фрагментов кода, описывающих часть интерфейса;
- объединение компонентов для создания полноценных пользовательских интерфейсов;
- абстрагирование от множества деталей визуализации, позволяя разработчикам сосредоточиться на дизайне.

React может быть использован для небольших частей интерфейса, но его преимущества проявляются при создании всего приложения с его помощью. Он не навязывает строгие правила организации кода, что позволяет командам структурировать проект по своему усмотрению. Он использует синтаксис HTML-in-JavaScript под названием JSX (JavaScript и XML).

Для разработки серверной части используются фреймворки Laravel и Express. Laravel – это фреймворк для быстрой разработки веб-приложений на PHP [23]. К нему написана интерактивная документация. Для каждой функции в Laravel есть отдельная статья с примерами. Это удобно как для начинающих, так и для опытных разработчиков. Laravel предоставляет удобный способ для работы с базами данных с помощью миграций. Миграции позволяют изменять структуру базы данных без необходимости писать SQL-запросы. Также для работы с базой используется Eloquent ORM, которая позволяет работать с базами данных в объектно-ориентированном стиле. Вместе с Laravel идёт Artisan – это консоль для разработчиков, которая упрощает взаимодействие с Laravel. С его помощью можно проводить миграции баз данных, настраивать авторизацию и взаимодействовать с компонентами фреймворка. У Laravel есть множество шаблонов для создания пользовательского интерфейса, аутентификации пользователей, что значительно упрощает процесс разработки. Laravel

также упрощает тестирование кода.

Express.js – это веб-фреймворк для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT [24]. Express представляет собой популярный веб-фреймворк, написанный на языке JavaScript. Он работает внутри среды исполнения Node.js и предназначен для создания веб-приложений и API. Express чаще всего используется в разработке бизнес-логики веб-сайтов. Его также можно применять в разработке мобильных и десктопных приложений. Преимуществами разработки на Express являются простота, гибкость, маршрутизация, поддержка промежуточного программного обеспечения, шаблоны для создания пользовательского интерфейса.

В качестве системы управления реляционными базами данных была выбрана MySQL [25]. СУБД MySQL – это одна из самых популярных и широко используемых систем управления базами данных в мире. Она имеет множество преимуществ, таких как:

1 Открытый исходный код. MySQL является свободной и открытой системой управления базами данных, которую можно использовать без оплаты лицензии, а также изучать и модифицировать её код.

2 Простота использования. MySQL имеет простой и интуитивно понятный интерфейс, что делает её легкой в использовании даже для новичков. Она также поддерживает множество языков программирования и инструментов для работы с данными.

3 Высокая производительность. MySQL обладает высокой скоростью обработки запросов и транзакций, а также оптимизирована для работы с большими объёмами данных. Она также поддерживает кэширование, индексирование, репликацию и другие механизмы, повышающие эффективность работы с данными.

4 Масштабируемость. MySQL способна работать с различными типами и размерами данных, от небольших локальных баз до крупных распределённых систем. Она также позволяет горизонтально и вертикально масштабировать свою архитектуру в зависимости от потребностей приложения.

5 Надёжность и безопасность. MySQL обеспечивает целостность и консистентность данных, а также защищает их от несанкционированного доступа и модификации. Она поддерживает транзакции, резервное копирование, восстановление, шифрование, аутентификацию, авторизацию и другие средства обеспечения безопасности данных.

6 Гибкость и расширяемость. MySQL позволяет выбирать различные типы таблиц, движков хранения, форматов данных, кодировок, коллаций и других параметров, в зависимости от специфики приложения. Она также поддерживает создание собственных функций, процедур, триггеров, представлений и других объектов базы данных. Кроме того, MySQL совместима с множеством фреймворков, платформ и стандартов веб-разработки.

Для контроля процесса разработки был использован Git – система управления версиями с распределенной архитектурой [26]. В отличие от некогда

популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме.

В качестве среды разработки был использован Visual Studio Code. Visual Studio Code (VS Code) – это текстовый редактор, разработанный Microsoft для Windows, Linux и macOS [27]. Он позиционируется как лёгкий редактор кода для кроссплатформенной разработки веб- и облачных приложений. Вот некоторые ключевые аспекты:

- поддержка множества языков программирования;
- подсветка синтаксиса, IntelliSense (автодополнение кода), рефакторинг и отладка;
- инструменты для работы с Git, что упрощает совместную разработку;
- доступность многих функций через палитру команд или JSON-файлы, что делает его гибким и настраиваемым;
- поддержка расширений, которые добавляют дополнительные функциональные возможности;
- возможность установить расширения для поддержки разных языков, инструментов и тем оформления;
- легкий по сравнению с полноценной средой разработки, но при этом обладает мощными возможностями;
- позволяет быстро начать работу над проектом без лишних настроек.

В целом, Visual Studio Code – это удобный и гибкий редактор кода, который пользуется популярностью среди разработчиков.

### **1.3.8 Выводы**

В результате изучения предметной области, анализа литературных источников и обзора существующих аналогов, технологий и подходов для создания платформы для видеовстреч, были сформулированы требования, которые позволяют осуществить успешное проектирование и разработку платформы для проведения онлайн-занятий и видеовстреч.

## **2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА СПЕЦИФИКАЦИИ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

### **2.1 Описание функциональности программного средства**

Для представления функциональной модели была выбрана диаграмма вариантов использования UML , которая отражает отношения между актерами и прецедентами и позволяет описать систему на концептуальном уровне [28]. Прецедент соответствуетциальному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. UML предназначен для определения, визуализации, проектирования и документирования программных систем.

Диаграмма вариантов использования представлена на рисунке 2.1. На диаграмме можно выделить два основных составляющих элемента – актер и прецедент. Актер – стилизованный человек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью. Прецедент – эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым актером результатам.

На основании представленной диаграммы вариантов использования можно сделать вывод, что в системе будет существовать два основных актера:

- зарегистрированный пользователь;
- незарегистрированный пользователь.

Ниже более подробно рассматривается перечень прецедентов для каждого актера.

Незарегистрированному пользователю предоставляются следующие возможности:

- регистрация в системе;
- подключение к видеозвонку по ссылке на звонок либо по номеру комнаты;
- отключение своего микрофона;
- отключение своего видео;
- использование совместной белой доски;
- завершение звонка;
- копирование ссылки на комнату для передачи другим пользователям;
- демонстрация своего экрана для других участников звонка.

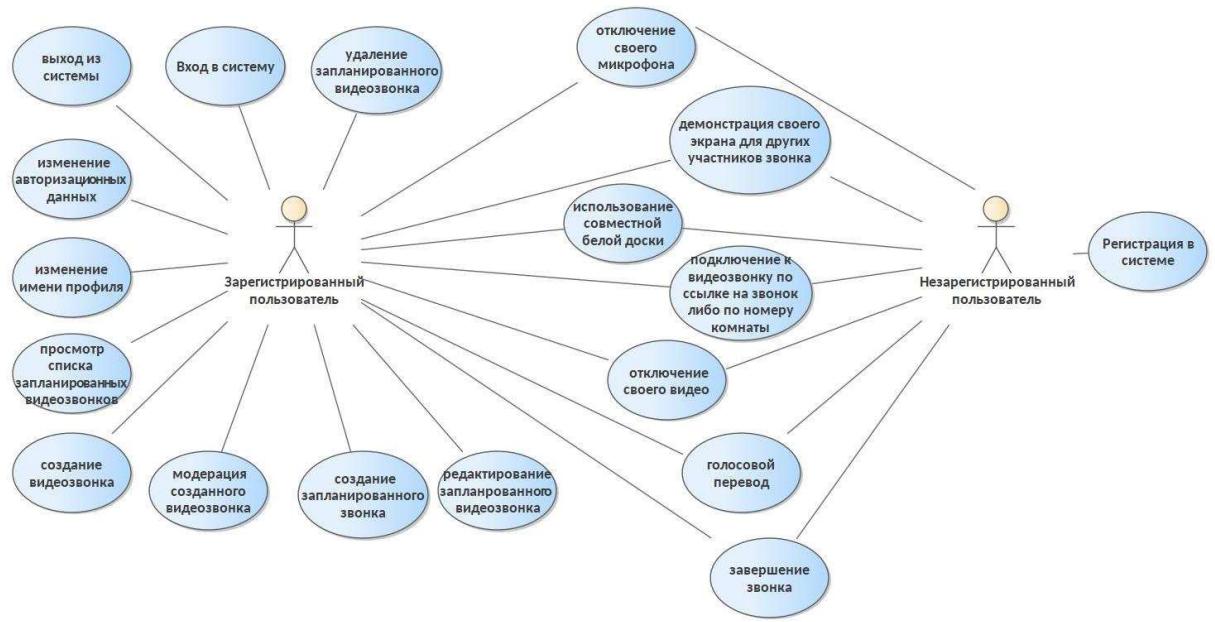


Рисунок 2.1 – Диаграмма вариантов использования

Зарегистрированный пользователь имеет доступ к следующим возможностям:

- выход из системы;
- вход в систему;
- изменение имени профиля;
- изменение авторизационных данных;
- создание видеозвонка;
- модерация созданного видеозвонка;
- создание запланированного звонка;
- просмотр списка запланированных видеозвонков;
- редактирование запланированного звонка;
- удаление запланированного звонка;
- подключение к видеозвонку по ссылке на звонок либо по номеру комнаты;
- отключение своего микрофона;
- отключение своего видео;
- использование совместной белой доски;
- показ своего экрана для других участников звонка.

## 2.2 Спецификация функциональных требований

Спецификация требований к программному обеспечению – это описание поведения системы, которую необходимо разработать. Спецификация функциональных требований включает пользовательские сценарии, которые носят название варианты использования. Они описывают все варианты взаимодействия между пользователями и информационной системой. Кроме

пользовательских сценариев спецификация также содержит требования, которые могут определять ограничения на реализацию, например, такие как стандарты качества, требования производительности или какие-либо проектные ограничения.

Для описания требуемых характеристик программного средства была составлена функциональная спецификация. Данная спецификация включает в себя следующие функциональные требования к разрабатываемому ПС:

- а) регистрация пользователя должна быть доступна любому неавторизованному пользователю веб-приложения;
- б) регистрация пользователя в веб-приложении должна производиться путем заполнения регистрационной формы;
- в) регистрационная форма должна содержать поля ввода адреса электронной почты и имени, которые являются уникальными, пароля;
- г) регистрационная форма должна содержать поле ввода подтверждения пароля;
- д) поля введённого пароля и подтверждённого пароля должны совпадать;
- е) пароль должен быть не менее 8 символов;
- ж) авторизация пользователя должна быть доступна зарегистрированному пользователю веб-приложения;
- з) авторизационная форма должна содержать поле ввода адреса электронной почты, пароля;
- и) авторизация пользователя в веб-приложении должна производиться после ввода адреса электронной почты и пароля со стороны пользователя и проверки авторизационных данных с данными, указанными при регистрации пользователя в веб-приложении;
- к) редактирование профиля пользователя должно быть доступно зарегистрированному пользователю веб-приложения;
- л) редактирование профиля пользователя должно включать в себя возможность изменения авторизационных данных (имени, адреса электронной почты и пароля);
- м) возможность удаления своего профиля;
- н) возможность создания видеозвонков должна быть доступна только зарегистрированному пользователю;
- о) возможность запланировать видеозвонок должна быть доступна только зарегистрированному пользователю;
- п) возможность просмотра списка запланированных видеозвонков должна быть доступна только зарегистрированному пользователю;
- р) возможность модерация созданного видеозвонка должна быть доступна только пользователю, который создал звонок;
- с) модератор видеозвонка имеет возможность отключать и включать микрофоны и видео других участников звонка;
- т) использование совместной белой доски во время видеозвонка должно быть доступно всем участникам звонка;

у) отключение микрофона либо видео во время видеозвонка должно быть доступно всем участникам звонка;

ф) подключение к видеозвонку по ссылке на звонок либо по номеру комнаты должно быть доступно всем пользователям;

х) показ своего экрана для других участников звонка должен быть доступен всем пользователям;

ц) функция голосового перевода во время видеозвонка должна быть доступна всем пользователям.

### **2.2.1 Требования к пользовательскому интерфейсу**

Пользовательский интерфейс – интерфейс, обеспечивающий передачу информации между пользователем и программно-аппаратными компонентами компьютерной системы.

Список требований к пользовательскому интерфейсу веб-приложения:

а) неавторизованный пользователь должен видеть на главном экране меню для возможности регистрации, входа и подключения к видеозвонку по ссылке либо номеру комнаты;

б) любой пользователь должен иметь возможность подключения к видеозвонку по ссылке либо номеру комнаты, введя непосредственно номер комнаты и своё имя, если пользователь не зарегистрирован;

в) авторизованный пользователь должен видеть на главном экране меню для просмотра запланированных видеозвонков, создания своего видеозвонка, подключения к видеозвонку по ссылке либо номеру комнаты;

г) авторизованный пользователь должен видеть сверху меню, где он может выйти из своего аккаунта либо изменить данные для входа в аккаунт;

д) любой пользователь должен иметь возможность видеть других пользователей во время видеозвонка;

е) любой пользователь во время видеозвонка должен видеть снизу меню с функционалом для включения либо отключения аудио и видео, показа своего экрана, голосового перевода, завершения звонка.

### **2.2.2 Роли пользователей**

В приложении должна быть построена политика управления доступом на основе ролей. Необходимо обеспечить две основные роли пользователей:

– зарегистрированный пользователь;

– незарегистрированный пользователь.

## **3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

### **3.1 Разработка программной архитектуры**

Прежде чем приступать к непосредственной реализации программного средства, необходимо определиться с архитектурой системы, а также компонентов, на основе которых будет построено конечное приложение.

В первую очередь, необходимо провести анализ необходимой аппаратной конфигурации, на которой будут работать части конечного программного средства, и описать их взаимодействие между собой. Для описания узлов и их связей может использоваться диаграмма развертывания, которая представлена на рисунке 3.1.

На основе представленной выше диаграммы были сформулированы следующие требования:

1 Программное обеспечение «SQL Server» устанавливается на отдельный выделенный сервер либо на кластер серверов.

2 Узлы могут располагаться на отдельных серверах, однако целесообразно всю инфраструктурную часть приложения поместить в локальную сеть.

3 Для снижения сетевых нагрузок сервер базы данных вынесен в отдельный узел.

4 Клиент может осуществлять работу с системой по протоколам HTTP и HTTPS. При этом обязательным условием является использования протокола HTTPS при необходимости аутентификации пользователя в системе. Пользователь осуществляет вход в программное средство через браузер.

5 Загрузка и обработка исходных данных пользователя должны осуществляться на выделенной рабочей станции. Данная функциональность может быть реализована с использованием как одной рабочей станции, так и посредством нескольких станций. На данной рабочей станции свою работу будет осуществлять веб-сервер на Laravel. Рабочая станция должны быть оснащены на выбор операционной системой Windows 8.1/10/11.

6 Функциональность видеозвонков будет осуществляться на специально выделенной рабочей станции для снижения нагрузки на сервер, отвечающий за загрузку и обработку исходных данных пользователя. На данной рабочей станции свою работу будет осуществлять веб-сокет сервер на Express. Рабочая станция должны быть оснащены на выбор операционной системой Windows 8.1/10/11. Диаграмма развертывания показывает архитектуру программного средства и продемонстрирована на рисунке 3.1.

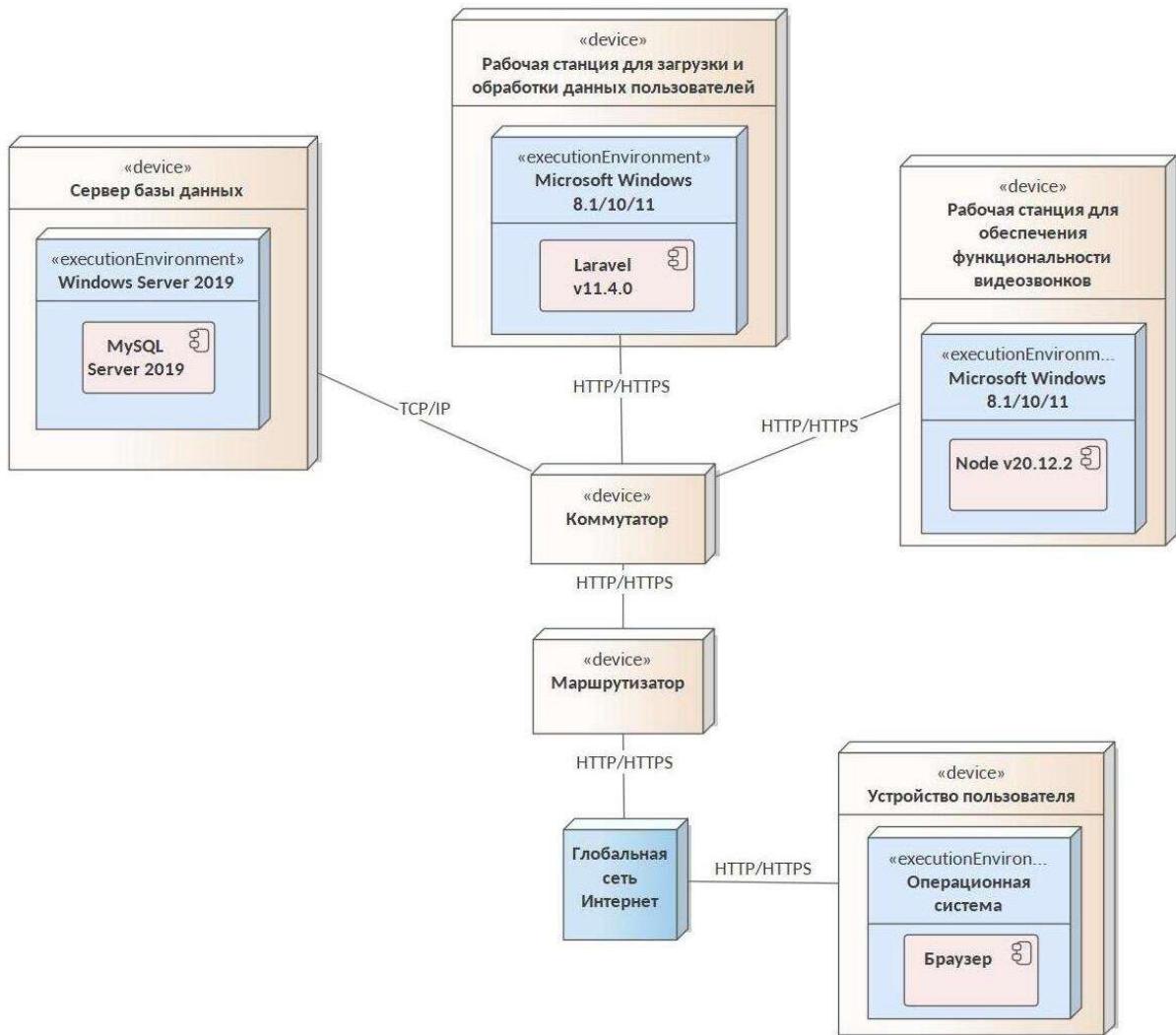


Рисунок 3.1 – Диаграмма развертывания

Для работы сервера базы данных требуются:

- процессор (один или несколько) уровня Intel Xeon i5-10400 с поддержкой 64-битной архитектуры и с тактовой частотой не менее 2 ГГц (либо более производительный);
- объем ОЗУ – не менее 8 Гбайт;
- два сетевых адаптера (допускаются встроенные), обеспечивающих доступ со скоростью 1 Гбит/сек;
- объем свободного пространства на жестком диске не менее 50 Гбайт для установки СУБД SQL Server и не менее 400 Гб для размещения файлов данных.

