МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного обеспечения для реализации системы видеозвонков и голосовых сообщений в мобильном приложении социальной сети»

Обучающийся А.А. Крюков
(Инициалы Фамилия) (личная подпись)

Руководитель канд. тех. наук, доцент, О. В. Аникина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. фил. наук, доцент, М. В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: «Разработка программного обеспечения для реализации системы видеозвонков и голосовых сообщений в мобильном приложении социальной сети».

Работа выполнена на 49 страницах, содержит 24 иллюстрации и 1 таблицу. Структура работы включает введение, три главы, заключение, список использованных источников и приложение.

Тематика исследования связана с созданием мобильного приложения социальной направленности, предназначенного для обмена голосовыми сообщениями и совершения видеозвонков. В работе обоснована актуальность разработки в условиях роста популярности мобильных коммуникационных решений.

Целью работы является разработка и реализация функционального мобильного приложения, обеспечивающего удобный и безопасный способ голосового общения и видеосвязи между пользователями.

В процессе работы были проанализированы существующие решения, определены требования к функционалу, разработан пользовательский интерфейс и реализованы основные функции с использованием современных технологий, таких как Zego Cloud SDK и socket.io.

В результате работы было создано и протестировано мобильное приложение, демонстрирующее корректную работу всех ключевых функций, включая обмен голосовыми сообщениями, а также установление видеосоединения между пользователями.

Abstract

The topic of the graduation thesis: “Development of software for implementing a video call and voice messaging system in a mobile social networking application.”

The thesis consists of 50 pages and includes 24 illustrations and 1 table. The structure of the work includes an introduction, three chapters, a conclusion, a list of references, and an appendix.

The subject of the research is related to the creation of a socially oriented mobile application designed for exchanging voice messages and making video calls. The relevance of the development is substantiated in the context of the growing popularity of mobile communication solutions.

The goal of the work is to develop and implement a functional mobile application that provides a convenient and secure method of voice and video communication between users.

During the course of the work, existing solutions were analyzed, functional requirements were defined, the user interface was designed, and core features were implemented using modern technologies such as Zego Cloud SDK and socket.io.

As a result of the project, a mobile application was created and tested, demonstrating the correct operation of all key features, including voice message exchange and establishing video connections between users.

Оглавление

Введение 5
Глава 1 Анализ предметной области 8
1.1 Описание компании 8
1.2 Выбор case–средств для описания бизнес–процессов 11
Глава 2 Описание задач и этапов разработки 14
2.1 Требования к персоналу, обязанности, требования к рабочему месту 14
2.2 Описание поставленной задачи 16
2.3 Выбор и обоснование методов решения задачи 18
2.4 Проектирование системы 19
2.5 Структура и компоненты системы 25
Глава 3 Разработка алгоритма, программирование и откладка 27
3.1 Описания средств для решения поставленных задач 27
3.2 Задачи решаемые в процессе разработки программного обеспечения 28
3.3 Выполнение задания 29
3.4 Тестирование 44
Заключение 46
Список используемой литературы 48
Приложение А Легенда к блок-схеме последовательности обработки голосового сообщения 50

Введение

Современные мобильные приложения социальных сетей играют ключевую роль в жизни общества, предоставляя пользователям удобные инструменты для общения и обмена информацией. Одной из наиболее востребованных функций таких приложений являются видеозвонки и голосовые сообщения, обеспечивающие оперативную связь в режиме реального времени. Развитие технологий передачи данных, в частности WebRTC и облачных сервисов, позволило значительно улучшить качество видеосвязи и повысить стабильность соединений [21]. Однако внедрение подобных решений требует проработки архитектуры, выбора оптимальных инструментов и учета ограничений мобильных устройств.

Актуальность темы обусловлена необходимостью создания эффективного и удобного программного обеспечения для видеозвонков и голосовых сообщений, которое будет обеспечивать стабильную работу даже в условиях нестабильного интернет–соединения. Существующие решения, такие как Zoom, WhatsApp и Telegram, предлагают широкий функционал, однако требуют значительных ресурсов или не предоставляют гибкости для интеграции в сторонние приложения [9],[13]. Разработка специализированной системы для мобильных социальных сетей позволит предложить пользователям удобный инструмент взаимодействия с учетом их потребностей.

Объект исследования – программное обеспечение для передачи аудио– и видеоданных в режиме реального времени.

Предмет исследования – технологии и методы реализации видеозвонков и голосовых сообщений в мобильных приложениях социальной сети.

Целью работы является разработка программного обеспечения для видеозвонков и голосовых сообщений, обеспечивающего высокое качество связи, безопасность передаваемых данных и удобную интеграцию с мобильным приложением социальной сети.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить существующие технологии и методы реализации видеозвонков и голосовых сообщений;
- провести сравнительный анализ решений для передачи аудио– и видеоданных;
- разработать архитектуру системы с учетом клиент–серверного взаимодействия;
- реализовать механизм видеозвонков с использованием Zego Cloud SDK;
- разработать функционал голосовых сообщений на основе Flutter Sound;
- протестировать и оптимизировать приложение для обеспечения стабильной работы в различных сетевых условиях.

Методы исследования включают анализ существующих технологий передачи мультимедиа, проектирование клиент–серверной архитектуры, разработку и тестирование программного обеспечения с последующей оптимизацией.

Практическая значимость работы заключается в разработке решения, которое может быть интегрировано в мобильные социальные сети, корпоративные коммуникационные платформы и сервисы удаленного взаимодействия. Полученные результаты могут быть использованы для дальнейшего совершенствования технологий видеосвязи в мобильных приложениях.

В первой главе рассматривается деятельность компании, для которой разрабатывается программное обеспечение. Анализируется структура предприятия и его потребности в области мобильных коммуникаций [10]. Отдельное внимание уделяется выбору CASE-средств, с помощью которых проводится моделирование бизнес-процессов. На основе этого создаётся основа для постановки задачи и определения функциональных требований к будущей системе.

Во второй главе формулируются задачи, решаемые в рамках проекта, и требования к разработке. Подробно описывается поставленная задача, включая цели и ожидаемые результаты. Рассматриваются методы реализации, обосновывается выбор инструментов и технологий. Также проводится проектирование архитектуры приложения, определяются его структура и компоненты, описываются элементы клиент-серверного взаимодействия и логика системы.

В третьей главе описываются этапы непосредственной реализации программного продукта. Приводятся используемые программные средства, библиотеки и окружение разработки. Детально описывается процесс программирования, от написания кода до создания интерфейса. Показано, как реализованы функции видеозвонков и голосовых сообщений. Завершается глава разделом, посвящённым тестированию, где представлены результаты проверки работоспособности приложения, оценка стабильности и выводы о готовности к использованию.

Глава 1 Анализ предметной области

1.1 Описание компании

Сфера деятельности компании, на базе которой выполнялась выпускная квалификационная работа – разработка программного обеспечения и информационных технологий с последующей их интеграцией в электронной коммерции.

Программное обеспечение, разрабатываемое «ВорлдИнтерТех РУС», выступает в роли связующего звена между компаниями и потребителями, предоставляя самые различные сервисы, например, в сфере оптовых и розничных продаж, предоставления услуг, аудита и маркетинга.

Функциональное место в организации – разработчик программного обеспечения, в обязанности которого входит разработка ПО, на основе технического задания.

Подразделение компании – отдел разработки.

Этот отдел занимается составлением технического задания на основе пожеланий заказчика и разработкой программного обеспечения в соответствии с составленным ТЗ.

«ВорлдИнтерТех РУС» имеет организационную структуру, представленную на рисунке 1.

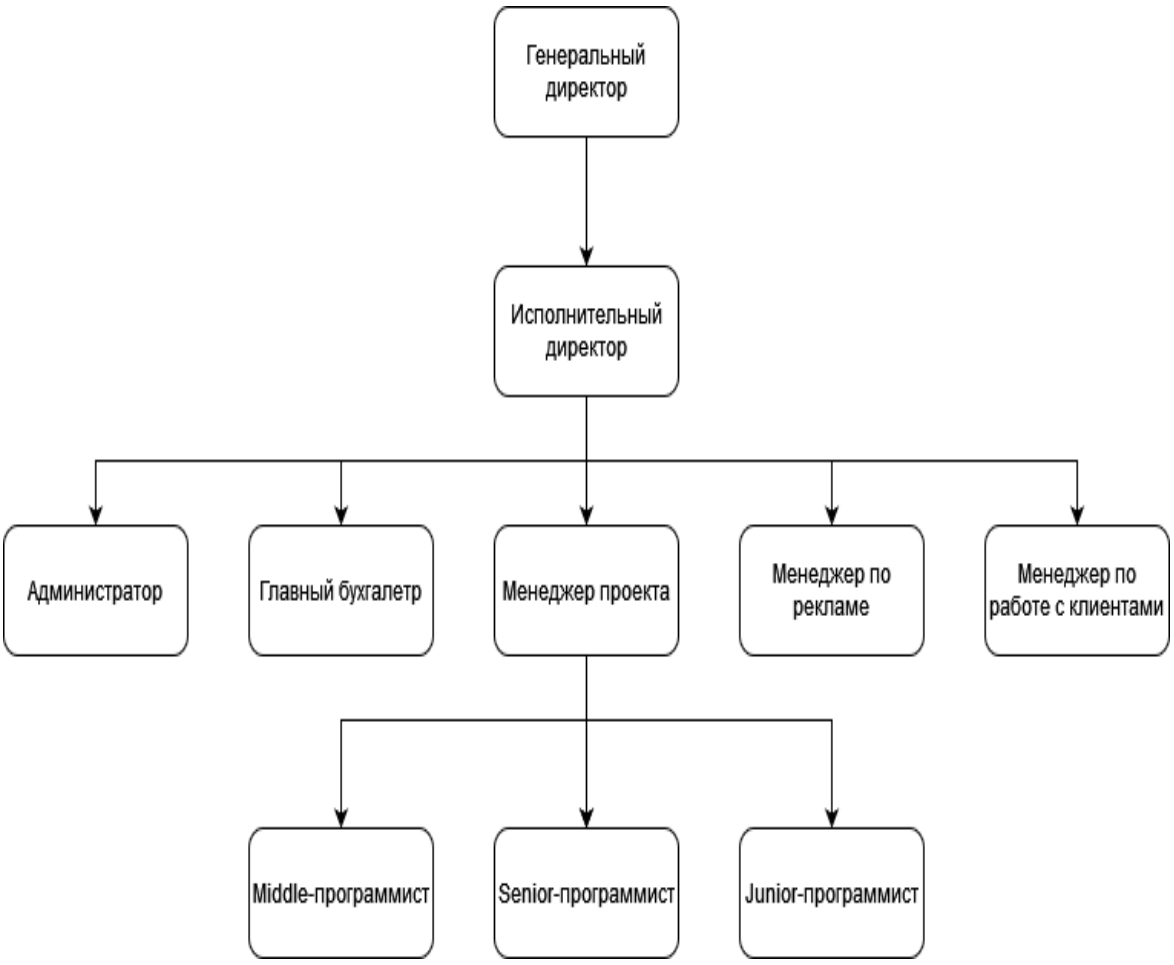


Рисунок 1 – Организационная структура «ВорлдИнтерТех РУС»

В отдел разработки входят следующие сотрудники:

- менеджер;
- senior–программист;
- middle–программист;
- junior–программист.