Представленное выше описание дает полное представление об технических характеристиках узлов, на которых будет эксплуатироваться данное программное средство.

## 3.2 Разработка модели базы данных

Процесс проектирования базы данных можно условно разделить на два этапа: логическое моделирование и физическое проектирование. Результатом первого из них является так называемая логическая модель данных, выражаемая обычно диаграммой «сущность-связь». Результатом второго этапа является готовая база данных.

Разрабатываемое программное средство использует базу данных для хранения данных о зарегистрированных пользователях, их сессиях, запланированных видеозвонках, прошедших видеозвонках, сообщениях в чате. Укрупненная модель базы данных представлена на рисунке 3.2.

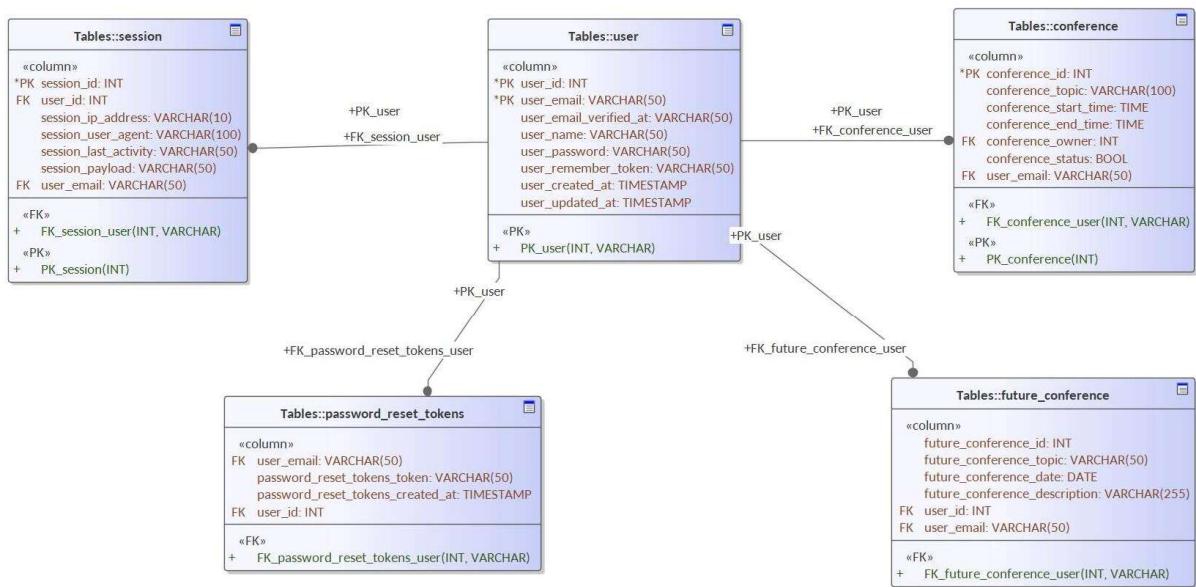


Рисунок 3.2 – Укрупненная информационная модель предметной области

Сущность *user* содержит следующие поля:

- *user\_id* – уникальный идентификатор пользователя (часть составного первичного ключа);
- *user\_email* – адрес электронной почты (часть составного первичного ключа);

– *user\_email\_verified\_at* – подтверждение электронной почты;

– *user\_name* – имя пользователя;

– *user\_password* – пароль пользователя;

– *user\_remember\_token* – токен пользователя;

– *user\_created\_at* – дата и время создания пользователя;

– *user\_updated\_at* – дата и время обновления данных о пользователе

Сущность *session* содержит следующие поля:

- *session\_id* – уникальный идентификатор сессии (первичный ключ);

- user\_id – внешний ключ на поле user\_id сущности user;
- session\_ip\_address – ip адрес сессии;
- session\_user\_agent – информация о пользователе во время сессии;
- session\_last\_activity – последняя активность пользователя в сессии;
- session\_payload – полезная нагрузка в сессии;
- user\_email – внешний ключ на поле user\_email сущности user.

Сущность password\_reset\_tokens содержит следующие поля:

- user\_email – внешний ключ на поле user\_email сущности user (первичный ключ);

- password\_reset\_tokens\_token – токен для сброса пароля;
- password\_reset\_tokens\_created\_at – дата и время создания токена;
- user\_id – внешний ключ на поле user\_id сущности user.

Сущность conference содержит следующие поля:

- conference\_id – уникальный идентификатор конференции (первичный ключ);

- conference\_topic – название конференции;
- conference\_start\_time – время начала конференции;
- conference\_end\_time – время окончания конференции;
- conference\_owner – внешний ключ на поле user\_id сущности user;
- conference\_status – статус конференции;
- user\_email – внешний ключ на поле user\_email сущности user.

Сущность future\_conference содержит следующие поля:

- future\_conference\_id – уникальный идентификатор запланированного звонка (первичный ключ);
- future\_conference\_topic – название запланированного звонка;
- future\_conference\_date – дата запланированного звонка;
- future\_conference\_description – описание запланированного звонка;
- user\_email – внешний ключ на поле user\_email сущности user.

### **3.3 Разработка алгоритмов программного средства**

Для последующей эффективной разработки программного средства необходимо разработать алгоритмы, позволяющие определить последовательность решения основных задач, решаемых в ходе разработки программного средства.

Далее приведены схемы вышеперечисленных алгоритмов, а также необходимые пояснения и уточнения.

### 3.3.1 Алгоритм создания видеозвонка

Схема алгоритма создания видеозвонка продемонстрирована на рисунке 3.3.

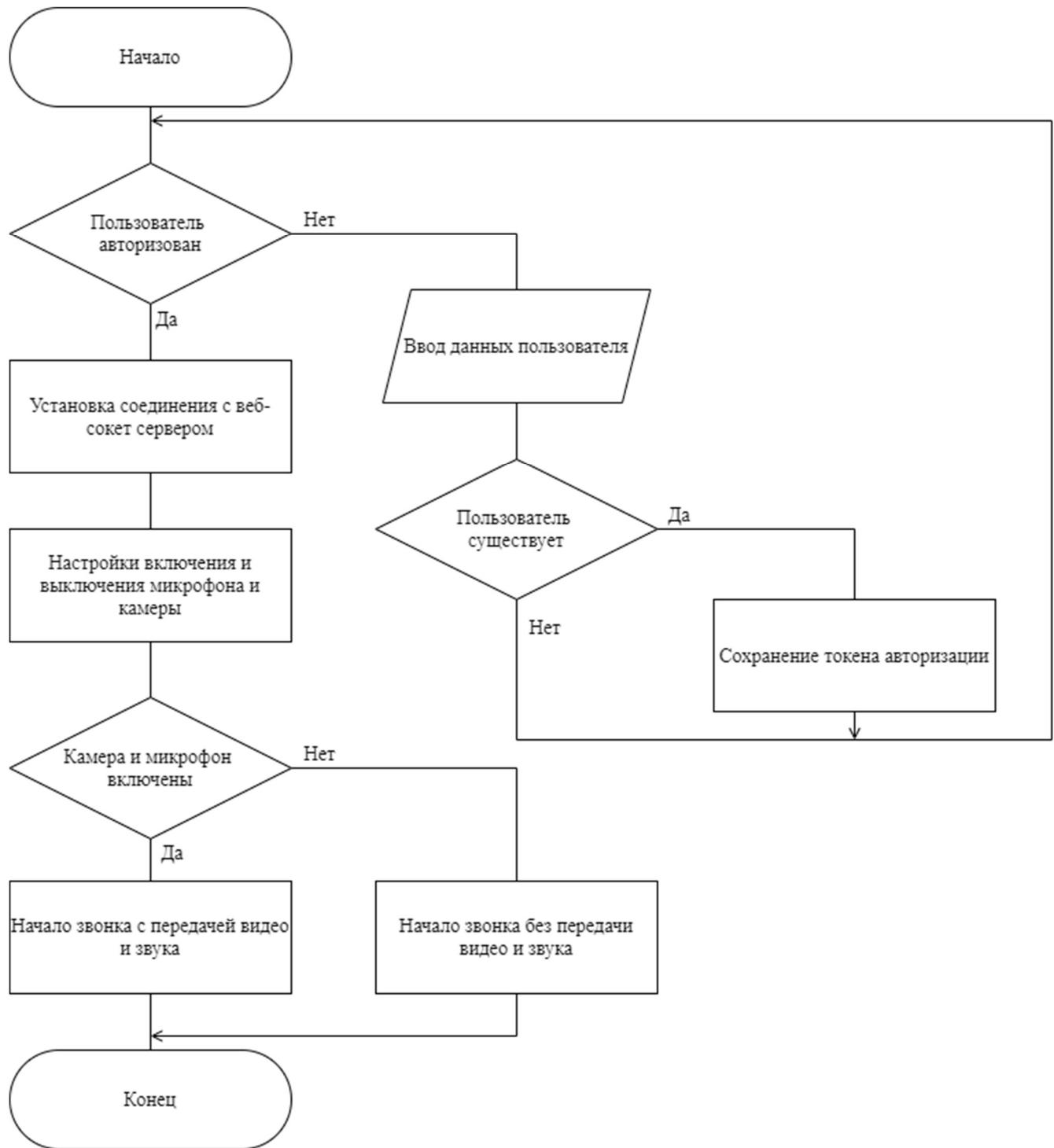


Рисунок 3.3 – Схема алгоритма создания видеозвонка

### 3.3.2 Алгоритм регистрации пользователя

Схема алгоритма регистрации пользователя продемонстрирован на рисунке 3.4.

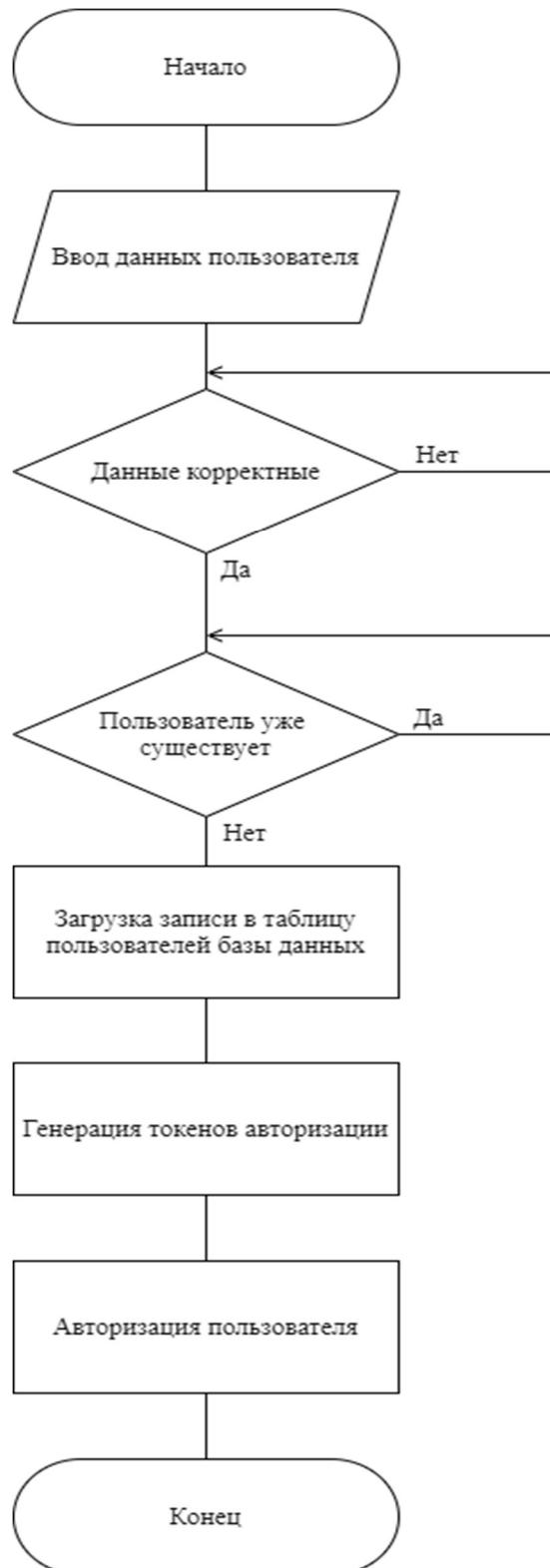


Рисунок 3.4 – Схема алгоритма регистрации пользователя

### 3.3.3 Алгоритм подключения нового пользователя к комнате

Схема алгоритма подключения нового пользователя к комнате продемонстрирован на рисунках 3.5, 3.6.

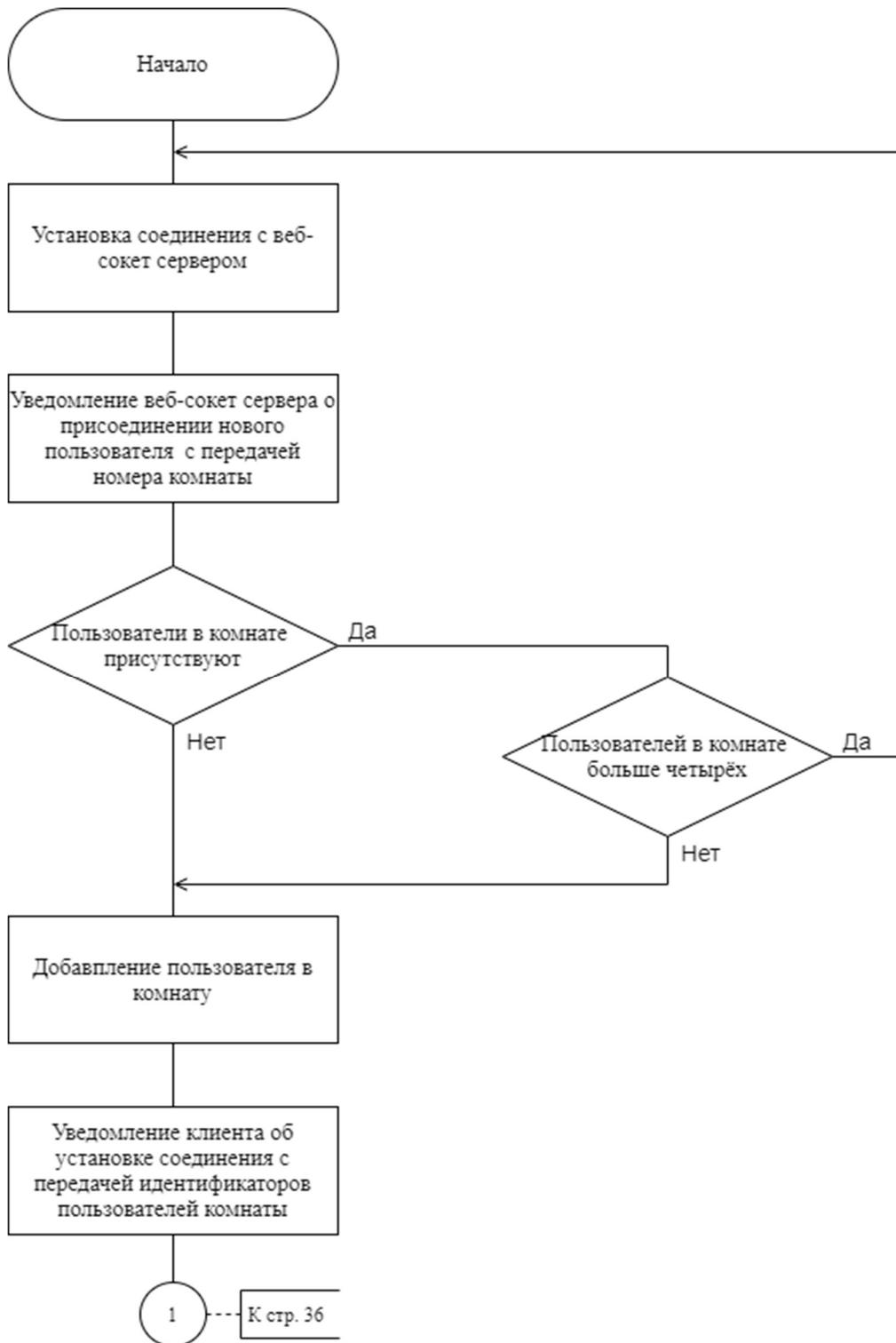


Рисунок 3.5, лист 1 – Схема алгоритма подключения нового пользователя

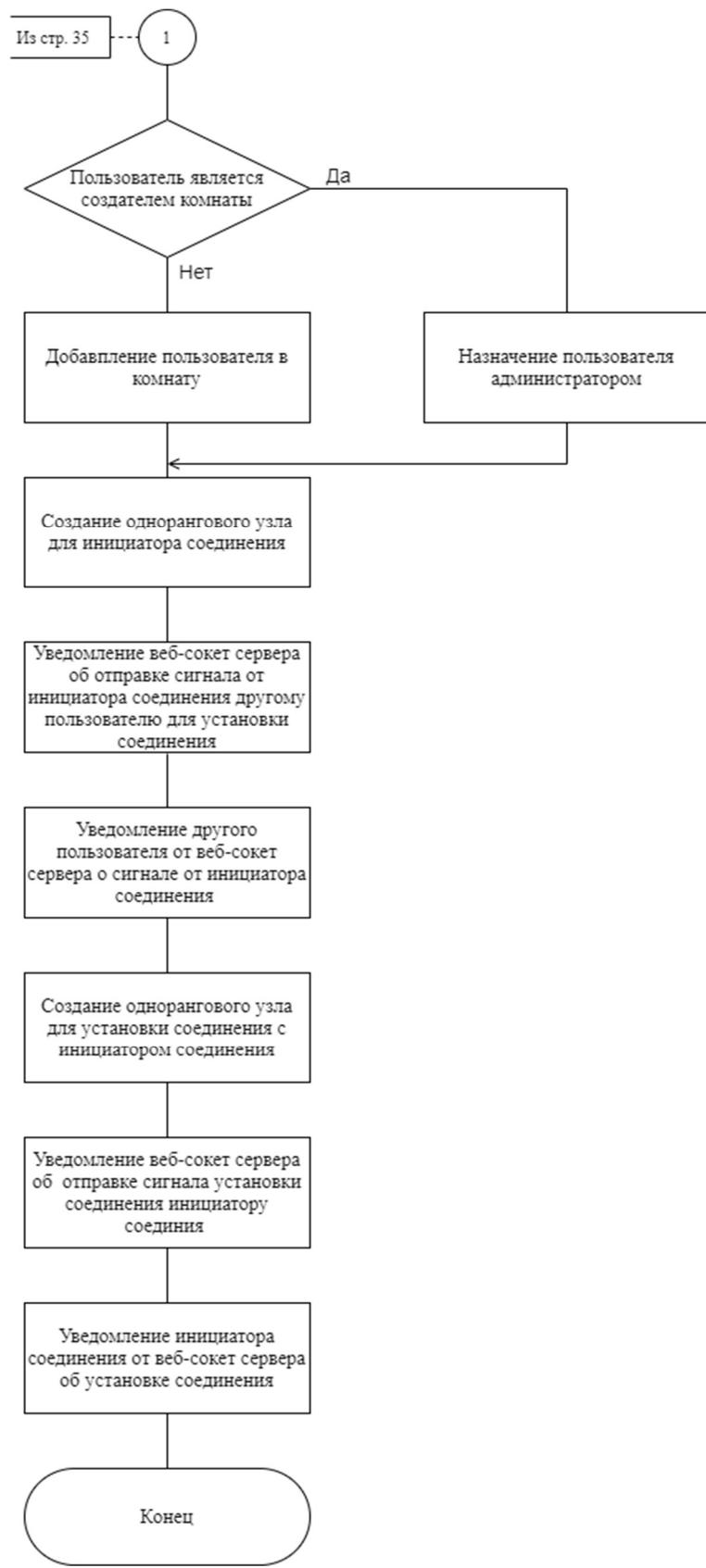


Рисунок 3.5, лист 2 – Схема алгоритма подключения нового пользователя

### 3.3.4 Алгоритм модерации комнаты

Схема алгоритма модерации комнаты продемонстрирован на рисунке 3.7.

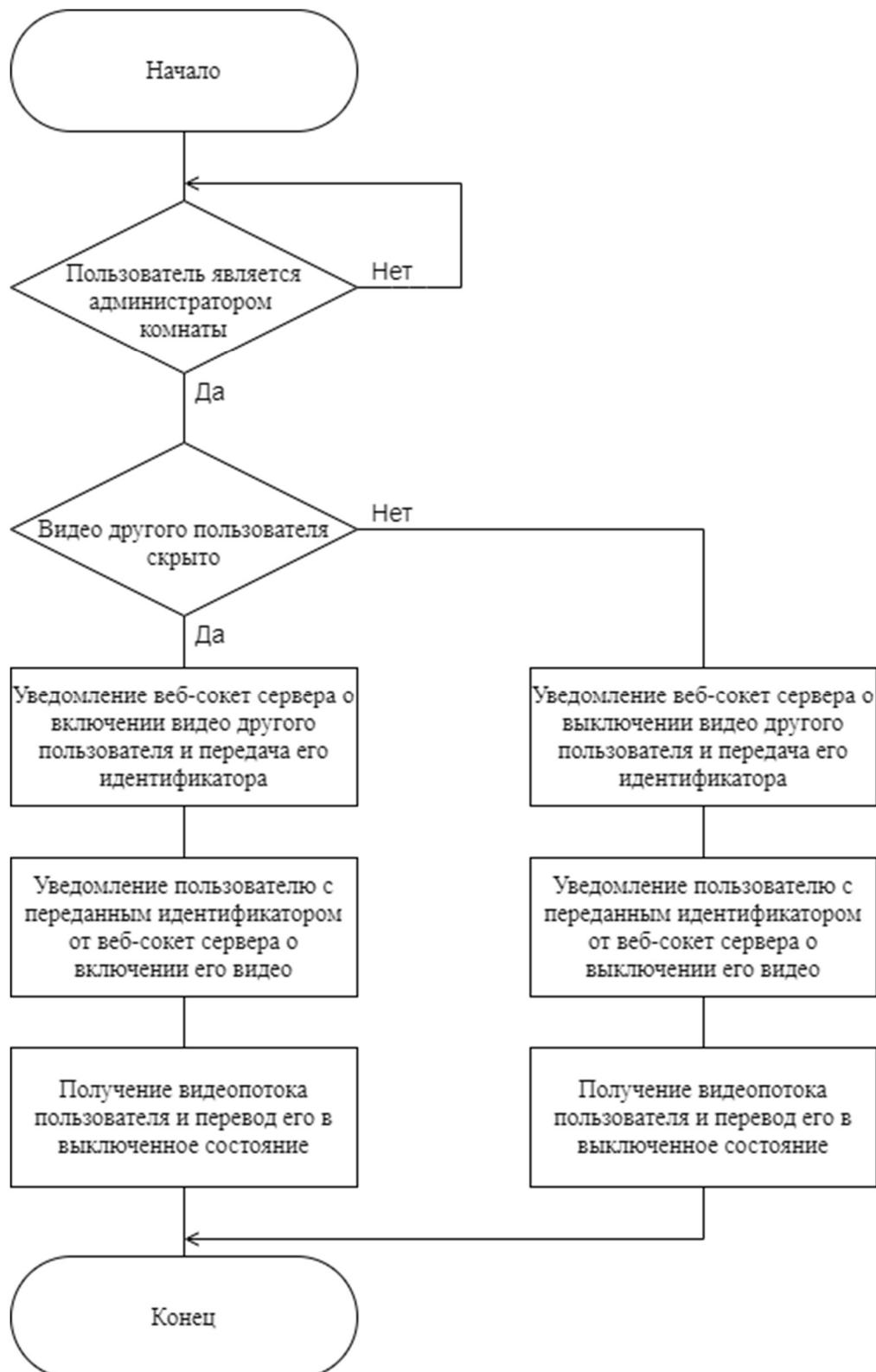


Рисунок 3.7 – Схема алгоритма модерации комнаты

### 3.3.5 Алгоритм демонстрации экрана

Схема алгоритма демонстрации экрана продемонстрирован на рисунке 3.8.

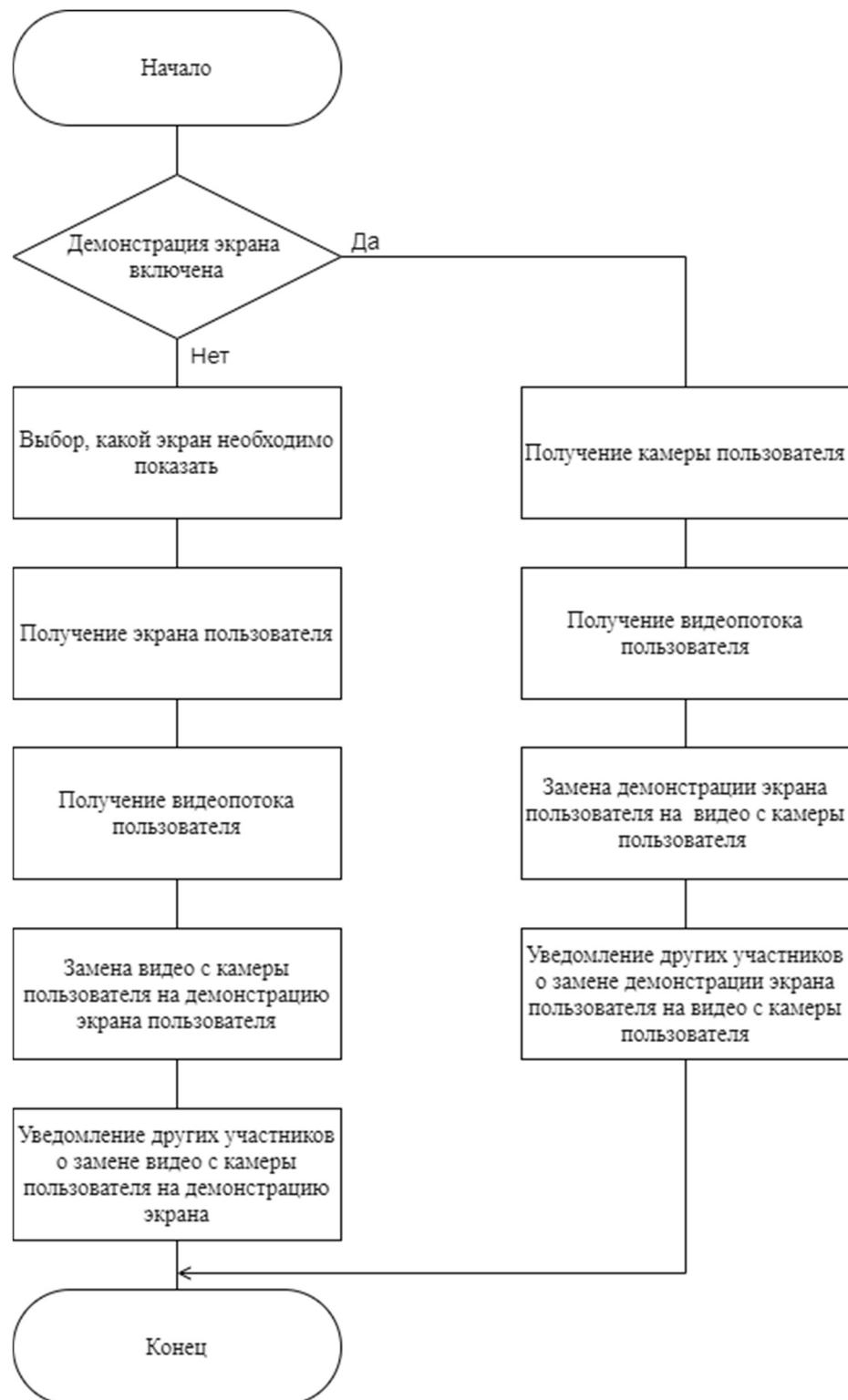


Рисунок 3.8 – Схема алгоритма демонстрации экрана

## **4 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

Разработка программного средства состоит из нескольких этапов и предполагает тщательный выбор технологий для прикладного использования в разработке, что, в свою очередь, требует наличия чёткого понимания архитектуры программного средства, его функциональных и нефункциональных требований, а также методов взаимодействия с базой данных. Соответствие данным параметрам позволяет разработать приложение, отвечающее поставленным запросам и реализующее заявленный функционал. Этапы разработки программного средства включают в себя анализ требований, проектирование, кодирование, тестирование и отладка, внедрение. Создание программного продукта целиком зависит от качественного выполнения каждой фазы.

Программное средство для проведения онлайн-занятий и встреч является микросервисным веб-приложением с клиент-серверной архитектурой, предоставляющее пользователям доступ посредством входа через браузер. Архитектурный стиль микросервисов – это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Существует абсолютный минимум централизованного управления этими сервисами. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных. В силу специфики поставленных задач было принято решение о том, что выбор микросервисной архитектуры в качестве модели приложения является оптимальным.

Данное программное средство включает в себя несколько сервисов, включая сервис для создания и проведения видеозвонков, сервис для проведения авторизации и регистрации, а также для созданием и ведением списка запланированных мероприятий. Для создания программного средства также разрабатывалась клиентская часть и база данных.

### **4.1 Взаимодействие с базой данных**

В качестве системы управления базами данных (СУБД) была выбрана система MySQL. Данная СУБД использует язык запросов SQL предоставляет весь необходимый для программного средства перечень возможностей для взаимодействия с базой данных: формирование запросов, осуществление поиска необходимых данных, синхронизация информации, а также выполнение аналитической обработки данных и получение разнообразных отчетов.

Фреймворк Laravel осуществляет поддержку MySQL делает достаточно простым взаимодействие с поддерживающими базами данных с помощью SQL

запросов, которые можно использовать благодаря фасаду DB, инструменту для построения запросов QueryBuilder, который предоставляет удобный и гибкий интерфейс для создания и выполнения запросов к базе данных и использует привязку параметров PDO для защиты приложения от атак с использованием SQL-инъекций, либо ORM Eloquent.

Для контроля версий базы данных используются миграции, которые позволяют структурировать и определять схему базы данных. Laravel обеспечивает независимую от системы управления базами данных поддержку миграций с помощью фасада Schema для создания таблиц и управления ими. Обычно при миграции данный фасад используется для редактирования таблиц и столбцов базы данных.

В программном средстве для проведения онлайн-занятий и встреч для взаимодействия с базой данных в Laravel используется ORM Eloquent. ORM или Object-Relational Mapping, объектно-реляционное отображение – технология программирования, суть которой заключается в создании «виртуальной объектной базы данных». Данная технология предоставляет возможность обращаться к базе данных с использованием упрощённого синтаксиса, без необходимости формировать запросы на чистом SQL, а также в значительной степени автоматизирует процессы создания запросов и процедур.

Eloquent является реализацией шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

Active record – шаблон проектирования приложений, описанный Мартином Фаулером в книге Patterns of Enterprise Application Architecture («Шаблоны архитектуры корпоративных приложений»). Active record является популярным способом доступа к данным реляционных баз данных в объектно-ориентированном программировании.

Схема Active Record описывает подход к доступу к данным в базе. Таблица базы данных или представление оборачивается в соответствующий класс, таким образом, экземпляру объекта соответствует единственная и уникальная строка таблицы. Операции сохранения и обновления информации об объекте класса взаимодействуют с этой строкой. Класс, обрабатывающий сущность, берёт на себя ответственность за предоставление методов доступа к информации о объектах и возможности изменения этой информации.

Eloquent поддерживает стандарты наименования классов и таблиц для унификация правил доступа и поддержки возможности автогенерации, что в значительно мере сокращает время, затрачиваемое разработчиком на конфигурацию связей между классами.

Также взаимодействие с базой данных происходит и со стороны фреймворка Express. Он также осуществляет поддержку MySQL. Непосредственно взаимодействие происходит только посредством SQL

запросов, а подключение к базе данных происходит благодаря специальному пакету mysql.

Таким образом, использование в качестве СУБД MySQL с дополнением в виде ORM Eloquet обеспечивает выполнение сразу нескольких требований к программному средству, а именно: обеспечение сохранности данных пользователей, автоматизация и унификация обращений к БД, а также использование удобного высокогоуровневого API, что позволяет повысить читаемость кода. При проектировании базы данных было выявлено отсутствие сложных связей между таблицами, объём данных не является чрезвычайно большим, что позволяет эффективно использовать ORM-систему без видимых потерь в производительности и быстродействии программного средства.

## 4.2 Реализация пользовательского интерфейса

Разработка пользовательского интерфейса является одной из определяющих частей при разработке программного средства. Непосредственно пользовательский интерфейс позволяет пользователям взаимодействовать с приложением и имеет ключевое значение для обеспечения удобства и эффективности работы пользователя с системой. Проектирование пользовательского интерфейса невозможно без чётко сформулированных функциональных и нефункциональных требований, определения стандартов безопасности и выбора методов их обеспечения. Пользовательский интерфейс уязвим с точки зрения обеспечения безопасности данных приложения и нуждается в гарантиях защиты от взлома, а также должен быть интуитивно понятным, обеспечивать быстрое выполнение задач, быстро реагировать на действия пользователя.

Для разработки пользовательского интерфейса была использована библиотека для разработки пользовательских интерфейсов React вместе с CSS-фреймворком tailwind. Одной из причин послужил компонентный подход, что позволяет декомпозировать весь интерфейс на более маленькие компоненты, что увеличивает читаемость кода и расширяемость системы. Другой причиной послужило достаточно простое взаимодействие с веб-сервером на базе фреймворка Laravel, а также websocket сервером на базе фреймворка Express. Причиной выбора фреймворка tailwind послужила быстрая разработка непосредственно части оформления.

Взаимодействие клиентской части на React с веб-сервером на Laravel необходимо для обеспечения аутентификации и авторизации пользователей, создания комнат для видеозвонков, управления списком запланированных звонков и происходит благодаря Inertia. Inertia позволяет создавать одностраничные приложения, полностью отображаемые на стороне клиента, без сложностей, свойственных современным одностраничным приложениям. Inertia не имеет маршрутизации на стороне клиента и не требует API. Inertia позволяет ускорить разработку одностраничного приложения. Приложения, написанные при помощи Inertia, выглядят и разрабатываются также, как

server-side rendering приложения, однако все представления Inertia – это компоненты React. Это достигается за счёт того, что по своей сути Inertia представляет собой библиотеку маршрутизации на стороне клиента. Это позволяет посещать страницы без принудительной полной перезагрузки страницы. Это делается с помощью Link компонента — легкой оболочки вокруг обычной ссылки. При нажатии на компонент ссылки, Inertia перехватывает клик и вместо этого совершает посещение через XHR. Когда Inertia совершает посещение XHR, сервер определяет, что это посещение Inertia, и вместо возврата полного ответа HTML возвращает ответ JSON с именем и данными компонента страницы JavaScript. Затем библиотека динамически заменяет предыдущий компонент страницы новым компонентом страницы и обновляет состояние истории браузера.

Взаимодействие клиентской части на React с websocket сервером на Express необходимо для организации видеозвонка и происходит благодаря библиотеке socket.io, основанной на веб-сокетах. Она использует веб-сокеты, когда они доступны, или такие технологии, как Flash Socket, AJAX Long Polling, AJAX Multipart Stream, когда веб-сокеты недоступны. Веб-сокеты позволяют клиенту установить такое соединение с сервером, при котором данные будут приходить при возникновении серверного события (event-driven) в режиме реального времени. В отличие от множества веб-технологий, веб-сокеты не следуют стратегии «Запрос-ответ», при которой соединение открывается для выполнения запроса, и закрывается сразу после его выполнения. Веб-сокеты держат соединение открытым. Следовательно, клиенту не требуется постоянно опрашивать сервер на предмет наличия обновлений, что значительно ускоряет быстродействие приложения при меньшем расходе ресурсов. Сам функционал видеозвонка на стороне клиента осуществляется за счёт следующих сервисов:

- сервис для работы с микрофоном AudioService, который отвечает за отключение и включение микрофона у себя либо других пользователей;
- сервис для работы с видео VideoService, который отвечает за отключение и включение видео у себя либо других пользователей;
- сервис PeerService для выполнения соединения пользователей во время звонка;
- сервис ShareScreenService для показа своего экрана другим пользователям во время звонка;
- сервис URLService для работы с url адресами.

### 4.3 Разработка сервиса аутентификации

Важными требованиями к программному средству для проведения онлайн- занятий и встреч, сформулированных в начале разработки, является проведения авторизации и регистрации пользователей, проверка на регистрацию либо авторизацию пользователя при выполнении определённых

действий, обеспечение безопасности хранимых пользовательских данных. Согласно международным стандартам, в частности, открытому стандарту RFC 7519, необходимо гарантировать легитимность запросов пользователя на любой доступ к данным, не зависимо от того, является ли целью операции изменение, удаление или же получение данных. Основной целью подобного механизма защиты является предотвращение подделки запросов мошенниками, а также защита от фишинговых и CSFR атак. Для выполнения данных требований необходим сервис для выполнения аутентификации и авторизации пользователей.

Для обеспечения аутентификации и авторизации, а также дополнительным безопасности используются различные средства и механизмы, такие как пользовательские сессии, шифрование данных, хеширование пароля, подтверждение электронной почты.

Пользовательская сессия – это определённый временной интервал, в ходе которого происходит взаимодействие пользователя с программным средством. Отсчёт данного временного интервала начинается с момента удостоверения программного средства в личности пользователя, т.е. с момента авторизации. В контексте разрабатываемого программного средства пользовательская сессия имеет ключевое значение ввиду того, что токен, получаемый пользователем при авторизации пользователя хранится именно в пользовательской сессии в базе данных и используется при каждом обращении пользователя к серверной части программного средства.