Рассмотрим их функции подробнее на рисунке 2.

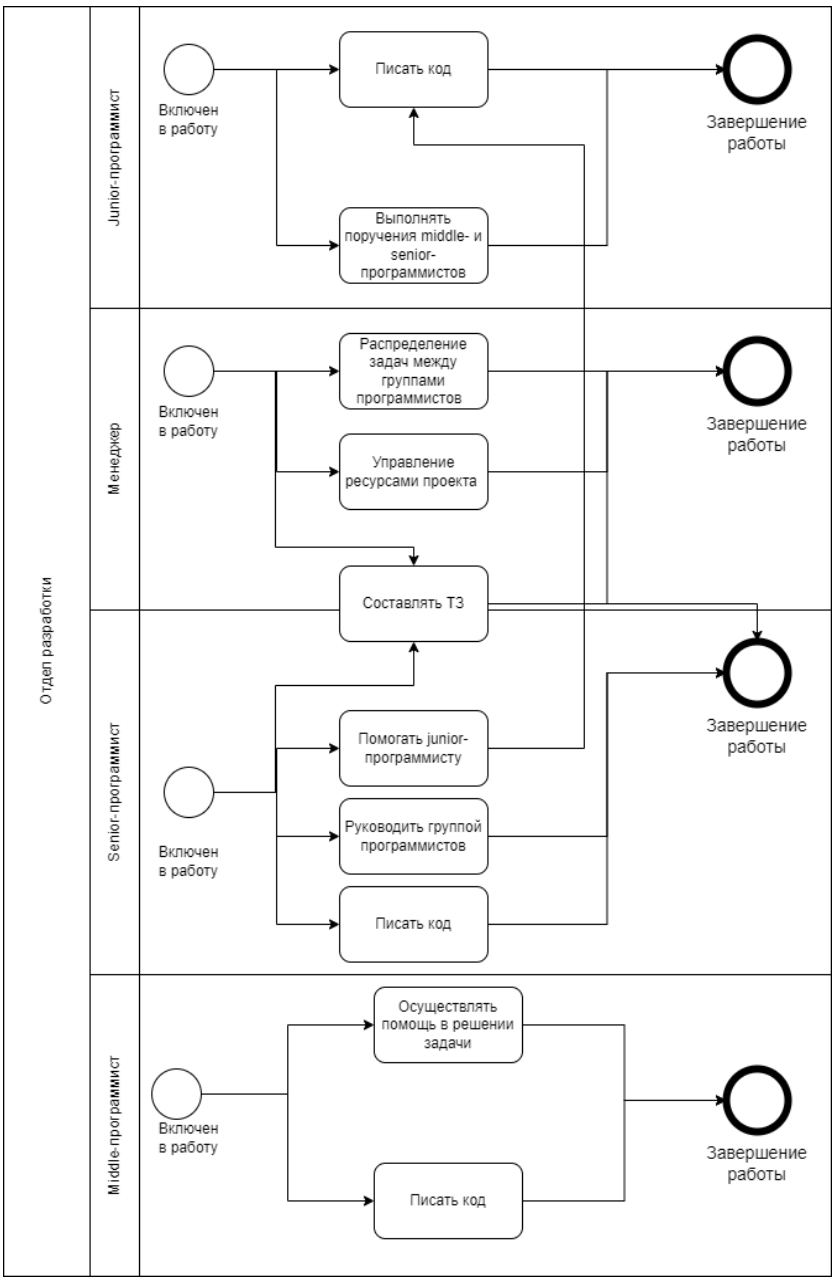


Рисунок 2 – Функции, выполняемые сотрудниками отдела разработок

Рассмотрим роли и функции сотрудников, которые представлены на рисунке 2.

Junior–программист. Его сфера деятельности сильно ограничена в связи с небольшим опытом работы и ограниченностью знаний в области разработки ПО. «Джун» выполняет небольшие задачи по сопровождению и улучшению уже разработанных сервисов. Junior–программист также может брать более сложные задания с целью повышения своей квалификации. Как правило,

«Джуны» сильно зависят от опытных программистов в силу своей неопытности.

Middle–программист. Это полностью самостоятельный программист, имеющий большую свободу действий, что характеризуется меньшим контролем действий в процессе решения задачи. Также «Мидл» курирует более младших программистов и помогают им в карьерном росте.

Senior–программист. Программист с большим опытом разработки, который лучше всех ориентируется в предметной области. Занимается исправлением сложных багов и разработкой новых сервисов. Выполняет проверку кода младших программистов.

Менеджер. Менеджер управляет ресурсами проекта, совместно с Senior–программистом устанавливает сроки выполнения тех или иных задач и составляют ТЗ, общаясь с заказчиком.

1.2 Выбор case–средств для описания бизнес–процессов

На этапе проектирования программного обеспечения важную роль играет использование CASE–средств (Computer Aided Software Engineering) – инструментов, предназначенных для автоматизации процесса анализа, моделирования и документирования информационных систем. Они позволяют разработчику наглядно представить архитектуру приложения, взаимодействие компонентов и логику пользовательского взаимодействия [1].

Существует несколько методов описания структуры и поведения системы:

- текстовый, при котором логика работы описывается словами;
- табличный, где используются структурированные таблицы;
- графический, при котором строятся диаграммы и схемы.

Наиболее эффективным в контексте проектирования мобильного приложения является графический способ, так как он обеспечивает

наглядность, упрощает анализ и позволяет визуально отразить логику взаимодействия элементов.

В ходе разработки были рассмотрены следующие CASE–средства:

- staruml – современный инструмент, ориентированный на моделирование с использованием UML–диаграмм. Поддерживает широкий набор диаграмм, включая диаграммы классов, прецедентов, активности и последовательностей. Имеет удобный интерфейс и расширяемую архитектуру за счёт плагинов;
- ramus 2.0 – средство для построения диаграмм IDEF0 и DFD, позволяющее структурировать функции системы, формировать отчёты и анализировать потоки данных. Особенно подходит для описания функциональной модели;
- microsoft visio – мощный инструмент для построения диаграмм, интегрирующийся с другими продуктами Microsoft. [1] Поддерживает широкий набор шаблонов, позволяет разрабатывать схемы бизнес–процессов, сетевых структур и моделей данных.

Для выбора наиболее подходящего инструмента были выделены ключевые критерии: доступность, простота интерфейса, поддержка экспортных форматов, набор доступных методологий и удобство командной работы. Результаты сравнительного анализа приведены в таблице 1.

Таблица 1 – Сравнительный анализ CASE–средств на основе выделенных критериев

Критерий	**StarUML**	**Ramus 2.0**	**Microsoft Visio**
Доступность	+	+	–
Простота интерфейса	+	–	+
Экспорт в различные форматы	+	+	+
Поддержка методологий	+	+	–
Возможности для команды	+	–	+

Таким образом, наилучшим выбором для моделирования архитектуры и логики взаимодействия в разрабатываемом мобильном приложении стало средство StarUML, соответствующее всем выделенным критериям и предоставляющее широкий набор UML–диаграмм для эффективного визуального проектирования.

Выводы по первой главе

В первой главе рассмотрена структура и деятельность компании, в которой выполнялась разработка, а также распределение ролей в команде. Проведён анализ и сравнение CASE–средств для моделирования бизнес–процессов.

По итогам анализа в качестве основного инструмента проектирования выбрано средство StarUML, обеспечивающее наглядность и удобство при разработке архитектуры приложения.

Данный этап является важной частью подготовки к дальнейшей реализации проекта и закладывает основу для разработки эффективной программной системы.

Глава 2 Описание задач и этапов разработки

2.1 Требования к персоналу, обязанности, требования к рабочему месту

Разработчик мобильных приложений должен обладать обширным техническим багажом и квалификацией, начиная от высшего образования в области компьютерных наук или смежной дисциплине. Требуется понимание основных принципов разработки мобильных приложений для iOS и/или Android, а также опыт работы с языками программирования, такими как Swift, Kotlin, Java или Dart (Flutter) [19], [20]. Опыт работы с RESTful API, системами контроля версий, адаптивным дизайном, безопасностью мобильных приложений и тестированием также являются важными компетенциями для разработчика [3].

Помимо технических навыков, разработчик мобильных приложений должен обладать способностью эффективно коммуницировать и работать в команде. Обязанности включают создание новых мобильных приложений, интеграцию новых функций в существующие приложения, тестирование и отладку для обеспечения качества и производительности приложений. Разработчик также должен участвовать в проектировании архитектуры приложений, обеспечивая их масштабируемость и надежность.

На рабочем месте при работе с ЭВМ необходимо знать и соблюдать технику безопасности для предупреждения ситуаций, в результате которых человек может получить вред [8]. К таким ситуациям, например, можно отнести короткое замыкание, приводящее к возгоранию или выходу из строя ЭВМ или его частей. Помимо этого, соблюдать требования техники безопасности необходимо и для снижения вредного воздействия ЭВМ на сотрудника и профилактики заболеваний, связанных с малоподвижной работой. Поэтому в «ВорлдИнтерТех РУС» применяются следующие основные правила техники безопасности, изложенные в «Типовой инструкции

по охране труда при работе на персональном компьютере» ТОИ Р–45–084–01 в п.2–4.

Требования безопасности перед началом работы:

- подготовить рабочее место;
- отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране;
- проверить правильность подключения оборудования к электросети;
- проверить исправность проводов питания и отсутствие оголенных участков проводов;
- убедиться в наличии заземления системного блока, монитора и защитного экрана;
- протереть антистатической салфеткой поверхность экрана и защитного экрана;
- проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение «мыши» на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела [7].

Работнику при работе на ПК запрещается:

- прикасаться к задней панели системного блока (процессора) при включенном питании;
- переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;
- допускать попадание влаги на поверхность системного блока(процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств;
- производить самостоятельное вскрытие и ремонт оборудования;
- работать на компьютере при снятых кожухах;

– отключать оборудование от электросети и выдергивать электровилку, держась за шнур.

Продолжительность непрерывной работы с компьютером без регламентированного перерыва не должна превышать 2–х часов.

Во время регламентированных перерывов с целью снижения нервно – эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления выполнять комплексы упражнений.

Во всех случаях обрыва проводов питания, неисправности заземления и других повреждений, появления гари, немедленно отключить питание и сообщить об аварийной ситуации руководителю.

Не приступать к работе до устранения неисправностей.