При использовании браузера для входа в сервис пользователь вводит свои данные форму входа. Если эти учетные данные верны, то приложение будет хранить информацию об аутентифицированном пользователе в сессии пользователя. Файл cookie, отправленный браузеру, содержит идентификатор сессии, чтобы последующие запросы к приложению могли связать пользователя с правильной сессией. После получения файла cookie сессии, приложение извлекает данные сессии на основе идентификатора сессии, отмечает, что аутентификационная информация была сохранена в сессии, и рассматривает пользователя как «аутентифицированного». Подобный механизм безопасности позволяет уменьшить риски таких атак, как похищение сессии и фиксация сессии, которые позволяют либо получить доступ к сервису на неограниченный срок, либо передать права на выполнение третьим лицам, неавторизованным в системе, что приводит к выполнению потенциально вредоносных запросов под личиной пользователя, авторизованного в системе. Более того, для обеспечения дополнительной защиты от взлома, все сессии, действующие во время создания новой сессии, автоматически признаются не валидными, что позволяет минимизировать риск захвата активной сессии злоумышленником.

Cookie – небольшой набор данных, отправляемый веб-сервером и хранимый на компьютере пользователя без изменений и какой-либо обработки. Веб-клиент всякий раз при обращении к соответствующему сайту пересыпает эти данные веб-серверу в составе HTTP-

запроса. cookie используется для аутентификации пользователей, хранения персональных предпочтений и настроек пользователя, отслеживания состояния сеанса доступа пользователя, хранения сведений статистики.

Laravel содержит встроенные службы аутентификации и сессии, которые обычно доступны через фасады Auth и Session. Данный функционал обеспечивают аутентификацию на основе файлов cookie для запросов, которые инициируются из веб-браузеров. Они предоставляют методы, которые позволяют проверять учетные данные пользователя и аутентифицировать пользователя. Кроме того, эти службы автоматически сохраняют необходимые данные аутентификации в сессии пользователя и выдают cookie сессии пользователя. Помимо встроенных служб Laravel предоставляет несколько пакетов, связанных с аутентификацией, таких как Passport либо Sanctum а также предоставляет стартовые комплекты для обеспечения аутентификации приложений. Одним из таких комплектов является Laravel Breeze, который включает в себя минимальную и простую реализацию необходимого набора функций аутентификации, включая вход в систему, регистрацию, сброс пароля, проверку и подтверждение электронной почты и подтверждение пароля. Помимо этого, Laravel Breeze включает в себя функционал для обновления своего имени, адреса электронной почты, сброса пароля и удаления аккаунта.

Laravel предоставляет простой и удобный интерфейс для шифрования и дешифрования текста через OpenSSL с использованием шифрования AES-256 и AES-128. Все зашифрованные значения Laravel подписываются с использованием кода аутентификации сообщения (MAC), поэтому их базовое значение не может быть изменено или подделано после шифрования.

Для хранения паролей пользователей в базе данных необходимо обеспечить хеширование. Для безопасного хеширования Bcrypt и Argon2 используется фасад Hash. По умолчанию для регистрации и аутентификации используется Bcrypt, который является отличным выбором для хеширования паролей, поскольку его рабочий коэффициент можно регулировать, а это означает, что время, необходимое для генерации хэша, может быть увеличено по мере увеличения мощности оборудования. При хешировании паролей медленный подход хорош. Чем дольше алгоритму требуется хеширование пароля, тем больше времени требуется злоумышленникам для создания таблиц всех возможных строковых хеш-значений, которые могут использоваться при атаках методом перебора на приложения.

Laravel предоставляет возможность создавать и настраивать посредники. Посредник или Middleware обеспечивает удобный механизм для проверки и фильтрации HTTP-запросов, поступающих приложение. Например, в Laravel по умолчанию содержится посредник, проверяющий аутентификацию взаимодействующего с приложением пользователя. Если пользователь не аутентифицирован, то посредник перенаправит пользователя на домашний экран приложения. Однако, если пользователь аутентифицирован, то посредник позволит запросу продолжить работу в

приложении. Гибкий механизм посредников позволяет определять систему доступов пользователей и разделять доставляемую информацию в зависимости от наличия регистрации, отсекая необходимость в многоуровневых проверках внутри методов выборки данных.

Посредники могут назначены конкретному маршруту или использоваться для обработки всех поступающих запросов, что обеспечивает возможность широкой настройки их использования для обработки данных в приложении.

Помимо встроенных служб аутентификации, Laravel также предлагает простой способ авторизации действий пользователя с конкретными ресурсами. Например, даже если пользователь аутентифицирован, то он может быть не авторизован для обновления или удаления определенных моделей Eloquent. Политики или Policies позволяют оперировать ролями пользователя, определяя уровни доступа. Политики следует использовать, когда возникает необходимость разрешить действие для конкретной модели или ресурса.

Вместо того чтобы вручную регистрировать политики модели, Laravel может автоматически обнаруживать политики, если модель и политика соответствуют стандартным соглашениям об именах Laravel, что значительно ускоряет и унифицирует процесс разработки и позволяет добиться единообразного стиля в приложении без значительных усилий со стороны разработчика.

Использование механизмов посредников и политик облегчает обработку данных и распределение прав доступа в программном средстве, отсекая необходимости формирования сложных условных конструкций для проверки, обеспечивая при этом достаточный уровень безопасности.

#### **4.4 Разработка веб-сокет сервиса**

Одним из основных требований к программному средству для проведения онлайн-занятий и встреч, сформулированных в начале разработки, является проведение видеозвонка. Также сформулированы требования для необходимого функционала во время звонка, включающий работу с видео и аудио, показа своего экрана другим пользователям, использование общей доски, подключение к видеозвонку по ссылке.

Для реализации необходимого функционала создания, проведения видеозвонка, а также дополнительного функционала во время звонка реализован websocket сервер, построенный на фреймворке Express и библиотеке socket.io. Главными преимуществами socket.io являются следующие:

- в отличие от веб-сокетов, socket.io позволяет отправлять сообщения всем подключенными клиентам с помощью одной операции;
- в веб-сокетах сложно использовать проксирование и балансировщики нагрузки. Socket.IO поддерживает эти технологии из коробки;

- socket.io поддерживает автоматическое переподключение при разрыве соединения;

- более лёгкая разработка.

Данный сервер отвечает за подключение новых пользователей, установке соединения между пользователями, уведомлении других пользователей об отключении от звонка, показ и скрытие камеры, микрофона. Работа сервера состоит в приёме событий от клиента и отправки уведомлений одному либо всем клиентам.

## 4.5 Разработка сервиса запланированных встреч

Для управления списком запланированных звонков необходимо реализовать функционал создания, редактирования, удаления звонка. Такой функционал можно реализовать с помощью CRUD операций.

Laravel по умолчанию предоставляет инструменты и механизмы для автоматического формирования маршрутов и конфигурации маршрутизации внутри программного средства. Для управления запланированными звонками, а также профилем зарегистрированного пользователя, реализованы классы для реализации всего функционала CRUD операций. Диаграмма классов управления профилем и запланированными звонками представлена на рис. 4.1.

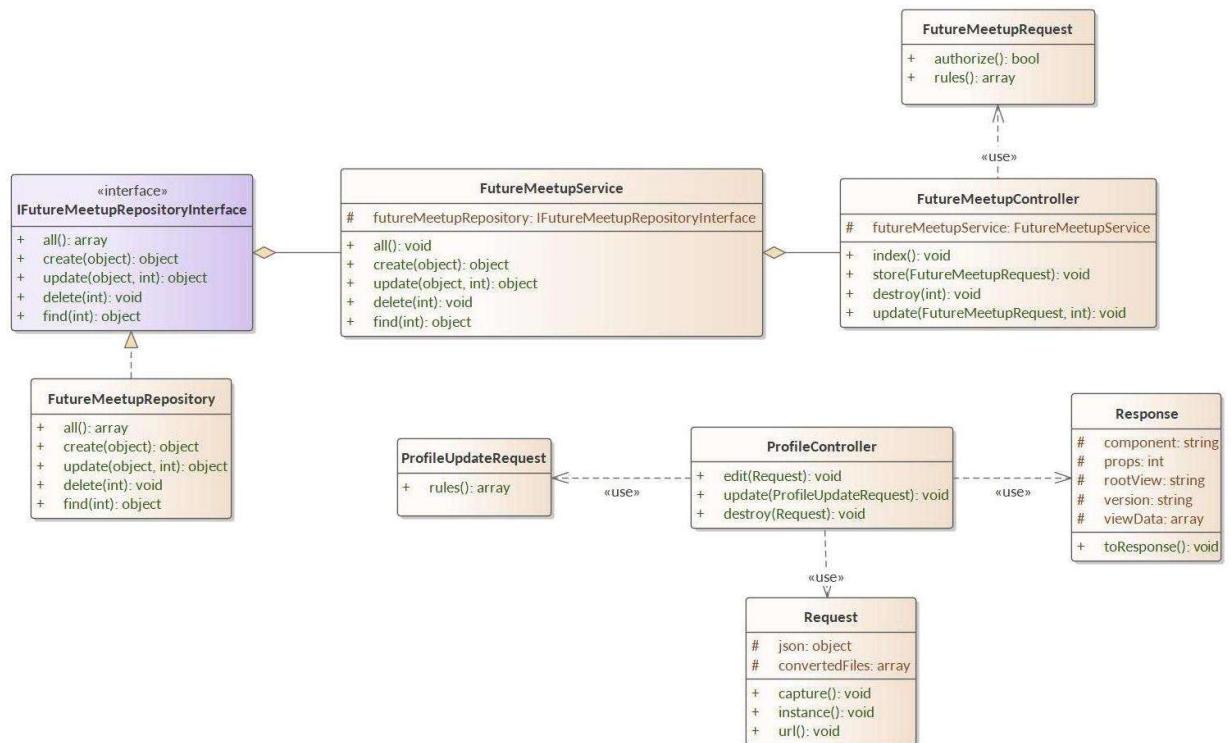


Рисунок 4.1 – Диаграмма классов

## 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Одним из обязательных этапов разработки программного средства является тестирование. Тестирование ПО – проверка соответствия между реальным и ожидаемым поведением программы, проводится на наборе тестов, который выбирается некоторым образом. Основными целями тестирования являются предоставление актуальной информации о состоянии продукта на данный момент и повышение лояльности компании к продукту. Тестирование как этап жизненного цикла программного средства позволяет повысить качество разработанного программного средства, проверить его на соответствие функциональным и нефункциональным требованиям и гарантировать работоспособность ПС в любых условиях эксплуатации, предусмотренных требованиями.

Фундаментально различают два типа методов тестирования, а именно структурные и функциональные. Структурное тестирование уделяет основное внимание детальному изучению логики программы и подборе тестов, которые гарантируют максимальное возможное число проверяемых логических ветвлений программы, её условий и операторов, и этот вид тестирования находит применение на более ранних этапах разработки программного средства. Структурное тестирование эффективно при выявлении ошибок циклов, путей и ветвлений. Полное покрытие кода с помощью структурного тестирования является сложным для реализации и применяется редко.

В свою очередь функциональное тестирование считается основным видом тестирования ПО. Каждая функция программы тестируется (проверяется на правильность в некоторых точках) и при этом делается вывод об ее правильности. Функциональное тестирование не столько является аналогом или заменителем структурного тестирования, сколько обязательным дополнением, так как позволяет обнаружить другие классы ошибок:

- некорректные функции;
- ошибки интерфейса;
- неточности во внешних структурах данных.

Для проведения функционального тестирования необходимо готовое функционирующее ПО. Следовательно, функциональное тестирование используется на поздних этапах разработки и предусматривает значительно более сложные сценарии. Зачастую функциональное тестирование проводится вручную, однако популярен подход с использованием автоматизированного тестирования.

Целями функционального тестирования являются:

- поиск в тестируемом ПО ошибок;
- документирование найденных ошибок с целью дальнейшего их исправления;
- определение соответствия тестируемого ПО предъявляемым к нему требованиям и принятие объективного заключения о возможности поставки

протестированного ПО заказчику.

В силу особенной и спецификаций рассмотренных выше видов тестирования, для проведения проверки работоспособности программного средства было выбрано функциональное тестирование. С целью его проведения был создан набор тестовых сценариев, определяющих наличие ошибок. Тестирование программного средства производилось на персональном компьютере с установленной операционной системой Windows 11.

Сформированные тестовые сценарии регистрации и авторизации представлены в таблице 5.1.

Таблица 5.1 – Тестовые сценарии регистрации и авторизации

Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	2	3	4
1 Регистрация с использованием электронной почты и пароля.	1. Открыть главную страницу сервиса. 2. Ввести имя в соответствующее поле. 3. Ввести адрес электронной почты в соответствующее поле. 4. Ввести пароль в соответствующее поле. 5. Нажать кнопку «Регистрация».	Отображение главной страницы с вкладкой меню пользователя.	Тест пройден успешно.
2 Регистрация с использованием имени, электронной почты и пароля. Валидация ошибок.	1. Открыть главную страницу сервиса. 2. Ввести имя, уже зарегистрированное в системе, в соответствующее поле 3. Ввести адрес электронной почты в соответствующее поле. 4. Ввести пароль в соответствующее поле. 5. Нажать кнопку «Регистрация».	Отображение сообщения о том, что указанное имя уже существует.	Тест пройден успешно.

Продолжение таблицы 5.1

1	2	3	4
3 Регистрация с использованием имени, электронной почты и пароля. Валидация ошибок.	1. Открыть главную страницу сервиса. 2. Ввести имя в соответствующее поле 3. Ввести адрес электронной почты, уже зарегистрированный в системе, в соответствующее поле. 4. Ввести пароль в соответствующее поле. 5. Нажать кнопку «Регистрация».	Отображение сообщения о том, что указанный адрес электронной почты уже существует.	Тест пройден успешно.
4 Регистрация с использованием имени, электронной почты и пароля. Валидация ошибок.	1. Открыть главную страницу сервиса. 2. Ввести имя в соответствующее поле 3. Ввести адрес электронной почты в соответствующее поле. 4. Ввести пароль в соответствующее поле. 5. Ввести подтверждение пароля, не совпадающее с паролем, соответствующее поле. 6. Нажать кнопку «Регистрация».	Отображение сообщения о том, что указанный и подтверждённый пароли не совпадают.	Тест пройден успешно
5 Авторизация с использованием электронной почты и пароля.	1. Открыть главную страницу сервиса. 2. Ввести адрес электронной почты, зарегистрированный в системе, в соответствующее поле. 3. Ввести пароль в соответствующее поле. 4. Нажать кнопку «Войти».	Отображение главной страницы с вкладкой меню пользователя.	Тест пройден успешно.

Продолжение таблицы 5.1

1	2	3	4
6 Авторизация с использование электронной почты и пароля. Валидация ошибок.	1. Открыть главную страницу сервиса. 2. Ввести адрес электронной почты, зарегистрированный в системе, в соответствующее поле. 3. Ввести некорректный пароль в соответствующее поле. 4. Нажать кнопку «Войти».	Отображение сообщения о том, что пароль или логин некорректен.	Тест пройден успешно.
7 Авторизация с использование электронной почты и пароля. Валидация ошибок.	1. Открыть главную страницу сервиса. 2. Ввести адрес электронной почты, не зарегистрированный в системе, в соответствующее поле. 3. Ввести некорректный пароль в соответствующее поле. 4. Нажать кнопку «Войти».	Отображение сообщения о том, что пароль или логин некорректен.	Тест пройден успешно.

Последующие тестовые сценарии требует предварительной авторизации для их успешного выполнения.

Таблица 5.2 – Тестовые сценарии для авторизованного пользователя

Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	2	3	4
1 Создание комнаты для проведения видеочата.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Начать встречу». 3. Нажать кнопку «Разрешить камеру и микрофон».	Переход на страницу комнаты с видеочатом.	Тест пройден успешно.

Продолжение таблицы 5.2

1	2	3	4
2 Подключение к существующей комнате с видеочатом.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Подключиться». 3. Ввести имя в соответствующее поле. 4. Ввести номер комнаты в соответствующее поле	Переход на страницу комнаты с видеочатом, который уже существует.	Тест пройден успешно
3 Открытие списка запланированных звонков.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Запланировать встречу».	Переход на страницу со списком всех запланированных звонков.	Тест пройден успешно.
4 Открытие календаря, в котором присутствуют запланированные звонки.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Запланировать встречу». 3. Нажать кнопку «Открыть календарь».	Отображение календаря со списком запланированных звонков.	Тест пройден успешно.
5 Добавление запланированного звонка.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Запланировать встречу». 3. Нажать кнопку «Добавить звонок». 4. Ввести название звонка. 5. Выбрать дату звонка в поле выбора даты. 6. Ввести описание.	Добавление запланированного звонка и возвращение на страницу со списком запланированных звонков.	Тест пройден успешно.

Продолжение таблицы 5.2

1	2	3	4
6 Удаление запланированного звонка.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Запланировать встречу». 3. Навести мышью на пункт в списке запланированных звонков, который необходимо удалить 4. Нажать кнопку «Удалить».	Удаление запланированного звонка.	Тест пройден успешно.
7 Редактирование запланированного звонка.	1. Открыть главную страницу сервиса. 2. Нажать кнопку «Запланировать встречу». 3. Навести мышью на пункт в списке запланированных звонков, который необходимо удалить 4. Нажать кнопку «Редактировать». 5. Ввести новое название звонка. 6. Ввести новую дату звонка.	Изменение названия и даты запланированного звонка.	Тест пройден успешно.
8 Выключение и включение собственного видео пользователя во время видеозвонка.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку выключения видео. 3. Нажать кнопку включения видео.	Собственное видео пользователя изначально выключается, затем включается.	Тест пройден успешно.

Продолжение таблицы 5.2

1	2	3	4
9 Выключение и включение собственного микрофона пользователя во время видеозвонка.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку выключения микрофона. 3. Нажать кнопку включения микрофона.	Собственный микрофон пользователя изначально выключается, затем включается.	Тест пройден успешно.
10 Завершение звонка.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку завершения звонка.	Выход из комнаты с видеочатом и переход в главное меню.	Тест пройден успешно.
11 Демонстрация экрана другим пользователям.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку демонстрации экрана. 3. Выбрать, какое приложение показывать. 4. Нажать кнопку «Поделиться».	Вместо видео пользователя происходит демонстрация экрана пользователя другим пользователям.	Тест пройден успешно.
12 Включение общей доски для всех участников звонка.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку для показа общей доски.	У всех пользователей на экране появляется доска.	Тест пройден успешно.
13 Копирование ссылки на комнату с видеочатом.	1. Повторить шаги, описанные в пункте 1 таблицы 5.2. 2. Нажать кнопку «Копировать ссылку».	Происходит копирование ссылки. Вместо надписи «Копировать ссылку» появляется надпись «Скопировано».	Тест пройден успешно.

Были проверены основные функциональные возможности приложения. Все тесты были пройдены успешно, что свидетельствует о том, что приложение успешно справилось с функциональными испытаниями.

## **6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ**

### **6.1 Установка программного средства**

Для корректной работы разрабатываемого программного средства персональный компьютер должен удовлетворять следующим минимальным требованиям:

- ОС: Windows 7;
- процессор: Intel Xeon E5 2,2 ГГц;
- объем ОЗУ: четыре Гб;
- жесткий диск: шесть Гб свободного места.

Перед установкой программного средства необходимо установить следующее программное обеспечение:

- СУБД MySQL 2019;
- Node.js версии 20.12.2;
- php версии 8.2.6;
- Laravel версии 11.4.0.

Для запуска веб-сервера необходимо зайти в папку laravel-server, выполнить команды `php artisan serve`, `npm i` и `npm run dev`. Для запуска веб-сокет сервера необходимо зайти в папку socket-server и выполнить команды `npm i` и `npm start`. В СУБД необходимо создать базу данных, которая в дальнейшем будет использоваться веб-сервисом. По необходимости можно сконфигурировать отдельного пользователя в СУБД для доступа к веб-сервису, которому необходимо предоставить права на чтение данных из созданной базы данных.

### **6.2 Руководство пользователя**

В представленном руководстве приведена основная информация об возможных этапах работы с программным средством.

Для начала работы пользователю не требуется установка и настройка программного средства. Необходимой и достаточной мерой является переход на сайт с использованием браузера пользователя.

При запуске веб-приложения пользователю будет отображена домашняя страница приложения. На домашней странице представлены кнопки «Вход», «Зарегистрироваться» и «Подключиться». Сверху над главным меню показан основной логотип приложения, который является ссылкой. Экран домашней страницы представлен на рис. 6.1.

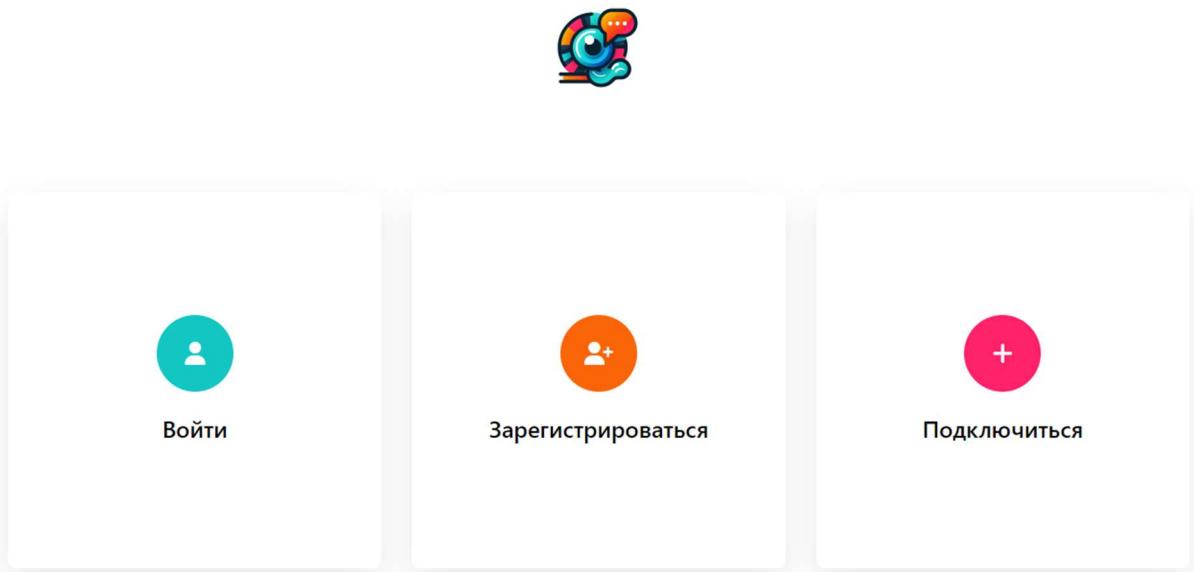


Рисунок 6.1 – Главный экран приложения

Для регистрации в приложении необходимо нажать кнопку «Зарегистрироваться». При нажатии кнопки происходит переход в меню регистрации, где необходимо заполнить своё имя, адрес электронной почты, пароль и подтверждение пароля. Пароль должен быть более 8 символов. Меню регистрации представлено на рис. 6.2.

The image shows the registration menu. At the top center is the same stylized logo. Below the logo are four input fields. The first field is labeled 'Имя' (Name) and contains a placeholder ' '. The second field is labeled 'Email' and contains a placeholder ' '. The third field is labeled 'Пароль' (Password) and contains a placeholder ' '. The fourth field is labeled 'Подтвердить пароль' (Confirm Password) and contains a placeholder ' '. At the bottom left of the form is a link 'Уже зарегистрированы?' (Already registered?). At the bottom right is a large black button with the text 'РЕГИСТРАЦИЯ' (Registration).

Рисунок 6.2 – Меню регистрации

Для авторизации в приложении необходимо нажать кнопку «Вход». При

нажатии кнопки происходит переход в меню авторизации, где необходимо заполнить адрес электронной почты и пароль. Меню авторизации представлено на рис. 6.3.

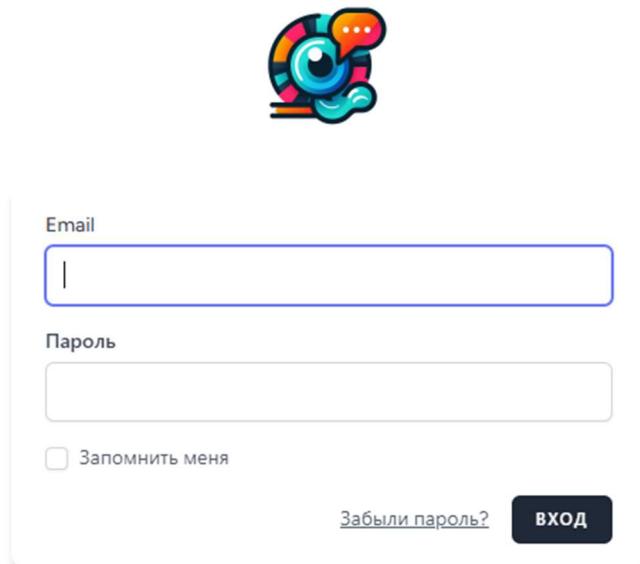


Рисунок 6.3 – Меню авторизации

Для подключения к видеозвонку необходимо нажать кнопку «Подключиться». При нажатии кнопки происходит открытие модального окна с формой ввода номера комнаты. Необходимо ввести номер комнаты видеозвонка, к которой необходимо подключиться. Модальное окно для подключения к видеозвонку представлено на рис. 6.4.

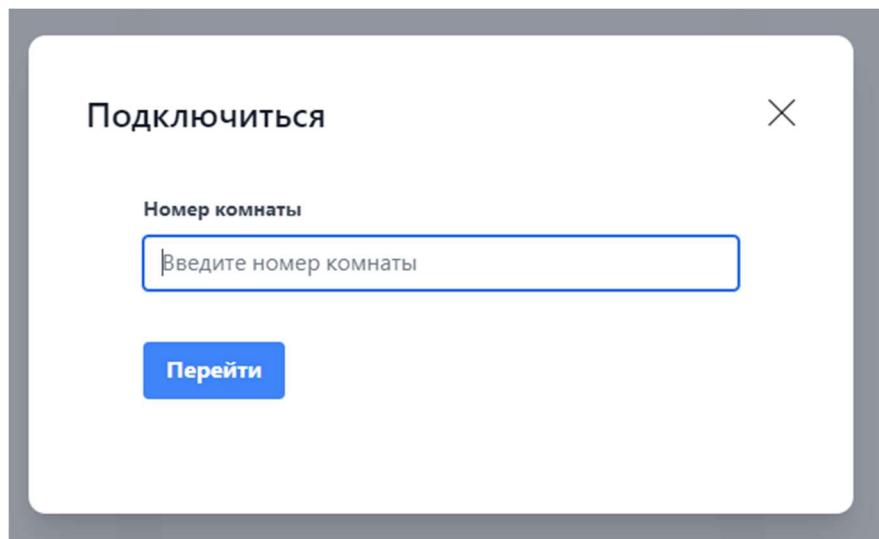


Рисунок 6.4 – Модальное окно для подключения к видеозвонку

После выполнения регистрации либо авторизации происходит переход в главное меню авторизованного пользователя. В главном меню представлены кнопки «Начать встречу», «Подключиться» и «Запланировать встречу». Сверху над главным меню показан основной логотип приложения, который является ссылкой и имя пользователя с меню настройки профиля. Главное меню авторизованного пользователя представлено на рис. 6.5.

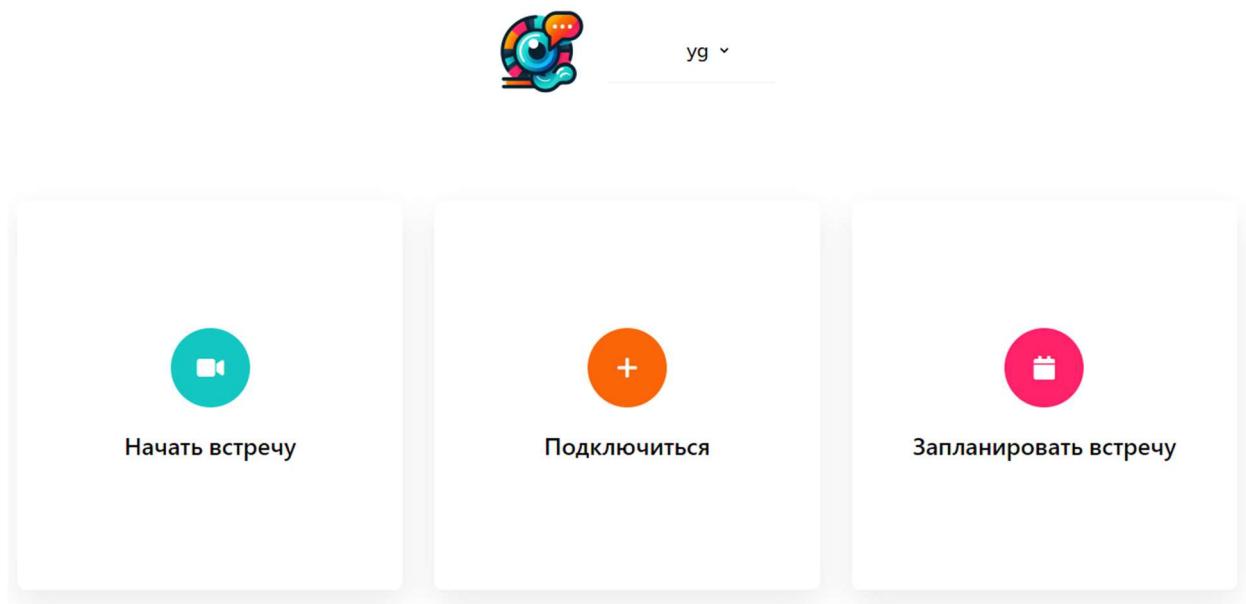


Рисунок 6.5 – Главное меню авторизованного пользователя

При нажатии на имя профиля появляется всплывающее меню в котором присутствуют кнопки для настройки профиля «Профиль», «Выйти». Для выхода из аккаунта необходимо нажать на кнопку «Выйти». Главное меню авторизованного пользователя представлено на рис. 6.6.

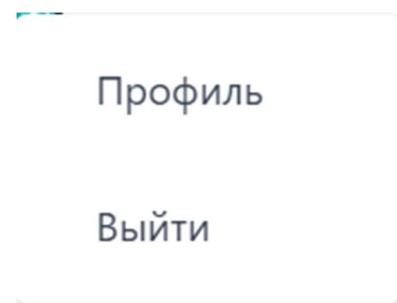


Рисунок 6.6 – Всплывающее меню для перехода в настройки профиля

При нажатии на кнопку «Профиль» происходит переход в меню настройки профиля, в котором есть возможность изменить своё имя профиля, адрес электронной почты, пароль, удалить аккаунт. Меню настройки профиля представлено на рис. 6.7.

**Информация о профиле**

Обновите информацию профиля вашей учетной записи и адрес электронной почты.

Имя

Email

**СОХРАНИТЬ**

---

**Изменить пароль**

Убедитесь, что в вашей учетной записи используется длинный случайный пароль, чтобы обеспечить безопасность.

Текущий пароль

Новый пароль

Подтвердите новый пароль

**СОХРАНИТЬ**

Рисунок 6.7 – Меню настройки профиля

При нажатии на кнопку «Начать встречу» происходит переход на страницу с комнатой для проведения видеочата. Перед началом звонка необходимо предоставить браузеру разрешение на пользование камерой и микрофоном. В комнате для видеочата располагаются видео участников видеочата, а также нижнее меню, где представлены кнопки для включения и отключения своего микрофона либо видео, копирования ссылки для захода других участников, демонстрации своего экрана, открытия доски для общего использования всеми участниками видеочата, завершения звонка. Также для каждого видео присутствуют элементы управления для остановки видео, увеличения либо уменьшения звука, открытия видео в полный экран, включения функции «Картинка в картинке». Создатель видеовстречи назначается администратором видеовстречи. У администратора видеовстречи есть дополнительная возможность включать и выключать микрофоны и видео других пользователей. Для подключения другим пользователям к текущей комнате необходимо перейти по ссылке, который пришлёт организатор встречи либо нажать на кнопку «Подключиться» и ввести номер комнаты.

Комната для видеочата представлена на рисунке 6.8.

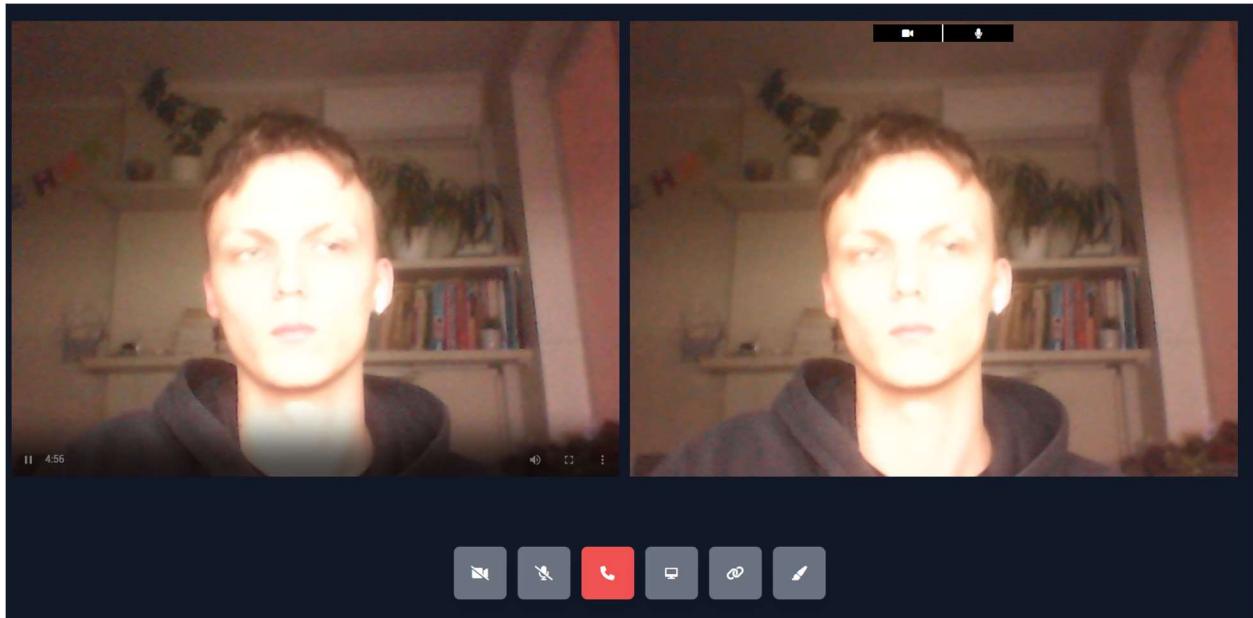


Рисунок 6.8 – Комната видеочата

Для демонстрации своего экрана необходимо нажать на кнопку демонстрации экрана в нижнем меню комнаты видеочата. После нажатия кнопки демонстрации экрана необходимо выбрать экран, который нужно продемонстрировать и нажать на кнопку «Поделиться». Функция демонстрации экрана представлена на рисунке 6.9.

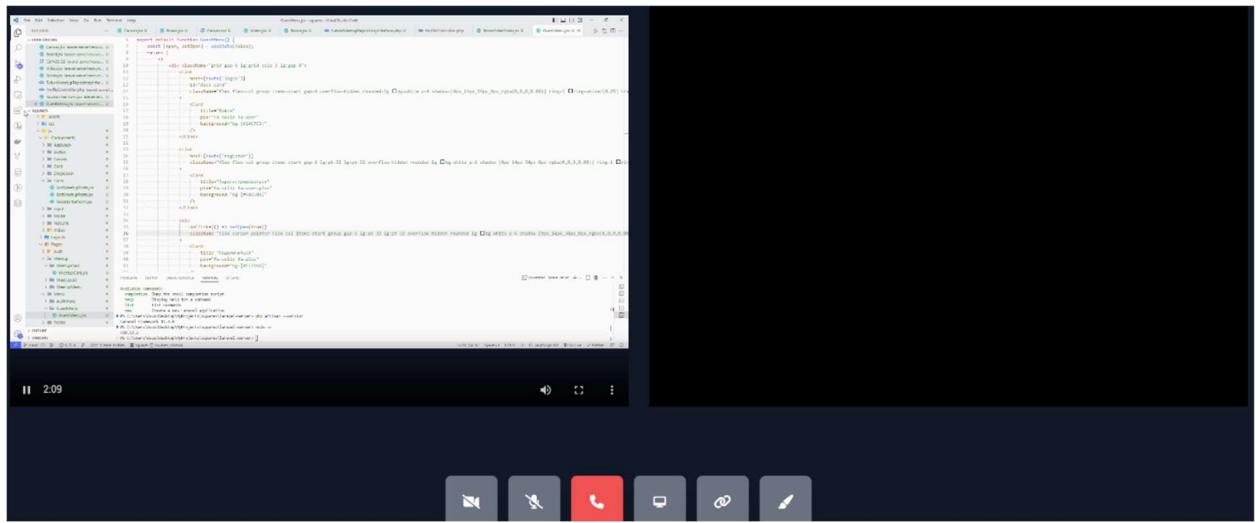


Рисунок 6.9 – Демонстрация экрана

Для открытия доски для совместной работы необходимо нажать соответствующую кнопку для открытия доски в нижнем меню комнаты видеочата. После нажатия кнопки отобразиться доска, а также слева от доски

меню, в котором можно выбрать цвет кисти для рисования, толщину кисти, кнопки для очистки доски и её скачивания. Помимо рисования на доску можно скопировать изображение. Доска для совместной работы показана на рисунке 6.10.