При получении травм или внезапном заболевании немедленно известить своего руководителя, организовать первую доврачебную помощь или вызвать скорую медицинскую помощь.

2.2 Описание поставленной задачи

Современные мобильные приложения социальных сетей играют важную роль в жизни общества, предоставляя пользователям разнообразные инструменты для общения, обмена информацией и взаимодействия. Одной из ключевых функций таких приложений являются видеозвонки и голосовые сообщения.

Разработка программного обеспечения, обеспечивающего эти функции, требует глубокого изучения предметной области, анализа существующих решений и разработки новых подходов, которые учитывают требования пользователей и современные технологические возможности.

Системы мультимедийного взаимодействия получили широкое распространение благодаря развитию высокоскоростного интернета и мобильных устройств. Такие приложения, как Zoom, Microsoft teams,

Whatsapp и Telegram, предоставляют пользователям возможности для удобного общения [1], [11].

Видеозвонки часто реализуются с использованием технологии WebRTC (Web Real–Time Communication), которая обеспечивает передачу аудио– и видеоданных в реальном времени [2], [21].

Голосовые сообщения создаются с использованием библиотек и фреймворков, таких как FFmpeg и OpenAL, которые обеспечивают высокое качество записи и передачи аудио [2], [12].

В рамках данной работы необходимо разработать программное обеспечение, которое:

- позволяет пользователям проводить видеозвонки в режиме реального времени с высоким качеством передачи аудио и видео;
- обеспечивает возможность записи, отправки и воспроизведения голосовых сообщений;
- гарантирует безопасность и приватность передаваемых данных;
- поддерживает стабильную работу в условиях низкоскоростного или нестабильного интернет–соединения;
- интегрируется в мобильное приложение социальной сети с удобным и интуитивно понятным интерфейсом.

Модель предметной области основывается на концепции клиент–серверного взаимодействия, где мобильное приложение выполняет роль клиента, а сервер обрабатывает запросы, управляет соединениями и хранит передаваемые данные.

Схема данного взаимодействия представлена на рисунке 3.

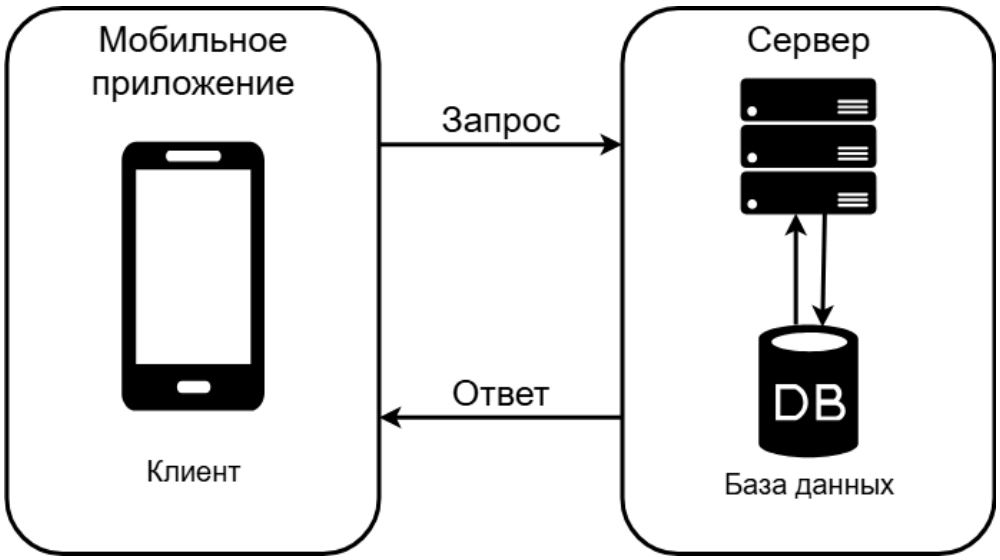


Рисунок 3 – Клиент–серверная архитектура

2.3 Выбор и обоснование методов решения задачи

Для реализации видеозвонков часто используют технологию WebRTC, которая предоставляет универсальные инструменты для передачи мультимедийных данных в реальном времени. WebRTC использует протоколы RTP/RTCP для передачи аудио– и видеопотоков, а также STUN и TURN–серверы для установления соединения между участниками, находящимися за NAT или межсетевыми экранами [5]. Однако полная настройка инфраструктуры WebRTC требует значительных усилий и технических ресурсов [17].

Для упрощения интеграции и ускорения разработки в рамках данной системы применена библиотека ZegoUIKit, предоставляющая готовое решение для видеозвонков с минимальными требованиями к конфигурации. Она скрывает сложность низкоуровневой настройки, обеспечивая при этом высокую производительность и стабильную передачу медиа–данных даже при нестабильном соединении.

Голосовые сообщения в системе реализуются с использованием библиотеки audio_waveforms, которая обеспечивает высокое качество записи,

отправки и воспроизведения аудио. Данный инструмент позволяет визуализировать звуковую волну, а также минимизирует задержки при передаче данных, что критически важно при работе в условиях ограниченной пропускной способности сети.

Одним из важнейших аспектов при выборе решений для видеосвязи и голосового взаимодействия является безопасность. WebRTC реализует сквозное шифрование потоков с использованием протоколов DTLS и SRTP, обеспечивая защищённую передачу аудио– и видеоданных. Облачная платформа Zego также предоставляет механизмы шифрования, аутентификации и защиты соединений, что позволяет соответствовать современным требованиям к конфиденциальности пользовательских данных.

Сравнительный анализ существующих решений показал, что WebRTC остаётся универсальным и мощным инструментом для реализации видеосвязи. Тем не менее, с точки зрения скорости интеграции и стабильности работы в реальных условиях предпочтение было отдано ZegoUIKit. В области голосовых сообщений библиотека audio_waveforms показала себя как надёжное и производительное решение.

Основными критериями выбора методов стали:

- простота интеграции;
- поддержка основных платформ (iOS и Android);
- наличие встроенных инструментов для обработки аудио и видео;
- оптимальная производительность в условиях ограниченной сети;
- энергоэффективность и оптимизация под мобильные устройства;
- безопасность и защита пользовательских данных.

2.4 Проектирование системы

Проектирование информационной системы является ключевым этапом в процессе разработки программного обеспечения. На данном этапе определяется структура, архитектура, логика взаимодействия компонентов и

формируется логическая модель данных, обеспечивающая целостность и взаимосвязанность информации.

Разрабатываемая система представляет собой мобильное приложение социальной сети, основной функцией которого является обмен сообщениями, в том числе голосовыми, а также возможность совершения видеозвонков. В основе архитектуры лежит клиент–серверная модель, где мобильное приложение (клиент) взаимодействует с удалённым сервером и базой данных.

Система включает следующие основные компоненты:

- клиентская часть, реализованная с использованием кроссплатформенного фреймворка Flutter. Отвечает за отображение пользовательского интерфейса, выполнение бизнес–логики, запись и воспроизведение голосовых сообщений, вызовы API сервера и взаимодействие с облачным сервисом видеосвязи;
- облачный сервис Zego Cloud SDK, используемый для реализации видеозвонков и аудиообщения в реальном времени. Поддерживает протокол WebRTC, обеспечивает стабильную передачу мультимедиа даже при нестабильном соединении; [21]
- модуль голосовых сообщений, реализованный через библиотеку Flutter Sound. Обеспечивает возможность записи и воспроизведения звуковых файлов, а также их прикрепление к сообщениям;
- серверная часть, реализованная с использованием REST API. Отвечает за обработку запросов от клиента, аутентификацию пользователей и управление сообщениями и вложениями;
- база данных MongoDB, выступающая в качестве основного хранилища данных пользователей, чатов, сообщений и вложений. Применение документно–ориентированной модели обеспечивает гибкость при расширении структуры данных и высокой производительности при работе с большими объемами информации.

Все взаимодействия между клиентом, сервером и облачными сервисами происходят через защищённые каналы передачи данных (HTTPS), что гарантирует безопасность передаваемой информации.

Для хранения и управления данными в разрабатываемом приложении используется документо–ориентированная база данных MongoDB, которая обеспечивает гибкую схему хранения информации и высокую масштабируемость.

Схема коллекций базы данных представлена на рисунке 4.

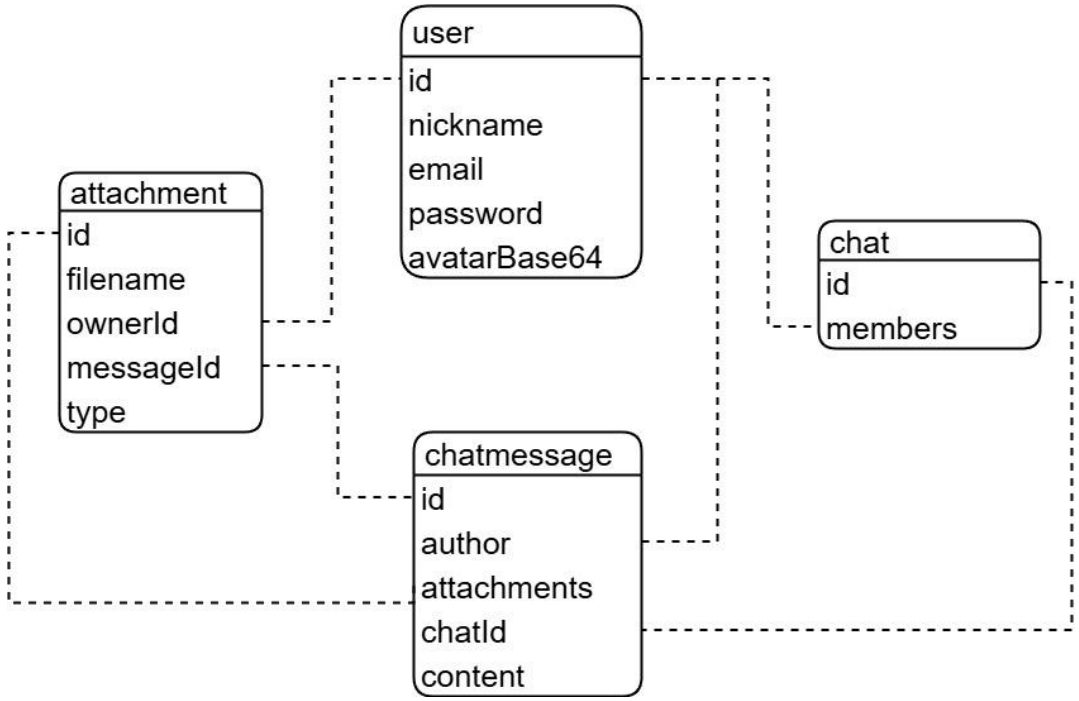


Рисунок 4 – Схема коллекций базы данных

Для обеспечения использования мобильного приложения пользователем применяется диаграмма прецедентов, которая иллюстрирует все возможные действия и взаимодействия пользователя с системой. Эта диаграмма представлена на рисунке 5.