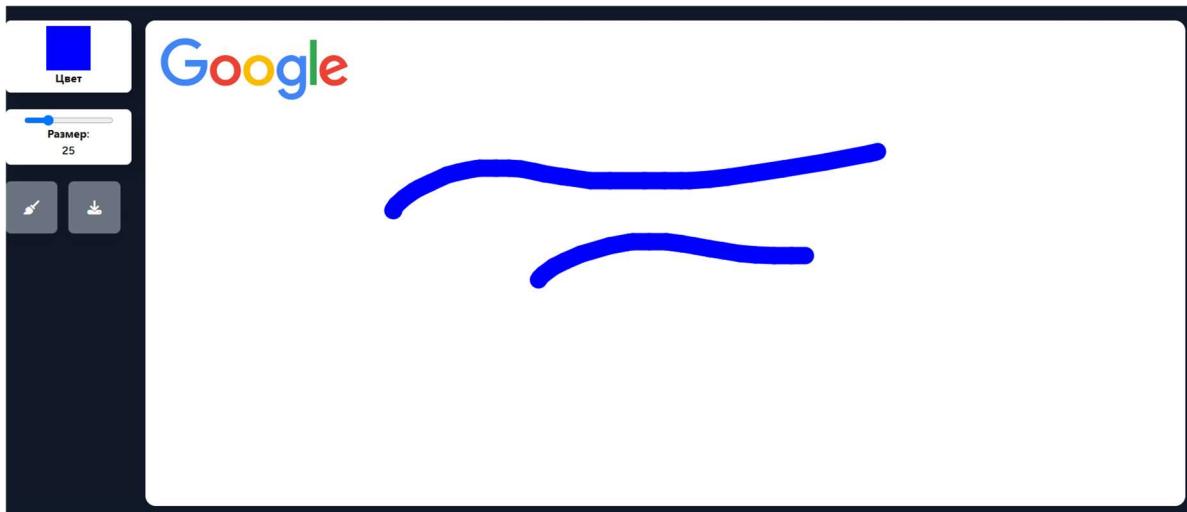


Рисунок 6.10 – Совместная доска

Для просмотра списка Запланированных встреч необходимо нажать на кнопку «Запланировать встречу». После нажатия кнопки происходит переход на страницу с запланированными встречами. Над списком запланированных встреч находится меню, в котором можно открыть список запланированных встреч в виде календаря или добавить встречу. Список запланированных встреч показан на рисунке 6.11.

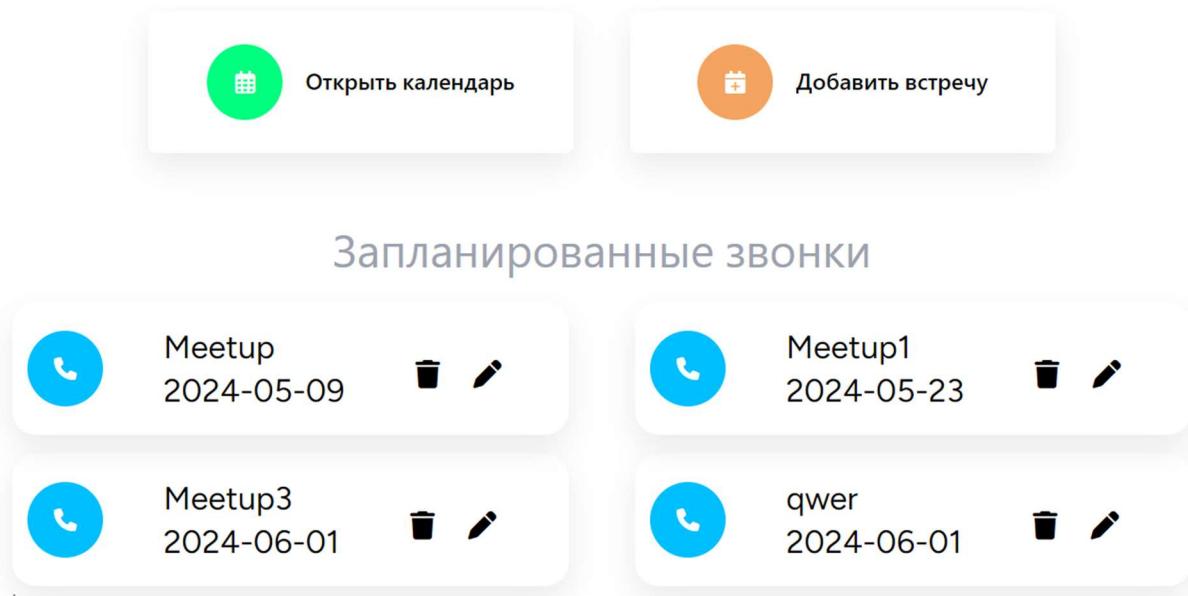


Рисунок 6.11 – Список запланированных встреч

Для добавления запланированной встречи необходимо нажать на кнопку «Добавить встречу». После нажатия на кнопку происходит открытие модального окна с формой для добавления запланированной встречи. Для добавления необходимо ввести название, дату и описание видеовстречи. Модальное окно для добавление запланированной встречи показано на рисунке 6.12.

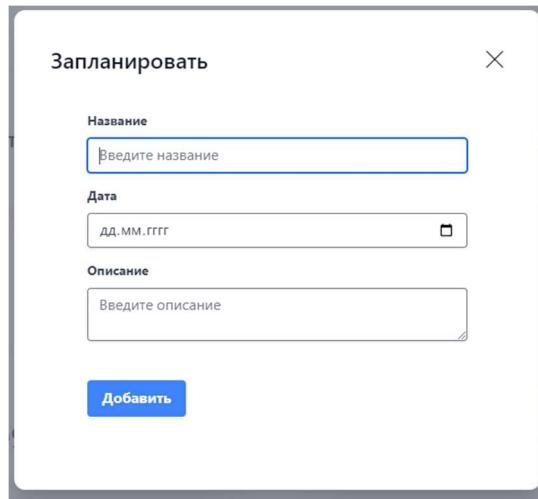


Рисунок 6.12 – Модальное окно для добавления запланированной встречи

Для удаления запланированной видеовстречи необходимо нажать на кнопку удаления соответствующей карточки. Для редактирования запланированной видеовстречи необходимо нажать на кнопку редактирования соответствующей карточки. После нажатия на кнопку происходит открытие модального окна с формой для редактирования запланированной встречи с заранее созданными данными. Модальное окно для изменения запланированной встречи показано на рисунке 6.13.

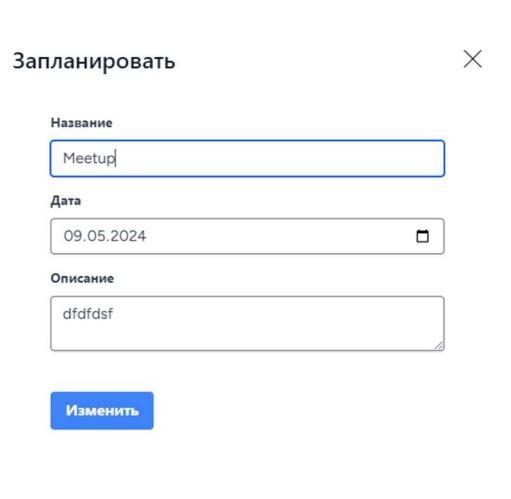


Рисунок 6.13 – Модальное окно для изменения запланированной встречи

## **7 ТЕХНИКО-ЭКОНОМИЧЕСКО ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПЛАТФОРМЫ ДЛЯ ПРОВЕДЕНИЯ ОНЛАЙН-ЗАНЯТИЙ И ВСТРЕЧ НА СТЕКЕ LARAVEL И REACT.JS**

### **7.1 Описание назначения, функций и потенциальных пользователей программного средства**

Платформа для проведения онлайн-занятий и встреч предназначена для обеспечения более эффективного и удобного взаимодействия между людьми в виртуальном пространстве, позволяя людям видеть друг друга и обмениваться файлами и документами в режиме реального времени. Платформа такого рода также обходится дешевле, чем командировочные расходы и экономят время, которое в другом случае было бы потрачено на поездки на различные встречи. С помощью платформы для видеоконференции можно провести встречу из любой точки мира, не покидая офис, учебное учреждение либо дом. Она позволяет пользователям организовывать и проводить различные виды онлайн-мероприятий.

Платформа для проведения онлайн-занятий и встреч предлагает следующий функционал:

- возможность создавать виртуальные комнаты для проведения видеоконференций, позволяя участникам видеть и слышать друг друга в режиме реального времени;
- предоставление инструмента для совместной работы над проектами и идеями, позволяя участникам делиться заметками, рисунками и другой необходимой информацией;
- предоставление чата для обмена текстовыми сообщениями во время встречи для обсуждения вопросов, которые не требуют голосового общения;
- возможность делиться содержимым своего экрана с другими участниками для демонстрации презентаций;
- возможность записи встречи для последующего просмотра для тех, кто не смог присутствовать на встрече в реальном времени;
- возможность перевода в режиме реального времени для того, чтобы стереть языковой барьер между людьми в разных точках мира.

В связи с тем, что возрос спрос на удалённый и гибридный форматы работы [1], онлайн-образование [2], возросла и необходимость в простой для освоения и функциональной платформе для онлайн-занятий и встреч. Потенциальными пользователями платформы являются:

- учебные заведения, в том числе школы, колледжи и университеты, которые могут использовать платформу для проведения онлайн-занятий и лекций;
- коммерческие и некоммерческие организации, которые могут

использовать платформу для проведения встреч, семинаров, тренингов и презентаций;

– тренеры и консультанты, которые могут использовать платформу для проведения и онлайн-сессий с клиентами;

– частные лица, которые могут использовать платформу для личного общения, обучения или работы над совместными проектами.

Использование платформы для проведения онлайн-занятий и встреч может привести к следующим результатам:

– улучшение качества онлайн-образования и увеличение готовности учреждений среднего и высшего образования к переходу на дистанционное обучение в случае необходимости;

– уменьшение расходов коммерческих и некоммерческих организаций на офисные и командировочные нужды;

– увеличение эффективности работы над различными проектами частных лиц.

Основным способом реализации продукта на рынке является продажа клиенту подписки, которая расширяет функционал для пользования платформой.

## 7.2 Расчет затрат на разработку ПО

Расчёт затрат на разработку модуля следует делать в разрезе следующих статей:

- затраты на основную заработную плату разработчиков;
- затраты на дополнительную заработную плату разработчиков;
- отчисления на социальные нужды;
- прочие затраты (амortизационные отчисления, расходы на электроэнергию, командировочные расходы, арендная плата за офисные помещения и оборудование, расходы на управление и реализацию и т.п.).

**7.2.1** Расчет затрат на основную заработную плату команды разработчиков производится по следующей формуле:

$$Z_o = K_{\text{пр}} \sum_{i=1}^n Z_{\text{ч},i} t_i \quad (7.1)$$

где  $n$  – количество исполнителей, занятых разработкой конкретного ПО;

$K_{\text{пр}}$  – коэффициент премий ( $K_{\text{пр}} = 1,3$ );

$Z_{\text{ч},i}$  – часовая заработка исполнителя  $i$ -го, р.;

$t_i$  – трудоемкость работ, выполняемых  $i$ -м исполнителем, ч.

Расчёт затрат на основную заработную плату осуществляется в табличной форме (табл. 7.1).

Таблица 7.1 – Расчёт затрат на основную заработную плату разработчиков

Наименование должности разработчика	Вид выполняемой работы	Месячная заработная плата, р.	Часовая заработная плата, р.	Трудоёмкость работ, ч	Сумма, р.
1	2	3	4	5	6
1. Руководитель проекта	Руководство проектом	5500	32,73	300	9819,00
2. Бизнес-аналитик	Анализ рынка	2500	14,88	80	1190,04
3. Программист	Проектирование и разработка клиентской части	3400	20,23	240	4855,20
4. Инженер-программист	Проектирование и разработка серверной части	4200	25,00	240	6000,00
5. Специалист по тестированию программного обеспечения	Проверка работоспособности	2400	14,28	72	1028,16
6. Специалист по маркетингу	Продвижение продукта на рынке	2000	11,90	80	952,00
Итого					23844,40
Премия (30%)					7153,30
Всего основная заработная плата					30997,70

**7.2.2** Дополнительная заработная плата включает выплаты, предусмотренные законодательством о труде, и рассчитывается по формуле 7.2:

$$Z_d = \frac{Z_o \cdot H_d}{100}, \quad (7.2)$$

где  $Z_o$  – затраты на основную заработную плату разработчиков, р.;

$H_d$  – норматив дополнительной заработной платы, ( $H_d = 15\%$ ).

Дополнительная заработка плата разработчиков составит:

$$Z_d = \frac{30997,70 \cdot 15}{100} = 4649,66 \text{ руб.}$$

**7.2.3** Отчисления на социальные нужды включают в предусмотренные законодательством отчисления в фонд социальной защиты населения и на обязательное страхование и рассчитываются по формуле:

$$P_{соц} = \frac{(Z_o + Z_d) \cdot H_{соц}}{100}, \quad (7.3)$$

где  $H_{соц}$  – норматив отчислений от фонда оплаты труда ( $H_{соц} = 35\%$ ).

Отчисления на социальные нужды составят:

$$P_{соц} = \frac{(30997,70 + 4649,66) \cdot 35}{100} = 12476,58 \text{ руб.}$$

**7.2.4** Прочие затраты включают аренду помещения, амортизацию основных средств производства, затраты на инфраструктуру, хранение данных.

Прочие затраты рассчитываются по формуле 7.4:

$$P_{пз} = \frac{Z_o \cdot H_{пз}}{100}, \quad (7.4)$$

где  $H_{пз}$  – норматив прочих затрат, ( $H_{пз} = 65\%$ ).

Таким образом, прочие затраты составят:

$$P_{пз} = \frac{30997,70 \cdot 65}{100} = 20148,51 \text{ руб.}$$

Полученные значения затрат на разработку по основным статьям представлены в таблице 7.2.

Таблица 7.2 – Затраты на разработку программного средства

Статья затрат	Сумма, руб.
Основная заработка команда разработки	30997,70
Дополнительная заработка команда разработки	4649,66
Отчисления на социальные выплаты	12476,58

Продолжение таблицы 7.2

Статья затрат	Сумма, руб.
Прочие затраты	20148,51
Общая сумма затрат на разработку	68272,45

### 7.3 Расчет экономического эффекта при разработке ПО для свободной реализации на рынке информационных технологий

Экономический эффект организации-разработчика программного средства представляет собой прирост чистой прибыли от продажи платных подписок.

Цена платной подписки была определена на основе цен схожих платформ для проведения онлайн-занятий и встреч. За основу взяты платформы Zoom и МТС Линк. Стоимость платной подписки платформы Zoom варьируется от 14,99 до 21,99 долларов в месяц [3], что в переводе составляет от 48,93 до 71,76 рублей. Стоимость платной подписки МТС Линк варьируется от 1249 до 11823 российских рублей [4], что в переводе составляет от 43,60 до 412,70 рублей. Исходя из цен на платные подписки аналогичных платформ, представленных на рынке, решено определить стоимость платной подписки равной 35 рублей.

Учитывая, что разрабатываемая платформа имеет достаточно широкую сферу применения, спрос на удалённый и гибридный форматы работы и онлайн-образование возрос, при этом не все потенциальные клиенты захотят приобретать платную подписку, число копий, которые возможно продать, оценивается в 10000 копий за год.

Расчет прибыли без учета налога на прибыль осуществляется по формуле 7.5:

$$\Pi = (\Pi_{\text{отп}} \cdot N - \text{НДС}) \cdot P_{\text{пр}} - I_{\text{разр}} \quad (7.5)$$

где  $\Pi_{\text{отп}}$  – отпускная цена копии (лицензии) программного средства, р.;

$N$  – количество копий (лицензий) программного средства, реализуемое за год, шт.;

НДС – сумма налога на добавленную стоимость, р.;

$P_{\text{пр}}$  – рентабельность продаж копий (лицензий) ( $P_{\text{пр}} = 35\%$ ).

Сумма НДС рассчитывается по формуле 7.6:

$$\text{НДС} = \frac{\Pi \cdot N \cdot H_{\text{дс}}}{100 + H_{\text{дс}}}, \quad (7.6)$$

где  $H_{\text{дс}}$  – ставка налога на добавленную стоимость согласно действующему законодательству ( $H_{\text{дс}} = 20\%$ ).

Таким образом, налог составит:

$$\text{НДС} = \frac{35 \cdot 10000 \cdot 20}{100 + 20} = 58333,33 \text{ руб.}$$

Значение прибыли составит:

$$\Pi = (35 * 10000 - 58333,33) * 0,35 - 68272,45 = 33810,88 \text{ руб.}$$

Расчет чистой прибыли осуществляется по формуле 7.7:

$$\Pi_{\text{ч}} = \Pi \cdot \left(1 - \frac{H_{\Pi}}{100}\right) \quad (7.7)$$

где  $H_{\Pi}$  – ставка налога на прибыль, 20%.

$$\Pi_{\text{ч}} = 33810,88 \cdot \left(1 - \frac{20}{100}\right) = 27048,26 \text{ руб.}$$

Уровень рентабельности рассчитывается по формуле 7.8:

$$y_p = \frac{\Pi_{\text{ч}}}{I_{\text{разр}}} \cdot 100 \quad (7.8)$$

Уровень рентабельности составит:

$$y_p = \frac{27048,26}{68272,45} \cdot 100 = 39,62\%$$

#### **7.4 Расчёт показателей эффективности инвестиций в разработку ПО**

Срок окупаемости можно рассчитать по формуле 7.9:

$$T_{\text{ок}} = \frac{\text{НДС} + I_{\text{разр}}}{\Pi_M}, \quad (7.9)$$

где  $\Pi_M$  – ежемесячная прибыль.

Ежемесячную прибыль можно рассчитать по формуле 7.10:

$$\Pi_M = \frac{I \cdot N}{12} \cdot \left(1 - \frac{H_{\Pi}}{100}\right), \quad (7.10)$$

где  $\Pi$  – цена реализации ПО заказчику, р.;  
 $H_{\pi}$  – ставка налога на прибыль ( $H_{\pi} = 20\%$ ).  
Ежемесячная прибыль составит:

$$\Pi_m = \frac{35 \cdot 10000}{12} \cdot \left(1 - \frac{20}{100}\right) = 23333,33 \text{ руб.}$$

Срок окупаемости составит:

$$T_{ok} = \frac{58333,33 + 68272,45}{23333,33} = 5,43 \text{ месяцев}$$

## 7.5 Вывод технико-экономического обоснования

В результате проведения технико-экономического обоснования разработки платформы для проведения онлайн-занятий и встреч были получены значения основных экономических показателей. Общие затраты на разработку составили 68272,45 руб., чистая прибыль – 27048,26 руб. Уровень рентабельности составил 39,62% при цене реализации 35 рублей и планируемом количестве проданных копий 10000 за год. Исходя из того, что показатель уровня рентабельности превышает ставку по долгосрочным депозитам, можно сделать вывод о целесообразности разработки и реализации программного средства.

Согласно данным банков Республики Беларусь, на март 2024 года средняя ставка по долгосрочным депозитам для юридических лиц составляет 2,48% [5]. Уровень рентабельности программного средства составил 39,62% при цене реализации 35 рублей и планируемом количестве проданных копий 10000.

Срок окупаемости программного средства составит 5,43 месяцев. Однако реальный срок окупаемости программного средства может отличаться от рассчитанного, так как при расчётах был принят ряд условностей, среди которых стабильная ситуация на рынке программного обеспечения, объём и срок продаж не меньше запланированного, неизменная цена.

## **ЗАКЛЮЧЕНИЕ**

В ходе работы над дипломным проектом была разработана платформа для проведения онлайн-занятий и встреч.