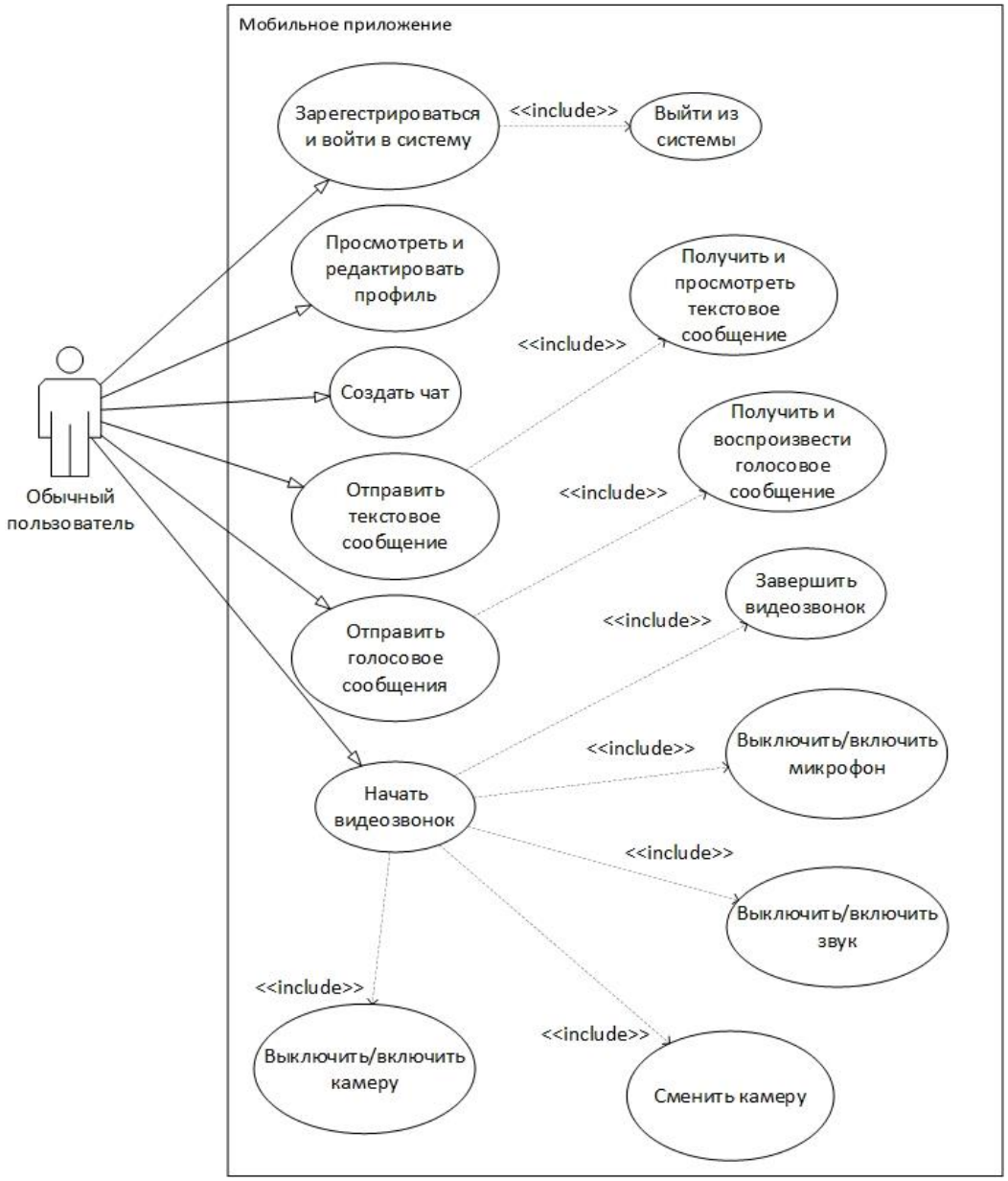


Рисунок 5 – Диаграмма вариантов использования

Для более наглядного представления последовательности действий пользователя и логики работы интерфейса мобильного приложения была разработана диаграмма активностей (рисунок 6). Она отражает основные этапы взаимодействия пользователя с системой – от запуска приложения и авторизации до отправки сообщений и совершения видеозвонков. Диаграмма позволяет визуализировать поток управления, определить ключевые сценарии

использования и выявить возможные точки улучшения пользовательского опыта.

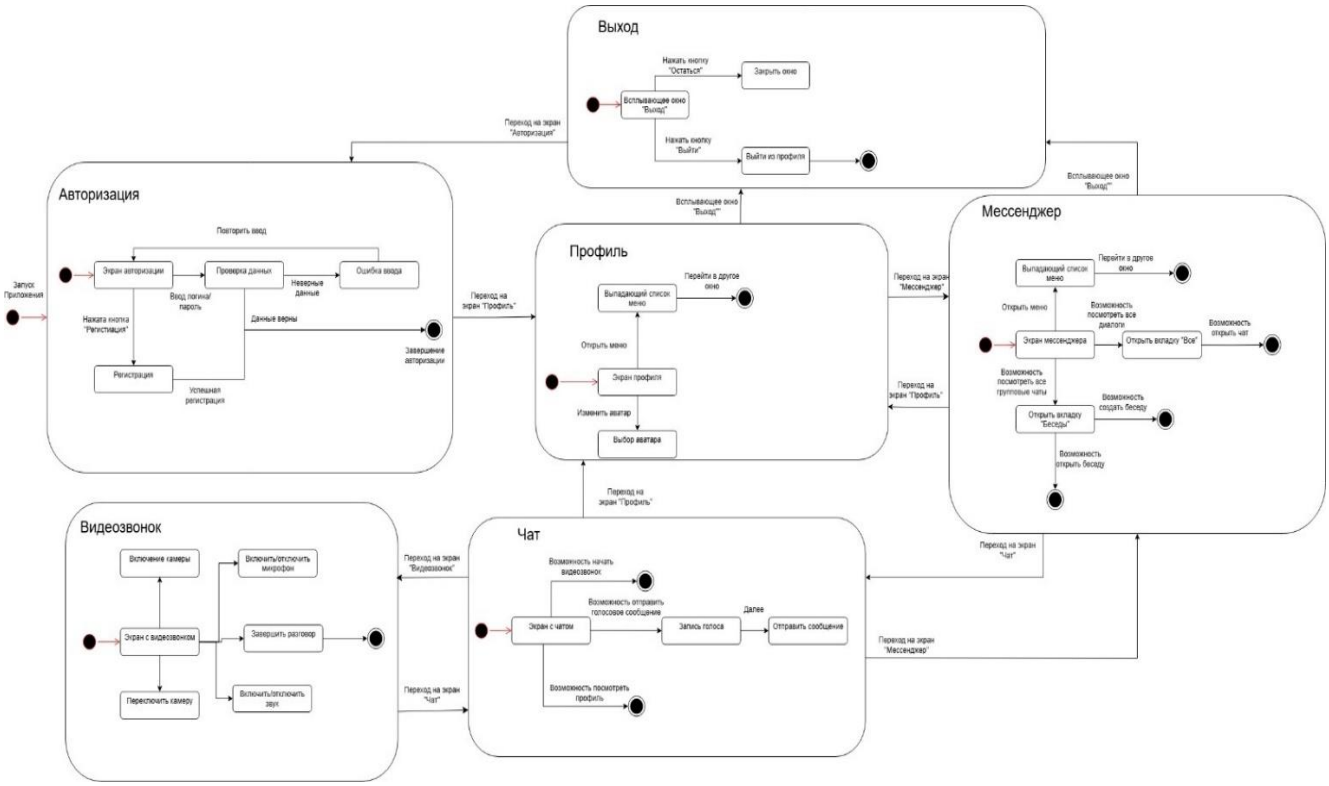


Рисунок 6 – Диаграмма активностей мобильного приложения

Для демонстрации визуального представления реализуемого функционала был разработан макет пользовательского интерфейса, включающего элементы взаимодействия для голосовых сообщений и видеозвонков.

Он служит основой для реализации UI–компонентов во Flutter и позволяет оценить удобство навигации и эргономику элементов управления на мобильных устройствах.

Данный макет пользовательского интерфейса отображен на рисунке 7.

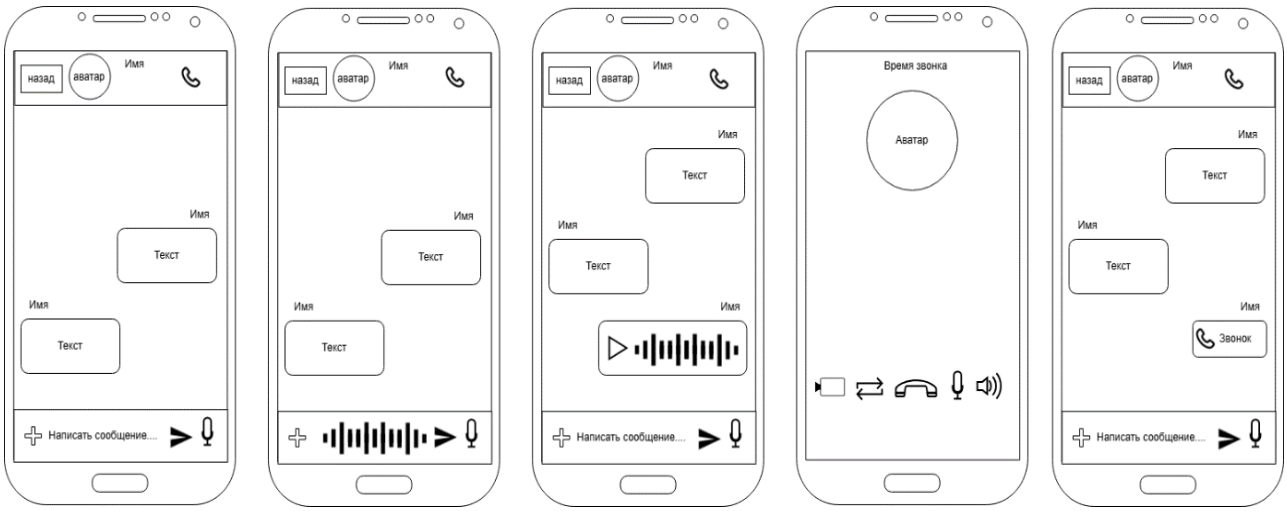


Рисунок 7 – Прототип интерфейса

На рисунке представлены основные сценарии взаимодействия пользователя:

- окно чата с обновлённым интерфейсом: добавлены кнопки микрофона – для начала записи голосового сообщения, и трубки – для инициации видеозвонка;
- состояние активной записи голосового сообщения: форма текстового ввода заменяется анимированной звуковой волной, отображающей интенсивность речи пользователя в реальном времени;
- отправленное голосовое сообщение отображается в ленте чата в виде аудиоблока, доступного для прослушивания;
- интерфейс активного видеозвонка: пользовательский экран с элементами управления – отключение микрофона, камера, завершение звонка и информация об участнике;
- специальное сообщение «Звонок» в окне чата, отображающее приглашение на видеозвонок. При нажатии пользователь присоединяется к текущему вызову.

2.5 Структура и компоненты системы

Система состоит из следующих компонентов:

- клиентское приложение на Flutter;
- сервер Zego Cloud;
- локальная обработка голосовых сообщений.

Клиентское приложение отвечает за пользовательский интерфейс и взаимодействие с сервером. Flutter позволяет эффективно разделить логику и интерфейс приложения, что значительно упрощает поддержку и масштабирование. На клиенте реализованы UI–модуль для отображения видеозвонков, а также модуль логики для обработки вызовов и API для работы с сервером Zego Cloud и локальными функциями записи голоса.

Сервер Zego Cloud выполняет маршрутизацию видеозвонков, управляет соединениями и балансирует нагрузку. [21] Он взаимодействует с клиентскими приложениями через API, обеспечивая надежную передачу данных в реальном времени и эффективную обработку видео– и аудиопотоков.

Локальная обработка голосовых сообщений – эта компонента отвечает за запись, обработку и воспроизведение голосовых сообщений на устройстве пользователя, минимизируя задержки и используя оптимальные алгоритмы для работы в условиях ограниченной пропускной способности сети.

Концептуальная модель реализована в виде взаимодействия модулей:

- ui–модуль для отображения видеозвонков;
- модуль логики для обработки вызовов;
- api для работы с Zego Cloud и локальными функциями записи голоса.

Концептуальная модель системы построена на классических принципах клиент–серверной архитектуры, где мобильное приложение выступает в роли клиента, а локальный сервер обеспечивает обработку запросов, управление данными и взаимодействие с пользователем. Локальный сервер играет ключевую роль, выступая центром логики и координации работы системы, что позволяет добиться высокой отзывчивости и эффективности работы

приложения. При этом для реализации видеозвонков используется сторонний облачный сервис Zego Cloud, который отвечает за маршрутизацию и передачу видеопотоков между участниками в режиме реального времени, обеспечивая высокое качество связи и минимальные задержки. Остальные функции, такие как обработка голосовых сообщений, а также управление пользовательским интерфейсом и бизнес-логикой, целиком сосредоточены на локальном сервере, что позволяет гибко контролировать процесс и поддерживать безопасность данных.

Выводы по второй главе

Во второй главе сформулированы требования к разработчику и рабочему месту, а также подробно описаны цели, задачи и методы реализации функциональности видеозвонков и голосовых сообщений в мобильном приложении.

Проведён обоснованный выбор технологий и инструментов, обеспечивающих надёжность, производительность и безопасность системы.

Представлены ключевые этапы проектирования, включая создание диаграмм, моделей данных и пользовательского интерфейса.

Разработана структура системы с указанием её основных компонентов и принципов взаимодействия, что обеспечило основу для дальнейшей реализации программного обеспечения.

Глава 3 Разработка алгоритма, программирование и откладка

3.1 Описания средств для решения поставленных задач

Разработка мобильного приложения велась на языке программирования Dart с использованием фреймворка Flutter. Выбор данной технологии обусловлен тем, что ранее проект уже был реализован мной в рамках работы в компании, где использовалась именно эта связка инструментов. Dart обладает лаконичным синтаксисом и обеспечивает высокую производительность, а Flutter предоставляет средства кроссплатформенной разработки, что позволяет создавать единый код для устройств под управлением Android и iOS [16], [18].

Благодаря встроенной поддержке компонентов UI и широкому сообществу, Flutter значительно сокращает время разработки и упрощает интеграцию внешних библиотек.

Таким образом, сохранение языка Dart и платформы Flutter позволяет не только использовать наработки предыдущей версии приложения, но и масштабировать проект с минимальными издержками на переобучение или рефакторинг [14].

В качестве среды разработки использовалась Android Studio, обеспечивающая полноценную поддержку Flutter и языка Dart. Среда обладает встроенным эмулятором Android, системой контроля версий, инструментами для профилирования и анализа производительности, что делает её удобным инструментом для реализации и тестирования мультимедийных функций [11], [12].

Кроме того, Android Studio позволяет быстро интегрировать внешние библиотеки и плагины, обеспечивая гибкость при адаптации архитектуры проекта под новые требования. Поддержка Hot Reload значительно ускоряет процесс отладки пользовательского интерфейса и логики взаимодействия [18].

Выбор данной среды обусловлен как опытом предыдущей работы над проектом, так и наличием всего необходимого инструментария для внедрения новых технологических решений в рамках данной работы.

Реализация видеозвонков и голосовых сообщений достигается благодаря интеграции Zego Cloud SDK и пакета flutter_sound. Zego Cloud предоставляет инструменты для маршрутизации аудио– и видеоданных, а также управления качеством соединения. Пакет flutter_sound упрощает работу с записью и воспроизведением аудиофайлов, обеспечивая гибкость в настройке голосовых сообщений [21].

Управление проектом осуществляется с помощью Git и платформы GitHub. Git обеспечивает контроль версий, позволяя легко отслеживать изменения и при необходимости откатывать их. GitHub используется для хранения кода и организации совместной работы, поддерживая автоматическое тестирование и деплой через CI/CD системы. Комплексное применение этих инструментов позволяет эффективно решать задачи разработки и соответствовать современным стандартам программного обеспечения [4].

3.2 Задачи решаемые в процессе разработки программного обеспечения

В ходе работы решаются следующие задачи, направленные на развитие профессиональных навыков и реализацию программного продукта. Первой задачей является изучение современных технологий, которые применяются для создания функционала видеозвонков и голосовых сообщений. Это включает знакомство с инструментами, такими как Flutter, Zego Cloud SDK и flutter_sound, а также их интеграцию в проект.

Важным этапом является проектирование архитектуры приложения, где разрабатывается структура данных, интерфейсы и взаимодействие между компонентами. Это помогает оптимизировать производительность и

обеспечить удобство использования. Также проводится работа над реализацией алгоритмов обработки аудио– и видеоданных, их маршрутизации и синхронизации между клиентами.

Не менее значимой задачей является тестирование готового решения. На данном этапе проводится анализ производительности приложения, оценка стабильности при различных условиях работы и проверка интуитивности пользовательского интерфейса. Тестирование позволяет выявить слабые места и внести необходимые коррективы для улучшения качества продукта.

3.3 Выполнение задания

В ходе выполнения задания используется ранее разработанное мною приложение социальной сети. Это приложение служит базовой платформой, в которую интегрируются новые функции: отправка голосовых сообщений и видеозвонки. Такой подход позволяет не только улучшить существующий продукт, но и адаптировать его к новым требованиям пользователей.

На первом этапе реализована функция голосовых сообщений. В файл AndroidManifest.xml добавлены разрешения на запись аудио (рисунок 8) [3].

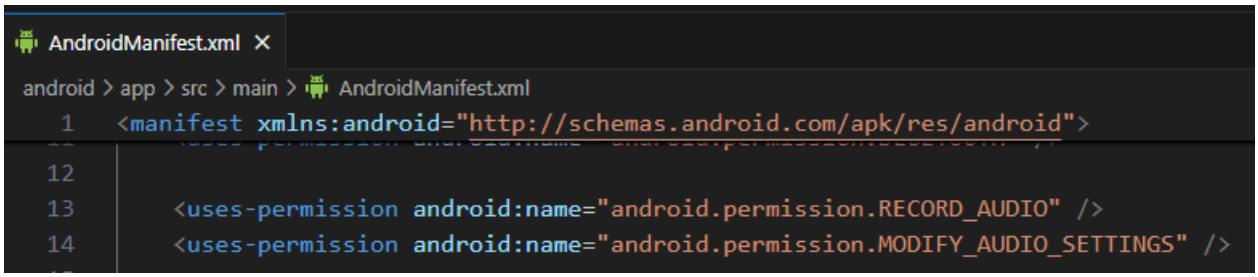


Рисунок 8 – Разрешения на запись

Затем в файл build.gradle внесена строка «minSdkVersion 24», определяющая минимальную поддерживаемую версию Android SDK, с которой будет работать приложение [2], [15].

Следующим шагом добавлена функция записи голосовых сообщений с визуализацией аудиоволны.

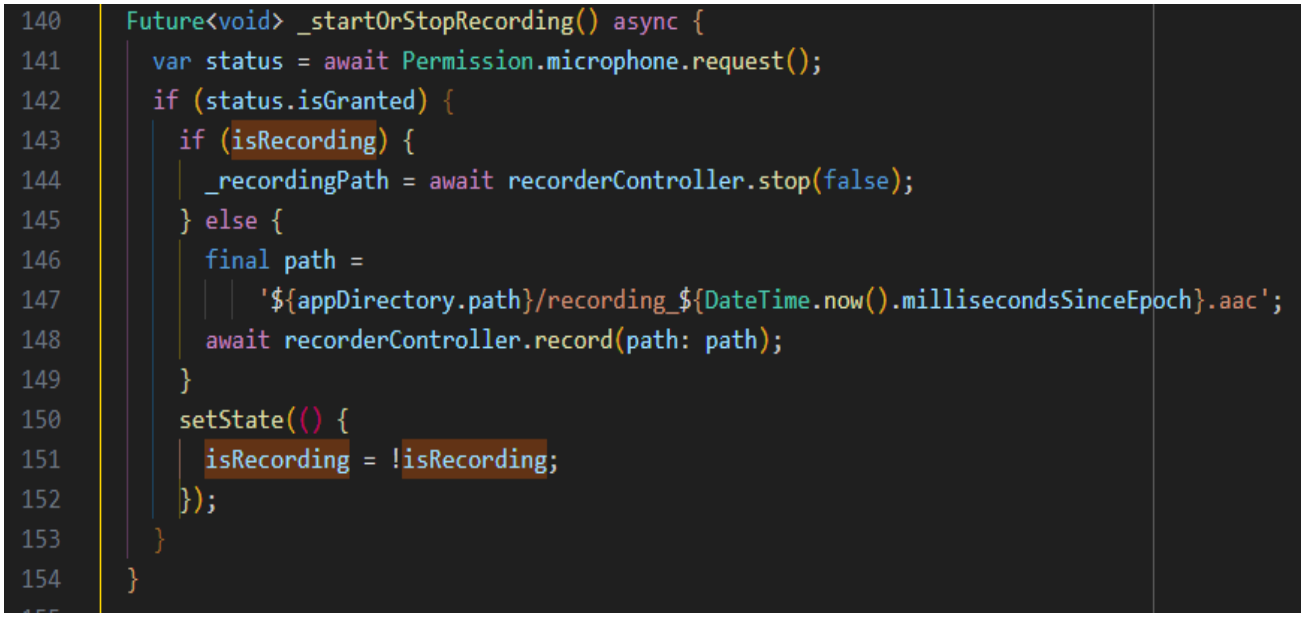
С использованием пакета audio_waveforms в методе initState создан объект RecorderController, а также настроена директория для сохранения файлов через метод _initializeAppDirectory (рисунок 9).