В процессе разработки произведен анализ предметной области. Были изучены литературные источники, исследованы существующие аналоги. Результатом этого анализа явилось обобщение преимуществ и недостатков существующих решений, которые были учтены при разработке функциональных требований данного проекта. Основными недостатками являются излишняя сложность программных средств, недостаточный функционал для взаимодействия участников видеовстречи.

На основе функциональных требований было произведено моделирование и проектирование программного средства, включающее в себя диаграмму развертывания, графическое представление модели базы данных, диаграмму классов серверной части программного средства, диаграмму вариантов использования, различные алгоритмы, среди которых можно выделить алгоритмы регистрации нового пользователя, создания новой комнаты, подключения нового пользователя к комнате, демонстрации экрана, модерации видеовстречи.

Согласно функциональным требованиям были разработаны тестовые сценарии, которые были успешно пройдены в ходе тестирования программного средства.

На завершающем этапе подробно описано руководство использования программного средства, которое включает в себя руководство для развертывания программного средства и его использование. Руководство использования позволяет полностью освоить работу с программным средством.

Также в ходе работы над дипломным проектом была рассмотрена экономическая сторона разработки. Полученные результаты свидетельствуют о том, что разработка программного средства является рентабельной.

Главной целью данного проекта было предоставление удобной платформы для проведения онлайн-занятий и встреч. Цель была успешна достигнута. При этом программное средство имеет значительный потенциал для дальнейшего совершенствования путем добавления нового функционала, такого, как функция голосового перевода в текст, передачу файлов больших размеров, размер комнаты на 80 и более человек. Таким образом, созданный продукт может оказаться востребованным на рынке сервисов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Понятие видеоконференции [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/vebinary-i-videokonferentsii-v-sisteme-distantionnogo-obucheniya/viewer>. – Дата доступа: 19.04.2024.
- [2] Понятие стандарта WebRTC [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/timeweb/articles/656947> – Дата доступа: 13.04.2024.
- [3] Официальный сайт IETF [Электронный ресурс]. – Режим доступа: <https://www.ietf.org/>. – Дата доступа: 13.04.2024.
- [4] Документация WebRTC [Электронный ресурс]. – Режим доступа: <https://www.w3.org/TR/webrtc/>. – Дата доступа: 13.04.2024.
- [5] Понятие протокола SDP [Электронный ресурс]. – Режим доступа: <https://www.3cx.ru/voip-sip/sdp/>. – Дата доступа: 13.04.2024.
- [6] Понятие стандарта REST [Электронный ресурс]. – Режим доступа: <https://www.codecademy.com/article/what-is-rest/>. – Дата доступа: 15.04.2024.
- [7] Спецификация RFC 6455 [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc6455/>. – Дата доступа: 15.04.2024.
- [8] Спецификация RFC 5245 [Электронный ресурс]. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc5245/>. – Дата доступа: 16.04.2024.
- [9] Понятие STUN-сервера [Электронный ресурс]. – Режим доступа: <https://www.3cx.ru/voip-sip/stun-server/>. – Дата доступа: 16.04.2024.
- [10] Понятие протокола DTLS [Электронный ресурс]. – Режим доступа: <https://www.3cx.ru/voip-sip/dtls/>. – Дата доступа: 16.04.2024.
- [11] Понятие протокола SRTP [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/SRTP/>. – Дата доступа: 16.04.2024.
- [12] Понятие RTP [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Real-time\\_Transport\\_Protocol/](https://ru.wikipedia.org/wiki/Real-time_Transport_Protocol/). – Дата доступа: 17.04.2024.
- [13] Понятие SCTP [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/SCTP/>. – Дата доступа: 17.04.2024.
- [14] Официальный сайт Zoom [Электронный ресурс]. – Режим доступа: <https://zoom.us/ru/>. – Дата доступа: 17.04.2024.
- [15] Официальный сайт Microsoft [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/ru-ru/microsoft-teams/log-in/>. – Дата доступа: 17.04.2024.
- [16] Официальный сайт Google Meet [Электронный ресурс]. – Режим доступа: <https://meet.google.com/?pli=1>. – Дата доступа: 17.04.2024.
- [17] Официальный сайт Jitsi Meet [Электронный ресурс]. – Режим доступа: <https://meet.jit.si/>. – Дата доступа: 18.04.2024.
- [18] Официальный сайт Яндекс Телемост [Электронный ресурс]. – Режим доступа: <https://telemost.yandex.ru/j/11208570373210/>. – Дата доступа: 18.04.2024.

- [19] Официальный сайт МТС Линк [Электронный ресурс]. – Режим доступа: <https://mts-link.ru/>. – Дата доступа: 18.04.2024.
- [20] Документация языка Javascript [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/JavaScript/>. – Дата доступа: 19.04.2024.
- [21] Документация языка PHP [Электронный ресурс]. – Режим доступа: <https://www.php.net/docs.php/>. – Дата доступа: 19.04.2024.
- [22] Документация библиотеки React.Js [Электронный ресурс]. – Режим доступа: <https://react.dev/>. – Дата доступа: 19.04.2024.
- [23] Документация фреймворка Laravel [Электронный ресурс]. – Режим доступа: <https://laravel.com/docs/11.x/readme/>. – Дата доступа: 20.04.2024.
- [24] Документация фреймворка Express.Js [Электронный ресурс]. – Режим доступа: <https://expressjs.com/ru/>. – Дата доступа: 20.04.2024.
- [25] Официальный сайт MySQL [Электронный ресурс]. – Режим доступа: <https://www.mysql.com/>. – Дата доступа: 20.04.2024.
- [26] Официальный сайт GitHub [Электронный ресурс]. – Режим доступа: <https://github.com/>. – Дата доступа: 20.04.2024.
- [27] Официальный сайт VS Code [Электронный ресурс]. – Режим доступа: <https://code.visualstudio.com/>. – Дата доступа: 21.04.2024.
- [28] Введение в язык UML [Электронный ресурс]. – Режим доступа: [https://www.bsuir.by/m/12\\_100229\\_1\\_65759.pdf/](https://www.bsuir.by/m/12_100229_1_65759.pdf/). – Дата доступа: 21.04.2024.

## ПРИЛОЖЕНИЕ А (обязательное)

### Исходный код серверной части

```
<?php

use App\Http\Controllers\FutureMeetupController;
use App\Http\Controllers\ProfileController;
use Illuminate\Foundation\Application;
use Illuminate\Support\Facades\Route;
use Inertia\Inertia;

Route::get('/', function () {
    return Inertia::render('Welcome/Welcome', [
        'canLogin' => Route::has('login'),
        'canRegister' => Route::has('register'),
        'laravelVersion' => Application::VERSION,
        'phpVersion' => PHP_VERSION,
    ]);
})->name('main');

Route::get('/room/{id}', function () {
    return Inertia::render('Room/Room');
})->name('room');

Route::controller(FutureMeetupController::class)->prefix('meetups')-
>group(function () {
    Route::get('/', 'index')->name('meetups.index');
    Route::post('/', 'store')->name('meetups.store');
    Route::delete('/{id}', 'destroy')->name('meetups.destroy');
    Route::patch('/{id}', 'update')->name('meetups.update');
})->middleware(['auth', 'verified'])->name('meetups');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->name('pro-
file.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('pro-
file.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])-
>name('profile.destroy');
});

require __DIR__.'/auth.php';

<?php

namespace App\Http\Controllers;

use App\Services\FutureMeetupService;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Log;
use Inertia\Inertia;

class FutureMeetupController extends Controller
{
    public function __construct(protected FutureMeetupService $future-
MeetupService)
    {
    }
}
```

```

public function index()
{
    $meetups = $this->futureMeetupService->all();

    return Inertia::render('Meetup/MeetupMenu/MeetupMenu', ['meetups' =>
$meetups]);
}

public function store(Request $request)
{
    Log::channel('stderr')->info($request);
    $meetup = $this->futureMeetupService->create($request);

    return redirect(route('meetups.index'));
}

public function destroy($id)
{
    Log::channel('stderr')->info($id);
    $this->futureMeetupService->delete($id);
}

public function update(Request $request, $id)
{
    Log::channel('stderr')->info($request);
    $meetup = $this->futureMeetupService->update($request, $id);

    return redirect('/meetups');
}
}

<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Inertia\Inertia;
use Inertia\Response;

class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    public function edit(Request $request): Response
    {
        return Inertia::render('Profile/Edit', [
            'mustVerifyEmail' => $request->user() instanceof MustVerifyEmail,
            'status' => session('status'),
        ]);
    }

    /**

```

```

        * Update the user's profile information.
    */
public function update(ProfileUpdateRequest $request): RedirectResponse
{
    $request->user()->fill($request->validated());

    if ($request->user()->isDirty('email')) {
        $request->user()->email_verified_at = null;
    }

    $request->user()->save();

    return Redirect::route('profile.edit');
}

/**
 * Delete the user's account.
 */
public function destroy(Request $request): RedirectResponse
{
    $request->validate([
        'password' => ['required', 'current_password'],
    ]);

    $user = $request->user();

    Auth::logout();

    $user->delete();

    $request->session()->invalidate();
    $request->session()->regenerateToken();

    return Redirect::to('/');
}
}

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class FutureMeetupRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array

```

```

    {
        return [
            //
        ];
    }
}

<?php

namespace App\Http\Requests;

use App\Models\User;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

class ProfileUpdateRequest extends FormRequest
{
    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\Rule|array|string>
     */
    public function rules(): array
    {
        return [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255',
Rule::unique(User::class)->ignore($this->user()->id)],
        ];
    }
}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class FutureMeetup extends Model
{
    use HasFactory;

    public $table = 'future_meetup';

    protected $fillable = [
        'future_meetup_title',
        'future_meetup_date',
        'future_meetup_description',
        'user_id',
    ];
}

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;

```

```

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}

<?php

namespace App\Repositories;

use App\Models\FutureMeetup;
use App\Repositories\Interfaces\FutureMeetupRepositoryInterface;

class FutureMeetupRepository implements FutureMeetupRepositoryInterface
{
    public function all()
    {
        return FutureMeetup::all();
    }

    public function create(object $data)
    {
        return FutureMeetup::create([
            'future_meetup_title' => $data->input('title'),

```

```

        'future_meetup_date' => $data->input('date'),
        'future_meetup_description' => $data->input('description'),
        'user_id' => $data->input('user_id'),
    ]);
}

public function update(object $data, $id)
{
    $futureMeetup = FutureMeetup::findOrFail($id);
    $futureMeetup->update([
        'future_meetup_title' => $data->input('title'),
        'future_meetup_date' => $data->input('date'),
        'future_meetup_description' => $data->input('description'),
        'user_id' => $data->input('user_id'),
    ]);

    return $futureMeetup;
}

public function delete($id)
{
    $futureMeetup = FutureMeetup::findOrFail($id);
    $futureMeetup->delete();
}

public function find($id)
{
    return FutureMeetup::findOrFail($id);
}
}

<?php

namespace App\Repositories\Interfaces;

interface FutureMeetupRepositoryInterface
{
    public function all();

    public function create(object $data);

    public function update(object $data, $id);

    public function delete($id);

    public function find($id);
}

```

**ПРИЛОЖЕНИЕ Б**  
**(обязательное)**  
**Исходный код клиентской части**

```
export function handleToggleAudio(stream) {
    try {
        console.log(stream);
        const audioTrack = stream
            .getTracks()
            .find((track) => track.kind === "audio");
        audioTrack.enabled = !audioTrack.enabled;
    } catch (err){
        console.log(err);
    }
};

export function handleToggleRemoteAudio (e, socket) {
    try {
        if (e.target.getAttribute("data-audio") === "hide") {
            e.target.setAttribute("data-audio", "show");
            socket.emit(
                "hide remote audio",
                e.target.getAttribute("data-id")
            );
        } else {
            e.target.setAttribute("data-audio", "hide");
            socket.emit(
                "show remote audio",
                e.target.getAttribute("data-id")
            );
        }
    } catch (err){
        console.log(err);
    }
};

export function hideAudio(stream) {
    const videoTrack = stream
        .getTracks()
        .find((track) => track.kind === "audio");
    videoTrack.enabled = false;
}

export function showAudio(stream) {
    const videoTrack = stream
        .getTracks()
        .find((track) => track.kind === "audio");
    videoTrack.enabled = true;
}

import Peer from "simple-peer";

export function createPeer(userToSignal, callerID, stream, socket) {
    const peer = new Peer({
        initiator: true,
        trickle: false,
        stream,
```

```

    });
    peer.on("signal", (signal) => {
        socket.emit("sending signal", {
            userToSignal,
            callerID,
            signal,
        });
        console.log("sending signal");
        console.log(userToSignal);
    });
    return peer;
}

export function addPeer(incomingSignal, callerID, stream, socket) {
    console.log(callerID);
    const peer = new Peer({
        initiator: false,
        trickle: false,
        stream,
    });

    peer.on("signal", (signal) => {
        console.log('returning signal');
        console.log(callerID);
        socket.emit("returning signal", { signal, callerID });
    });
    console.log('incoming signal');
    console.log(incomingSignal);
    setTimeout(() => peer.signal(incomingSignal), 3000);
    return peer;
}

export function showShareScreen(userstream, peers, screenTrack) {
    navigator.mediaDevices
        .getDisplayMedia({ cursor: true, audio: true })
        .then((stream) => {
            const videoTrack = userstream
                .getTracks()
                .find((track) => track.kind === "video");
            screenTrack.current = stream.getTracks()[0];
            userstream.removeTrack(videoTrack);
            userstream.addTrack(screenTrack.current);
            peers.map((peer) => {
                console.log(peer);
                peer.peer.replaceTrack(
                    videoTrack,
                    screenTrack.current,
                    userstream
                );
            });
        });
}

export function hideShareScreen(userstream, peers, screenTrack) {
    navigator.mediaDevices
        .getUserMedia({ video: true, audio: true })
        .then((stream) => {
            console.log("videotrack");
            console.log(stream.getTracks());

```

```

        const videoTrack = stream
            .getTracks()
            .find((track) => track.kind === "video");
        userstream.removeTrack(screenTrack.current);
        userstream.addTrack(videoTrack);
        peers.map((peer) => {
            console.log(peer);
            peer.peer.replaceTrack(
                screenTrack.current,
                videoTrack,
                userstream
            );
        });
    });

export function copyURL() {
    const el = document.createElement("input");
    el.value = window.location.href;
    document.body.appendChild(el);
    el.select();
    document.execCommand("copy");
    document.body.removeChild(el);
}

export function handleToggleVideo(stream){
    try {
        console.log(stream);
        const videoTrack = stream
            .getTracks()
            .find((track) => track.kind === "video");
        videoTrack.enabled = !videoTrack.enabled;
    } catch(err){
        console.log(err)
    }
};

export function handleToggleRemoteVideo (e, socket){
    try{
        console.log('remote video');
        if (e.target.getAttribute("data-video") === "hide") {
            e.target.setAttribute("data-video", "show");
            socket.emit(
                "hide remote cam",
                e.target.getAttribute("data-id")
            );
        } else {
            e.target.setAttribute("data-video", "hide");
            socket.emit(
                "show remote cam",
                e.target.getAttribute("data-id")
            );
        }
    } catch {
        console.log(err);
    }
};


```

```

export function hideVideo(stream) {
    console.log(stream);
    console.log("hide");
    const videoTrack = stream
        .getTracks()
        .find((track) => track.kind === "video");
    videoTrack.enabled = false;
}

export function showVideo(stream) {
    const videoTrack = stream
        .getTracks()
        .find((track) => track.kind === "video");
    videoTrack.enabled = true;
}

import CustomButton from "@/Components/Button/CustomButton";
import MeetupList from "../MeetupList/MeetupList";
import { Link } from "@inertiajs/react";
import img from "../../../../../assets/logo.png";
import AddMeetupForm from "@/Components/Form/AddMeetupForm";
import { useState } from "react";
import Header from "@/Layouts/Header";

export default function MeetupMenu({ meetups, auth }) {
    const [open, setOpen] = useState(false);
    return (
        <>
            <div className="flex flex-col justify-center">
                <Header auth={auth} />
                <div>
                    <div className="flex mt-10 mx-auto w-[50%] justify-between">
                        <div className="group cursor-pointer flex gap-6 w-[450px] h-[150px] p overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#00FF7F] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]">
                            <CustomButton
                                title="Открыть календарь"
                                pin="fa-regular fa-calendar-days"
                                background="bg-[#00FF7F]"
                            />
                        </div>
                        <div
                            onClick={() => setOpen(true)}
                            className="group flex cursor-pointer gap-6 w-[450px] h-[150px] p overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#F4A460] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]">
                            <CustomButton
                                title="Добавить встречу"
                                pin="fa-solid fa-calendar-plus"
                                background="bg-[#F4A460]"
                            />
                        </div>
                    </div>
                </div>
            </div>
        </>
    )
}

```

```

        <div className="mt-20 w-[65%] mx-auto my-0">
            {meetups.length ? (
                <>
                    <p className="text-center text-5xl text-gray-400">
                        Запланированные звонки
                    </p>
                    <MeetupList meetups={meetups} />
                </>
            ) : (
                <p className="text-center text-6xl text-gray-400">
                    Запланированных звонков нет
                </p>
            )
        )
    </div>
</div>
<AddMeetupForm open={open} setOpen={setOpen} id={auth.user.id} />
</>
);
}

import MeetupCard from "../MeetupCard/MeetupCard";
import { useEffect } from "react";

export default function MeetupList(props) {
    useEffect(() => {
        console.log(props.meetups);
    }, []);

    return (
        <>
            <div className="grid grid-cols-2 gap-y-[20px] gap-x-[70px] mt-[30px]">
                {props.meetups.map((meetup) => {
                    return <MeetupCard key={meetup.title} data={meetup} />;
                })}
            </div>
            ;
        </>
    );
}

import CustomButton from "@/Components/Button/CustomButton";
import MeetupList from "../MeetupList/MeetupList";
import { Link } from "@inertiajs/react";
import img from "../../../../../assets/logo.png";
import AddMeetupForm from "@/Components/Form/AddMeetupForm";
import { useState } from "react";
import Header from "@/Layouts/Header";

export default function MeetupMenu({ meetups, auth }) {
    const [open, setOpen] = useState(false);
    return (
        <>
            <div className="flex flex-col justify-center">
                <Header auth={auth} />
                <div>
                    <div className="flex mt-10 mx-auto w-[50%] justify-between">

```

```

        <div className=" group cursor-pointer flex gap-6 w-[450px] h-[150px] p overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#00FF7F] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]">
            <CustomButton
                title="Открыть календарь"
                pin="fa-regular fa-calendar-days"
                background="bg-[#00FF7F]"
            />
        </div>
        <div
            onClick={() => setOpen(true)}
            className=" group flex cursor-pointer gap-6 w-[450px] h-[150px] p overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#F4A460] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]">
            <CustomButton
                title="Добавить встречу"
                pin="fa-solid fa-calendar-plus"
                background="bg-[#F4A460]"
            />
        </div>
    </div>
    <div className="mt-20 w-[65%] mx-auto my-0">
        {meetups.length ? (
            <>
                <p className="text-center text-5xl text-gray-400">
                    Запланированные звонки
                </p>
                <MeetupList meetups={meetups} />
            </>
        ) : (
            <p className="text-center text-6xl text-gray-400">
                Запланированных звонков нет
            </p>
        )}
    </div>
</div>
<AddMeetupForm open={open} setOpen={setOpen} id={auth.user.id} />
</>
);
}

import { Link } from "@inertiajs/react";
import { v1 as uuid } from "uuid";
import Card from "../../Components/Card/Card";
import { useState } from "react";
import RoomEnterForm from "../../Components/Form/RoomEnterForm";

export default function AuthenticatedMenu() {
    const [open, setOpen] = useState(false);
    return (
        <>