Это позволило определить единое хранилище для аудиофайлов.

```
117   @override
118   void initState() {
119     // TODO: implement initState
120     super.initState();
121     initData();
122     initUser();
123     initMessages();
124     initSocket();
125     _scrollController.addListener(() {
126       _scrollListener();
127     });
128
129     //Контроллер записи
130     recorderController = RecorderController();
131     //Инициализация места хранения файлов
132     _initializeAppDirectory();
133   }
134
135   //Инициализация путя
136   Future<void> _initializeAppDirectory() async {
137     appDirectory = await getApplicationDocumentsDirectory();
138   }
```

Рисунок 9 – Инициализация объектов

Запись сообщений реализована в методе _startOrStopRecording (рисунок 10), где осуществляется проверка доступа к микрофону через библиотеку permission_handler. При активной записи происходит её остановка и сохранение пути к аудиофайлу. В случае отсутствия активной записи создаётся новый файл формата .aac и начинается запись.



```
Future<void> _startOrStopRecording() async {
  var status = await Permission.microphone.request();
  if (status.isGranted) {
    if (isRecording) {
      _recordingPath = await recorderController.stop(false);
    } else {
      final path =
          '${appDirectory.path}/recording_${DateTime.now().millisecondsSinceEpoch}.aac';
      await recorderController.record(path: path);
    }
    setState(() {
      isRecording = !isRecording;
    });
  }
}
```

Рисунок 10 – Метод _startOrStopRecording

После завершения записи аудиофайл сохраняется локально на устройстве и автоматически добавляется в список _pickedFile, который используется для хранения всех выбранных или созданных файлов. Это позволяет удобно управлять мультимедийным контентом внутри приложения.

Далее файл отправляется на сервер с помощью метода postAttachments (рисунок 11).

Этот метод формирует MultipartRequest – запрос, который включает не только сам аудиофайл, но и важные метаданные – messageId, chatId и userId.

Такая структура запроса обеспечивает правильную идентификацию и обработку файла на сервере, позволяя связать его с конкретным сообщением, чатом и пользователем, что гарантирует корректную работу функционала обмена голосовыми сообщениями.

```dart
Future<void> postAttachments(List<PlatformFile> file, String cookie) async {
  var url = Uri.parse('$SERVER_URL/api/user/postAttachments');

  var request = http.MultipartRequest(
    "POST",
    url,
  )
    ..fields['messageId'] = messageId
    ..fields['chatId'] = widget.idChat
    ..fields['id'] = currentUser!.id
    ..headers['Content-Type'] = 'multipart/form-data'
    ..headers['cookie'] = cookie;

  for (int i = 0; i < file.length; i++) {
    request.files
        .add(await http.MultipartFile.fromPath("attachments", file[i].path!));
  }
  try {
    var response = await request.send();
    var responseData = await response.stream.bytesToString();

    print(responseData);

    if (response.statusCode == 200) {
      print('[Server](testChatPage: postAttachments): ${responseData}');

      Map<String, dynamic> _json = json.decode(responseData);
      _attachsList = _json['attachs'];
    } else {
      print('Необработанный статус код: ${response.statusCode}');
    }
  } catch (e) {
    print('Ошибка при отправке запроса: $e');
  }
}
```

Рисунок 11 – Метод postAttachments на клиенте

На стороне сервера этот запрос обрабатывается маршрутом /postAttachments (рисунок 12), который принимает MultipartRequest, содержащий один или несколько файлов и соответствующие мета–данные. После получения данных сервер сохраняет загруженные файлы, создавая для каждого из них объект модели Attachment, включающий информацию о владельце, типе файла и связях с сообщением и чатом. Каждый объект сохраняется в базе данных, а по завершении процесса сервер формирует и

отправляет клиенту JSON–ответ, подтверждающий успешную загрузку и содержащий мета–данные загруженных вложений.

```
/**
 * @name POST /postAttachments
 * @route {POST} /postAttachments
 */
userRouter.post("/postAttachments", attachUpload.any("attachments"), (req, res) => {
    let { id } = req.session.user || jwt.verify(req.token, process.env.JWT_SECRET || "development");

    let attachPromises = req.files.map(file => {
        console.log(file);
        let attach = new Attachment({
            id: uuid(),
            filename: file.filename,
            ownerId: id,
            chatId: req.body.chatId,
            messageId: req.body.messageId,
            type: mime.getType(file.filename).split("/")[0]
        });

        return attach.save();
    });

    Promise.all(attachPromises).then(data => {
        res.json({ attachmentsUploaded: true, attachs: data });
    });
});
```

Рисунок 12 – Метод postAttachments на сервере

Получив ответ, приложение вызывает метод sendMessage (рисунок 13), который через сокет отправляет событие private_message, содержащее информацию о новом сообщении и ссылку на вложение.

Этот процесс реализует асинхронную отправку сообщения, позволяя мгновенно уведомить других участников чата о новом сообщении без необходимости обновления страницы. Подобный подход обеспечивает реализацию функционала в режиме реального времени и улучшает взаимодействие пользователей с приложением, создавая ощущение «живого» диалога.

```
468  void sendMessage(String message, List<dynamic> attachs) {
469    Map messageMap = {
470      'content': message.trim(),
471      'author': currentUser!.id,
472      'chatId': widget.idChat,
473      'id': messageId,
474      'timestamp': new DateTime.now().millisecondsSinceEpoch,
475    };
476
477    MyMessage item = MyMessage(
478      id: messageId,
479      author: currentUser!.id,
480      attachments: attachs,
481      chatId: widget.idChat,
482      fromId: widget.userId,
483      timestamp: new DateTime.now().millisecondsSinceEpoch,
484      content: message,
485      authorNickname: '',
486    ); // MyMessage
487    messagesManager.addMessages(item);
488
489    socket.emit('private_message', jsonEncode(messageMap));
490    _messageController.clear();
491  }
```

Рисунок 13 – Метод sendMessage

На сервере WebSocket–событие private_message (рисунок 14) обрабатывается следующим образом: принимается объект data, содержащий данные о сообщении, создаётся новый объект ChatMessage, который сохраняется в базе данных, затем происходит агрегация вложений, связанных с этим сообщением, и формируется итоговая структура ответа. В ответе указывается никнейм отправителя и список вложений, после чего событие private_message повторно отправляется сервером другим участникам чата.

Такой механизм обработки позволяет не только сохранить сообщение в базе, но и обеспечить его полную доставку с прикреплёнными файлами и необходимыми мета-данными. Это важно для отображения всех элементов интерфейса получателя.

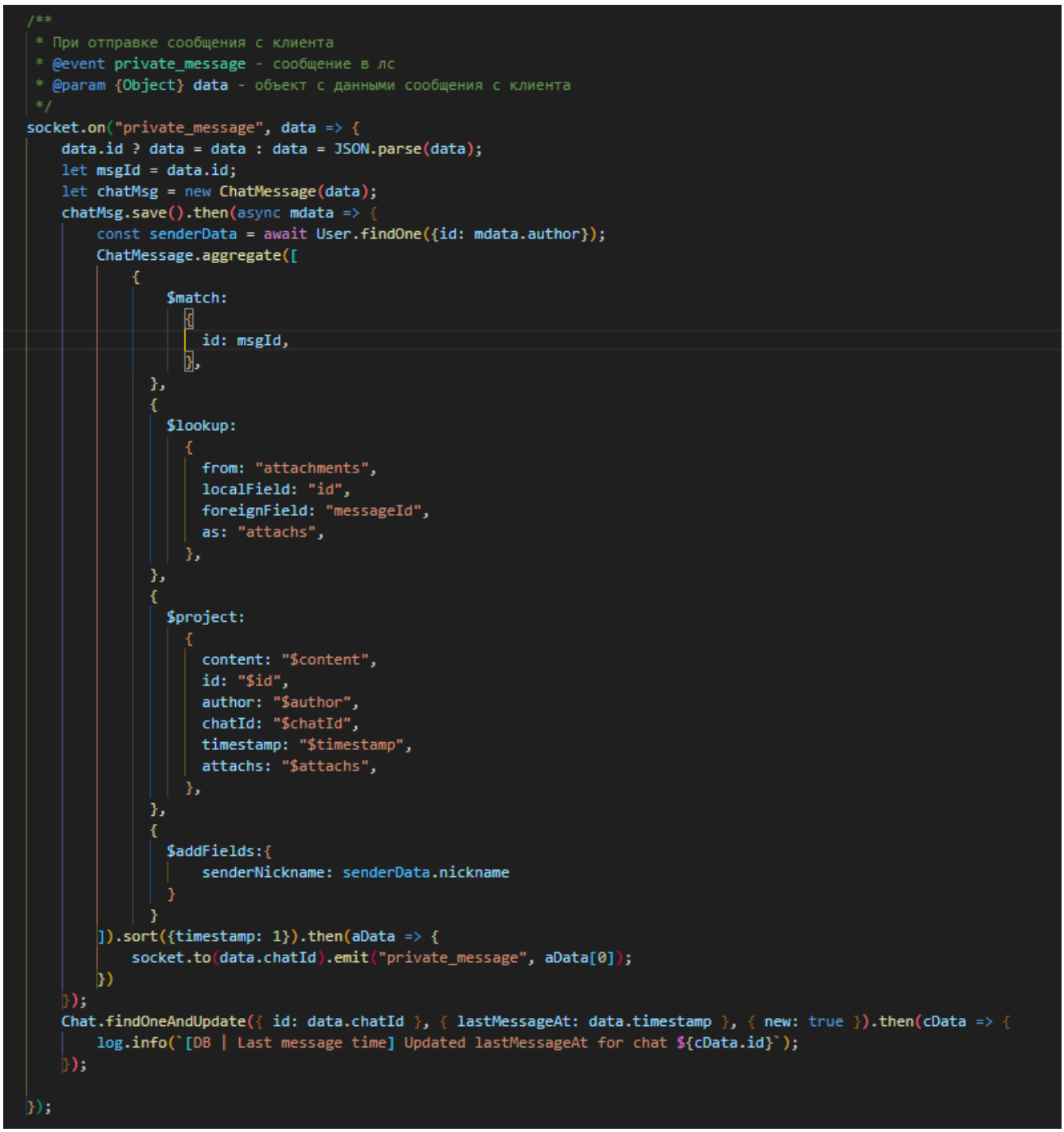


Рисунок 14 – Событие private_message на сервере

На рисунке 15 представлена блок–схема, иллюстрирующая последовательность действий при отправке голосового сообщения от клиента к серверу, включая обработку вложений, формирование WebSocket–сообщения и доставку его другим участникам чата. Легенда к условным обозначениям схемы представлена в приложении А.