```

```

        <div className="grid gap-6 lg:grid-cols-3 lg:gap-8">
          <Link
            href={`/room/${uuid()}`}
            id="docs-card"
            className="flex flex-col group items-start gap-6 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#14C7C3] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] lg:pb-32 lg:pt-32 dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
          >
            <Card
              title="Начать встречу"
              pin="fa-solid fa-video"
              background="bg-[#14C7C3]"
            />
          </Link>

          <div
            onClick={() => setOpen(true)}
            className="flex flex-col cursor-pointer group items-start gap-6 lg:pb-32 lg:pt-32 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#FA650A] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
          >
            <Card
              title="Подключиться"
              pin="fa-solid fa-plus"
              background="bg-[#FA650A]"
            />
          </div>

          <Link
            href={`/meetups`}
            className="flex flex-col items-start group gap-6 lg:pb-32 lg:pt-32 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
          >
            <Card
              title="Запланировать встречу"
              pin="fa-solid fa-calendar"
              background="bg-[#FF216A]"
            />
          </Link>
        </div>
        <RoomEnterForm open={open} setOpen={setOpen} />
      </>
    );
  };

import { Link } from "@inertiajs/react";
import Card from "../../Components/Card/Card";
import { useState } from "react";
import RoomEnterForm from "../../Components/Form/RoomEnterForm";

```

```

export default function GuestMenu() {
  const [open, setOpen] = useState(false);
  return (
    <>
      <div className="grid gap-6 lg:grid-cols-3 lg:gap-8">
        <Link
          href={route("login")}
          id="docs-card"
          className="flex flex-col group items-start gap-6 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#14C7C3] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] lg:pb-32 lg:pt-32 dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
        >
          <Card
            title="Войти"
            pin="fa-solid fa-user"
            background="bg-[#14C7C3]"
          />
        </Link>

        <Link
          href={route("register")}
          className="flex flex-col group items-start gap-6 lg:pb-32 lg:pt-32 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:bg-[#FA650A] hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
        >
          <Card
            title="Зарегистрироваться"
            pin="fa-solid fa-user-plus"
            background="bg-[#FA650A]"
          />
        </Link>

        <div
          onClick={() => setOpen(true)}
          className="flex cursor-pointer flex-col items-start group gap-6 lg:pb-32 lg:pt-32 overflow-hidden rounded-lg bg-white p-6 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
        >
          <Card
            title="Подключиться"
            pin="fa-solid fa-plus"
            background="bg-[#FF216A]"
          />
        </div>
      </div>
      <RoomEnterForm open={open} setOpen={setOpen} />
    </>
  );
}

import React, { useEffect, useRef, useState } from "react";

```

```

import io from "socket.io-client";
import { Link } from "@inertiajs/react";
import VideoButtonCard from "@/Components/Card/VideoButtonCard";
import {
    handleToggleVideo,
    handleToggleRemoteVideo,
    hideVideo,
    showVideo,
} from "@/Services/VideoService";
import {
    handleToggleAudio,
    handleToggleRemoteAudio,
    hideAudio,
    showAudio,
} from "@/Services/AudioService";
import { copyURL } from "@/Services/URLService";
import { addPeer, createPeer } from "@/Services/PeerService";
import {
    showShareScreen,
    hideShareScreen,
} from "@/Services/ShareScreenService";
import Canvas from "../../Components/Canvas/Canvas";
import Video from "../../Components/Video/Video";

const getLastItem = (path) => path.substring(path.lastIndexOf("/") + 1);

export default function Room({ auth }) {
    const [peers, setPeers] = useState([]);
    const socketRef = useRef();
    const userVideo = useRef();
    const peersRef = useRef([]);
    const userStream = useRef();
    const [copied, setCopied] = useState(false);
    const [isVideo, setIsVideo] = useState(false);
    const [isAudio, setIsAudio] = useState(false);
    const [isSharing, setIsSharing] = useState(false);
    const [isCanvas, setIsCanvas] = useState(false);
    const isAdmin = useRef(false);
    const screenTrack = useRef();
    const hideButtons = useRef([]);
    const roomID = getLastItem(window.location.href);

    useEffect(() => {
        socketRef.current = io.connect(":8000");
        console.log(roomID);
        console.log(peers);
        console.log(navigator.mediaDevices);
        navigator.mediaDevices
            .getUserMedia({ video: true, audio: true })
            .then((stream) => {
                userStream.current = stream;
                userVideo.current.srcObject = stream;

                socketRef.current.emit("join room", roomID);

                socketRef.current.on("all users", (users) => {
                    console.log(`users: ${users}`);
                    if (users.length === 0) {
                        isAdmin.current = true;
                    }
                })
            })
    })
}

```

```

        const peers = [];
        users.forEach(userID) => {
            const peer = createPeer(
                userID,
                socketRef.current.id,
                stream,
                socketRef.current
            );
            peersRef.current.push({
                peerID: userID,
                peer,
            });
            peers.push({ peerID: userID, peer });
        });
        setPeers(peers);
    });

    socketRef.current.on("user joined", (payload) => {
        if (isAdmin.current) {
            console.log("i am admin");
            hideButtons.current.push(
                <>
                    <div className="flex bg-black w-[200px] h-[25px] rounded-lg shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] absolute top-[5px] left-[40%] z-50 ">
                        <button
                            onClick={(e) =>
                                handleToggleRemoteVideo(
                                    e,
                                    socketRef.current
                                )
                            }
                            data-id={payload.callerID}
                            data-video="hide"
                            className="flex bg-black items-center justify-center w-[50%] h-full pl-4 pr-4 border-r-2 border-white z-1000 text-white hover:bg-gray-500"
                        >
                            <i className="fa-solid fa-video text-white text-sm text-center"></i>
                        </button>
                        <button
                            onClick={(e) =>
                                handleToggleRemoteAudio(
                                    e,
                                    socketRef.current
                                )
                            }
                            data-id={payload.callerID}
                            data-audio="hide"
                            className="flex bg-black items-center justify-center w-[50%] h-full pl-4 pr-4 z-1000 text-white hover:bg-gray-500"
                        >
                            <i className="fa-solid fa-microphone text-white text-sm text-center"></i>
                        </button>
                    </div>
                </>
            );
            console.log("user joined");
        }
    });
}

```

```

        console.log(peers);
        const peer = addPeer(
            payload.signal,
            payload.callerID,
            stream,
            socketRef.current
        );
        peersRef.current.push({
            peer,
            peerID: payload.callerID,
        });
        const peerObj = { peer, peerID: payload.callerID };
        setPeers((users) => [...users, peerObj]);
    }
});

socketRef.current.on("receiving returned signal", (payload) => {
    const item = peersRef.current.find(
        (p) => p.peerID === payload.id
    );
    console.log("receiving returned signal");
    console.log(payload.signal);
    item.peer.signal(payload.signal);
});

socketRef.current.on(
    "hide cam",
    hideVideo.bind(null, userStream.current)
);
socketRef.current.on(
    "show cam",
    showVideo.bind(null, userStream.current)
);
socketRef.current.on(
    "hide audio",
    hideAudio.bind(null, userStream.current)
);
socketRef.current.on(
    "show audio",
    showAudio.bind(null, userStream.current)
);

socketRef.current.on("canvas", () => {
    setIsCanvas(!isCanvas);
});

socketRef.current.on("user left", (id) => {
    const peerObj = peersRef.current.find(
        (p) => p.peerID === id
    );
    if (peerObj) {
        peerObj.peer.destroy();
    }
    const peers = peersRef.current.filter(
        (p) => p.peerID !== id
    );
    peersRef.current = peers;
    setPeers(peers);
});
})

```

```

        .catch((error) => console.log(error));
    }, []);
```

```

const handleToggleShareScreen = (userstream) => {
    !isSharing
        ? showShareScreen(userstream, peers, screenTrack)
        : hideShareScreen(userstream, peers, screenTrack);
};
```

```

return (
    <>
        <div class="h-full">
            <div className="py-12 bg-gray-900">
                <div className="flex flex-col h-full w-full justify-center items-center">
                    <div
                        className={
                            !isCanvas
                                ? "flex w-[95%] h-[750px] overflow-y mx-auto pt-10 overflow-auto"
                                : "flex w-full h-[750px] overflow-y pt-10 overflow-auto"
                        }
                    >
                        {isCanvas && <Canvas socket={socketRef.current} />}
                        <div
                            className={
                                !isCanvas
                                    ? "grid grid-cols-2 gap-4"
                                    : "grid grid-cols-1"
                            }
                        >
                            <video
                                className={
                                    !isCanvas
                                        ? "w-[100%] h-[650px]"
                                        : "w-[300px] h-[100px]"
                                }
                                controls
                                playsInline
                                ref={userVideo}
                                autoPlay
                            />
                            {peers.map((peer, index) => {
                                return (
                                    <>
                                        <div className="relative">
                                            <Video
                                                isCanvas={isCanvas}
                                                key={peer.peerID}
                                                peer={peer.peer}
                                            />
                                            {!isCanvas &&
                                                hideButtons.current[index]}
                                        </div>
                                    </>
                                );
                            })}
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </>
)
```

```

        <div className="flex mt-10 justify-center">
          <div
            onClick={() => {
              handleToggleVideo(userStream.current);
              setIsVideo(!isVideo);
            }}
            className="flex mr-4 w-[75px] h-[75px] items-center justify-center group gap-6 overflow-hidden rounded-lg bg-gray-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 "
          >
            {isVideo ? (
              <i className="fa-solid fa-video text-xl text-white"></i>
            ) : (
              <i className="fa-solid fa-video-slash text-xl text-white"></i>
            )}
          </div>
          <div
            onClick={() => {
              handleToggleAudio(userStream.current);
              setIsAudio(!isAudio);
            }}
            className="flex mr-4 w-[75px] h-[75px] items-center justify-center group gap-6 overflow-hidden rounded-lg bg-gray-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
          >
            {isAudio ? (
              <i className="fa-solid fa-microphone text-xl text-white"></i>
            ) : (
              <i className="fa-solid fa-microphone-slash text-xl text-white"></i>
            )}
          </div>
          <Link
            href="/"
            className="flex mr-4 w-[75px] h-[75px] items-center justify-center group gap-6 overflow-hidden rounded-lg bg-red-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
            onClick={() => {
              console.log("disconnect");
              socketRef.current.emit("exit");
            }}
          >
            <i className="fa-solid fa-phone text-xl text-white"></i>
          </Link>
          <div
            onClick={() => {
              handleToggleShareScreen(userStream.current);
              setIsSharing(!isSharing);
            }}

```

```

        }
        className="flex mr-4 w-[75px] h-[75px] items-center justify-center group gap-6 overflow-hidden rounded-lg bg-gray-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
      >
        <i className="fa-solid fa-desktop text-white"></i>
      </div>
      <div
        className="flex mr-4 w-[75px] h-[75px] items-center justify-center group gap-6 overflow-hidden rounded-lg bg-gray-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 hover:bg-[#FF216A] focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20]"
        onClick={() => {
          setCopied(true);
          copyURL();
        }}
      >
        <i className="fa-solid fa-link text-xl text-white"></i>
      </div>
      <div
        className="flex mr-4 cursor-pointer w-[75px] h-[75px] items-center justify-center group gap-6 border-1 rounded-[9px] bg-gray-500 shadow-[0px_14px_34px_0px_rgba(0,0,0,0.08)] hover:bg-[#FF216A] ring-1 ring-white/[0.05] transition duration-300 hover:ring-black/40 focus:outline-none focus-visible:ring-[#FF2D20] dark:bg-zinc-900 dark:ring-zinc-800 dark:hover:text-white/70 dark:hover:ring-zinc-700 dark:focus-visible:ring-[#FF2D20] "
        onClick={() => {
          console.log(isCanvas);
          setIsCanvas(!isCanvas);
          socketRef.current.emit("start canvas");
        }}
      >
        <i className="fa-solid fa-paintbrush text-xl text-white"></i>
      </div>
    </div>
  </div>
</>
);
}

import AuthenticatedLayout from "@/Layouts/AuthenticatedLayout";
import DeleteUserForm from "./Partials/DeleteUserForm";
import UpdatePasswordForm from "./Partials/UpdatePasswordForm";
import UpdateProfileInformationForm from "./Partials/UpdateProfileInformationForm";
import { Head, Link } from "@inertiajs/react";
import img from "../../assets/logo.png";

export default function Edit({ auth, mustVerifyEmail, status }) {

```

```

        return (
            <AuthenticatedLayout
                user={auth.user}
                header={
                    <h2 className="font-semibold text-xl text-gray-800 leading-tight">
                        Profile
                    </h2>
                }
            >
                <Head title="Profile" />

                <div className="py-12">
                    <div className="max-w-7xl mx-auto sm:px-6 lg:px-8 space-y-6">
                        <div className="p-4 sm:p-8 bg-white shadow sm:rounded-lg">
                            <UpdateProfileInformationForm
                                mustVerifyEmail={mustVerifyEmail}
                                status={status}
                                className="max-w-xl"
                            />
                        </div>

                        <div className="p-4 sm:p-8 bg-white shadow sm:rounded-lg">
                            <UpdatePasswordForm className="max-w-xl" />
                        </div>

                        <div className="p-4 sm:p-8 bg-white shadow sm:rounded-lg">
                            <DeleteUserForm className="max-w-xl" />
                        </div>
                    </div>
                </AuthenticatedLayout>
            );
        }
    }

    import { Link, Head } from "@inertiajs/react";
    import img from "../../assets/logo.png";
    import GuestMenu from "../Pages/Menu/GuestMenu/GuestMenu";
    import AuthenticatedMenu from "../Pages/Menu/AuthMenu/AuthMenu";

    export default function Main(auth) {
        return (
            <>
                <Head title="Welcome" />
                <div className="text-black/50 dark:bg-black dark:text-white/50">
                    <div className="relative flex flex-col items-center justify-center selection:bg-[#FF2D20] selection:text-white">
                        <div className="relative w-full max-w-2xl px-6 lg:max-w-7xl">
                            <main className="mt-12">
                                {auth.auth.auth ? (
                                    <AuthenticatedMenu></AuthenticatedMenu>
                                ) : (
                                    <GuestMenu></GuestMenu>
                                )}
                            </main>
                            <footer className="py-16 text-center text-sm text-black dark:text-white/70"></footer>
                        </div>
                    </div>
                </div>
            </>
        );
    }
}

```

```

        </div>
        <script src="../path/to/flowbite/dist/flowbite.min.js"></script>
    </>
);
}

import ApplicationLogo from "@Components/AppLogo/ApplicationLogo";
import { Link } from "@inertiajs/react";
import img from "../assets/logo.png";

export default function Guest({ children }) {
    return (
        <div className="min-h-screen flex flex-col sm:justify-center items-center pt-6 sm:pt-0">
            <div>
                <Link href="/">
                    <img
                        className="h-32 w-32 text-white lg:text-[#FF2D20]"
                        viewBox="0 0 62 65"
                        src={img}
                    ></img>
                </Link>
            </div>

            <div className="w-full sm:max-w-md mt-6 px-6 py-4 bg-white shadow-md overflow-hidden sm:rounded-lg">
                {children}
            </div>
        </div>
    );
}

import Header from "./Header";
import Main from "./Main";
import Footer from "./Footer";

export default function UserLayout(auth) {
    return (
        <>
            <Header auth={auth}></Header>
            <Main auth={auth}></Main>
            <Footer></Footer>
        </>
    );
}

```

**ПРИЛОЖЕНИЕ В**  
**(обязательное)**  
**Исходный код веб-сокет сервера**

```
require('dotenv').config();
const express = require("express");
const http = require("http");
const app = express();
const server = http.createServer(app);
const socket = require("socket.io");
const cors = require('cors');

const users = {};
const socketToRoom = {};
const corsOptions = {
  origin: '*',
  optionsSuccessStatus: 200,
};

app.use(cors(corsOptions));

const io = require("socket.io")(server, {
  cors: {
    origin: "*",
    methods: ["GET", "POST"]
  }
});

io.on('connection', socket => {
  socket.on("join room", roomID => {
    console.log('join room');
    console.log(roomID);
    if (users[roomID]) {
      const length = users[roomID].length;
      if (length === 4) {
        socket.emit("room full");
        return;
      }
      users[roomID].push(socket.id);

    } else {
      users[roomID] = [socket.id];
    }
    socketToRoom[socket.id] = roomID;
    const usersInThisRoom = users[roomID].filter(id => id !== socket.id);
    console.log(socket.id);
    console.log(usersInThisRoom)
    socket.emit("all users", usersInThisRoom);
  });
}

socket.on("sending signal", payload => {
  console.log('sending signal');
  console.log(payload.callerID);
```

```

        io.to(payload.userToSignal).emit('user joined', { signal: payload.signal,
callerID: payload.callerID })
        console.log("user joined");
    });

socket.on("returning signal", payload => {
    console.log('returning signal');
    console.log(payload.callerID);
    io.to(payload.callerID).emit('receiving returned signal', { signal: payload.signal, id: socket.id })
});

socket.on('disconnect', () => {
    const roomID = socketToRoom[socket.id];
    let room = users[roomID];
    if (room) {
        room = room.filter(id => id !== socket.id);
        users[roomID] = room;
    }
});

socket.on('hide remote cam', targetId => {
    console.log(targetId);
    io.to(targetId).emit('hide cam');
});

socket.on('show remote cam', targetId => {
    console.log(targetId);
    io.to(targetId).emit('show cam');
});

socket.on('hide remote audio', targetId => {
    console.log(targetId);
    io.to(targetId).emit('hide audio');
});

socket.on('show remote audio', targetId => {
    console.log(targetId);
    io.to(targetId).emit('show audio');
});

socket.on('disconnect', () => {
    console.log('disconnect')
    const roomId = socketToRoom[socket.id];
    let room = users[roomId];
    if (room) {
        room = room.filter(id => id !== socket.id);
        users[roomId] = room;
    }
    socket.broadcast.emit('user left', socket.id);
});

socket.on('exit', () => {
    console.log('disconnect')
    const roomId = socketToRoom[socket.id];
    let room = users[roomId];
    if (room) {
        room = room.filter(id => id !== socket.id);
        users[roomId] = room;
    }
});

```

```
        }
        socket.broadcast.emit('user left', socket.id);
    });

    socket.on('start canvas', () => {
        console.log('canvas');
        socket.broadcast.emit('canvas');
    })

    socket.on('canvasImage', data => {
        socket.broadcast.emit('canvasImage', data)
    });
}

server.listen(8000, () => console.log('server is running on port 8000'));
```