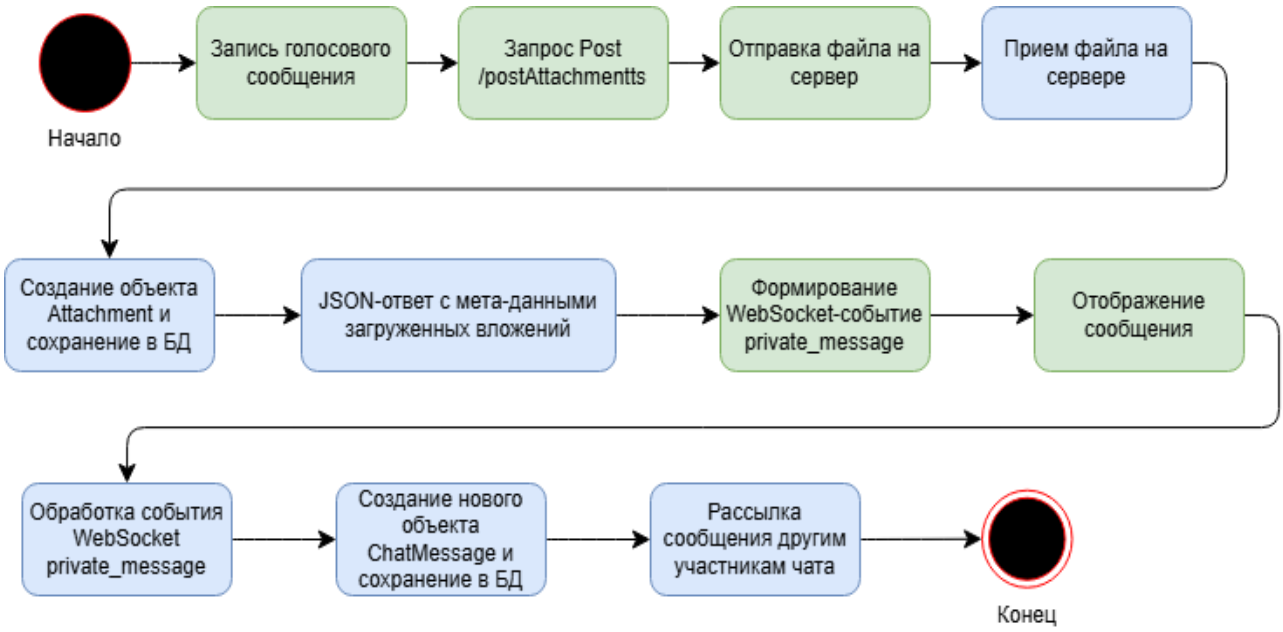


Рисунок 15 – Последовательность обработки голосового сообщения на клиенте и сервере

Для отображения и воспроизведения полученных голосовых сообщений в приложении использован специализированный виджет WaveBubble, который визуализирует аудиосигнал в виде волновой формы, делая процесс прослушивания более наглядным и удобным для пользователя. Взаимодействие с аудиофайлами осуществляется через контроллер PlayerController [6], который отвечает за управление воспроизведением – запуск, паузу, остановку и регулировку громкости, обеспечивая плавный и отзывчивый пользовательский опыт.

После загрузки аудиофайлы сохраняются в специальной директории внутри приложения, что позволяет надежно хранить их на устройстве и быстро получать к ним доступ при необходимости воспроизведения. Далее эти сохранённые файлы проходят подготовку.

На рисунках 16 и 17 наглядно показаны этапы отображения и управления голосовыми сообщениями.

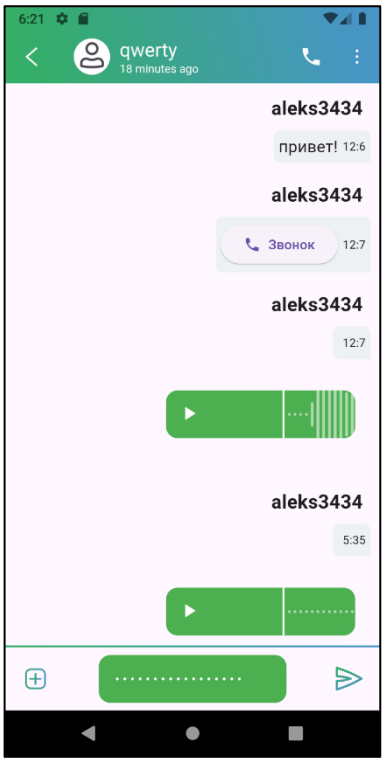


Рисунок 16 – Пользовательский интерфейс при записи голосового сообщения

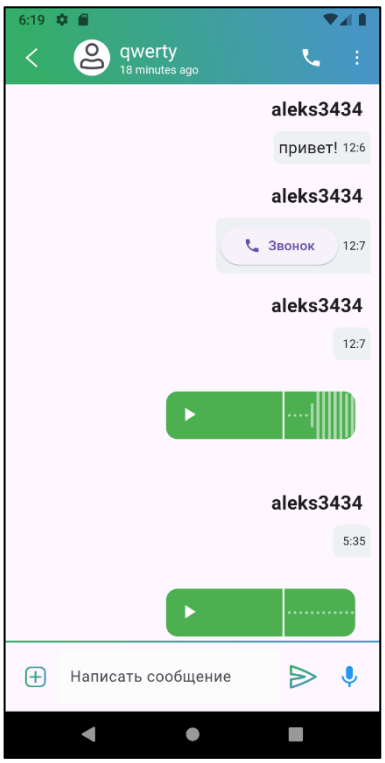


Рисунок 17 – Отображение голосовых сообщений

После реализации голосовых сообщений была выполнена интеграция видеозвонков. Для настройки ZegoCloud в проекте на Flutter были добавлены необходимые зависимости в pubspec.yaml (рисунок 18) [3].

Затем в файл AndroidManifest.xml были внесены необходимые разрешения для корректной работы функций, таких как видеозвонки, запись звука и управление настройками устройства (рисунок 19).

В папке проекта был создан файл proguard-rules.pro (рисунок 20), используемый для настройки инструмента ProGuard, отвечающего за оптимизацию и минимизацию кода Android-приложения.

В данном файле была прописана директива, запрещающая обфускацию и оптимизацию классов и методов, относящихся к библиотекам Zego, что необходимо для их корректного функционирования даже после сборки и минимизации проекта.

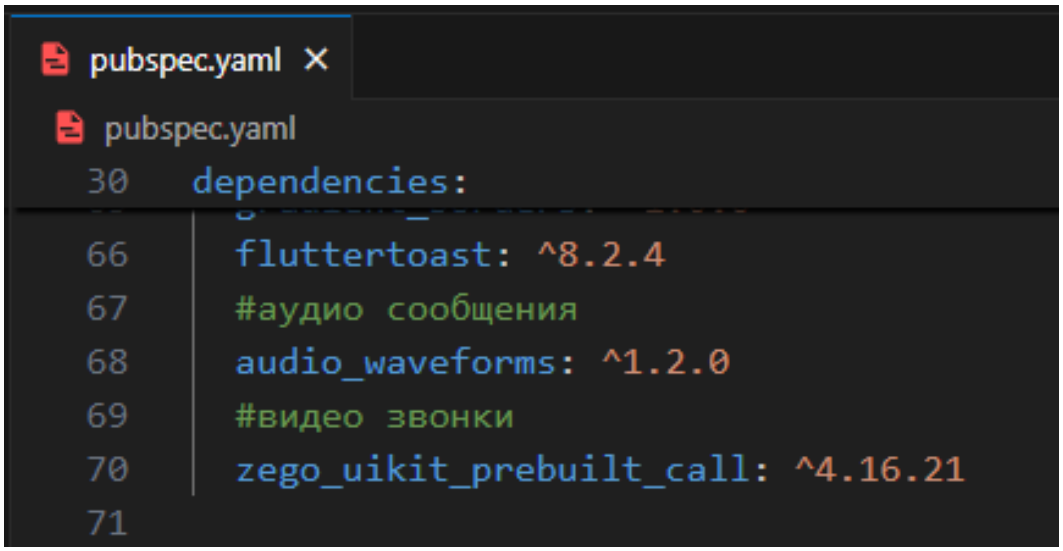


Рисунок 18 – Зависимости в pubspec.yaml

AndroidManifest.xml

android > app > src > main > AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO"/>
    <uses-permission android:name="android.permission.READ_MEDIA_AUDIO"/>

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.BLUETOOTH" />

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

    <uses-permission android:name="android.permission.CAMERA" />

    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.VIBRATE"/>

    <uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY"/>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>

    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />

```

Рисунок 19 – Разрешения в AndroidManifest.xml

proguard-rules.pro

android > app > proguard-rules.pro

```
-keep class **.zego.** { *; }

```

Рисунок 20 – Файл proguard–rules.pro

После завершения этапа настройки была выполнена реализация функционала видеозвонков. Для интеграции видеозвонков в мобильное приложение использован виджет ZegoUIKitPrebuiltCall, обеспечивающий удобное и быстрое подключение. Основная логика интерфейса и подключения реализована в классе CallPage (рисунок 21), в котором осуществляется

передача необходимых параметров (roomID, userID, userName) и настройка конфигурации вызова. При этом обработка связи между пользователями осуществляется не внутри приложения, а на стороне сервера – в облачной инфраструктуре Zego. [21]

```
2382  class CallPage extends StatelessWidget {
2383    const CallPage(
2384        {Key? key,
2385        required this.RoomID,
2386        required this.userID,
2387        required this.userName})
2388        : super(key: key);
2389    final String RoomID;
2390    final String userID;
2391    final String userName;
2392
2393    @override
2394    Widget build(BuildContext context) {
2395      return ZegoUIKitPrebuiltCall(
2396        appID: 1220081376, // Your App ID
2397        appSign:
2398            '7752b3aa7dde9a25dd76cf64f4105a9bbcc91cc62cdb0402ab3f758e0662ad5f',
2399        userID: userID,
2400        userName: userName,
2401        callID: RoomID,
2402        config: ZegoUIKitPrebuiltCallConfig.oneOnOneVideoCall(),
2403      ); // ZegoUIKitPrebuiltCall
2404    }
2405  }
```

Рисунок 21 – Класс CallPage

Платформа Zego Cloud использует архитектуру облачного взаимодействия. Это означает, что все ключевые процессы – установление соединения, маршрутизация медиапотоков, передача сигнальных сообщений и оптимизация качества связи – выполняются на серверах Zego без участия стороннего backend–разработчика. На сервере Zego обработка вызова организована следующим образом.

Когда пользователь запускает ZegoUIKitPrebuiltCall, SDK инициирует защищённое соединение с серверами Zego, передавая параметры авторизации:

appID, appSign, userID, userName и callID. На основе этих данных сервер проверяет подлинность клиента и разрешает доступ к указанной комнате. [21] Если комната с заданным callID уже существует, пользователь подключается к ней; в противном случае – комната создаётся автоматически.

После успешного подключения начинается захват аудио– и видеопотоков с устройства пользователя. Эти данные передаются в облако, где маршрутизируются в реальном времени к другим участникам вызова. Одновременно с этим передаются сигнальные сообщения (например, информация о присоединении или отключении пользователя, включении или отключении микрофона и камеры). Все эти события обрабатываются сервером Zego и доставляются участникам звонка, что обеспечивает синхронность и стабильную работу видеосвязи.

Кроме того, облачная система Zego реализует механизмы адаптивной передачи. Серверы автоматически отслеживают сетевые параметры (задержку, пропускную способность, потери пакетов) и регулируют качество аудио– и видеопотоков, обеспечивая максимально комфортную связь для пользователей.

Таким образом, вся серверная логика взаимодействия между участниками видеозвонка реализована средствами Zego. Разработчику необходимо лишь корректно настроить клиентскую часть и передать необходимые параметры – остальное выполняет облачная платформа.

Дополнительно была реализована логика уведомления о видеозвонке: когда один из участников чата инициирует звонок (рисунок 22), в диалог автоматически отправляется системное сообщение с интерактивной кнопкой. Нажав на неё, второй пользователь получает возможность мгновенно подключиться к текущему вызову (рисунок 23).

Рисунок 22 – Пользовательский интерфейс при видеозвонке

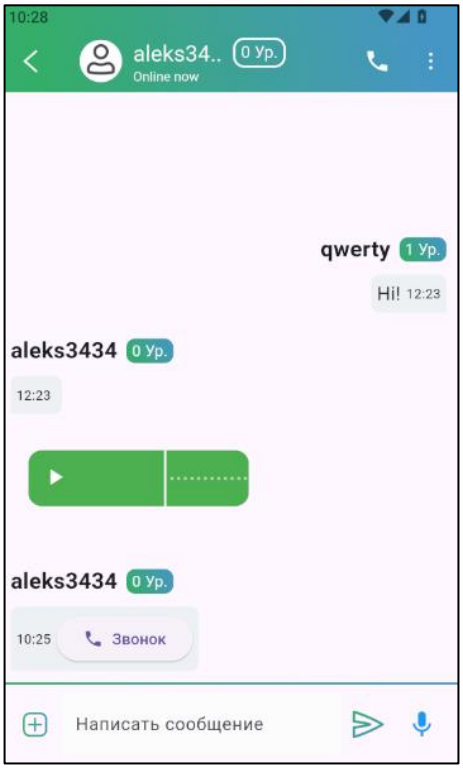


Рисунок 23 – Отображение кнопки присоединиться к звонку

Для лучшего понимания архитектуры видеозвонков, реализуемой с помощью ZegoUIKit, ниже представлена общая схема, демонстрирующая взаимодействие между мобильными клиентами, серверной

частью приложения, базой данных и облачной платформой ZEGOCLOUD.

Эта схема отражает поток данных и основные компоненты, задействованные в процессе установления и обработки видеозвонков в реальном времени.

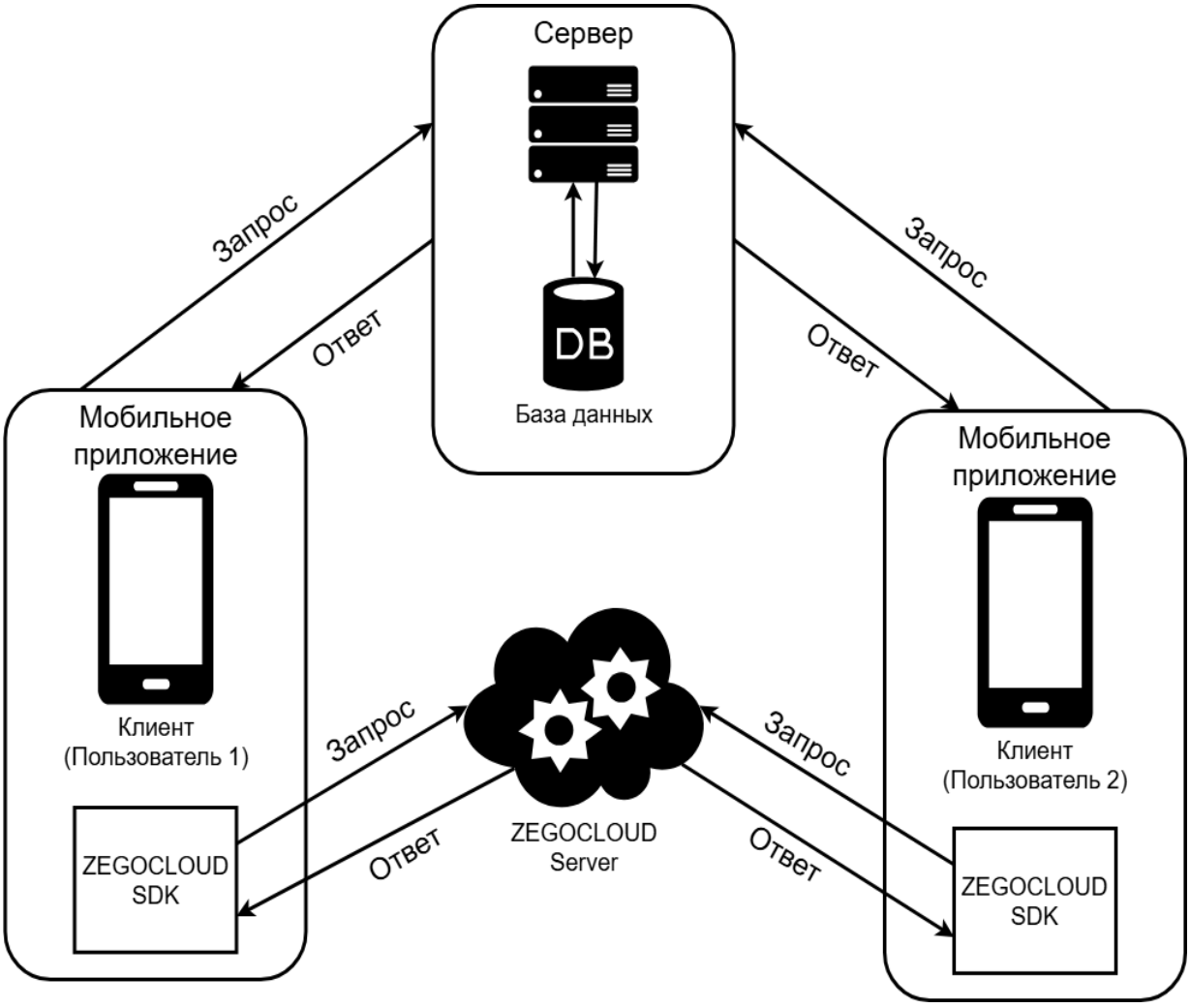


Рисунок 24 – Архитектура взаимодействия приложения с ZEGOCLOUD при видеозвонке

Таким образом, используя этот класс, мне удалось быстро реализовать интерфейс и логику подключения видеозвонков, минимизируя сложность настройки и интеграции. Это решение позволяет сосредоточиться на пользовательском опыте и функциональности.

3.4 Тестирование

Тестирование функционала видеозвонков позволило мне оценить их производительность и выявить ключевые аспекты, требующие доработки. Результаты показали, что приложение успешно справляется с подключением к звонкам в условиях стабильного интернета, обеспечивая высокое качество видео и аудио. При использовании Wi–Fi задержка составляла не более 100 мс, что обеспечивает практически незаметную синхронизацию между участниками. В условиях мобильной сети, особенно 4G, качество оставалось приемлемым, хотя задержка увеличивалась до 100–200 мс.

Тесты на стабильность показали, что приложение устойчиво к многократным входам и выходам из комнаты. Даже при отключении одного из участников система корректно завершала его сеанс, освобождая ресурсы, такие как камера и микрофон, и обновляла интерфейс для оставшихся участников.

Дополнительно были проведены тесты корректности отображения интерфейса видеозвонков на устройствах с различными диагоналями и разрешениями экранов. Интерфейс адаптивно подстраивался под размеры экрана, элементы управления (кнопки отключения камеры, микрофона, завершения вызова) оставались доступными и не перекрывались. Также успешно прошла проверку логика подключения по приглашению: при получении сообщения с кнопкой вызова другой пользователь мог без ошибок присоединиться к текущему видеозвонку, и соединение происходило корректно.

Проведение этих тестов позволило не только подтвердить работоспособность реализованного функционала, но и улучшить общее качество пользовательского опыта, выявив направления для дальнейшего развития и оптимизации.

Выводы по третьей главе

В третьей главе подробно рассмотрен процесс реализации мобильного приложения, включающего функции видеозвонков и голосовых сообщений.

Приведены используемые инструменты, обоснован выбор технологий, описаны ключевые этапы программирования – от настройки среды и интеграции библиотек до разработки пользовательского интерфейса и логики обработки аудио- и видеоданных.

Отдельное внимание уделено взаимодействию клиента с серверной частью и облачным сервисом Zego.

Проведённое тестирование подтвердило стабильность работы приложения в различных условиях, корректность отображения интерфейса и высокую степень готовности программного продукта к практическому применению.

Заключение

В ходе выполнения данной бакалаврской работы была разработана система видеозвонков и голосовых сообщений для мобильного приложения социальной сети. Основной целью проекта стало создание функционального, устойчивого и безопасного решения, обеспечивающего высокое качество связи, простоту взаимодействия и удобную интеграцию с существующей платформой.

На первом этапе работы проведён детальный анализ предметной области, изучены современные технологии передачи аудио– и видеоданных, такие как WebRTC, Zego Cloud SDK и Flutter Sound. Выбор указанных инструментов обоснован их высокой производительностью, простотой интеграции, а также поддержкой кроссплатформенной разработки, что особенно важно в условиях современной мобильной среды.

На основе результатов анализа спроектирована клиент–серверная архитектура системы, включающая локальный сервер для обработки пользовательских запросов и облачный сервис Zego Cloud для маршрутизации видеозвонков и обеспечения стабильной передачи данных.

Реализация системы потребовала решения ряда технических задач. На клиентской стороне, разработанной с использованием фреймворка Flutter, были интегрированы специализированные библиотеки: flutter_sound – для записи и воспроизведения голосовых сообщений, а также ZegoUIKit – для реализации видеозвонков.

Особое внимание было уделено обеспечению безопасной передачи данных через HTTPS и WebSocket, а также оптимизации работы системы при нестабильном интернет–соединении.

Серверная часть реализована с использованием REST API для обработки вложений и WebSocket-соединений для мгновенной доставки сообщений в режиме реального времени.

Проведённое тестирование подтвердило работоспособность всех ключевых компонентов системы. Видеозвонки демонстрируют стабильную работу с задержкой не более 100–200 мс даже в условиях использования мобильных сетей. Голосовые сообщения корректно записываются, передаются и воспроизводятся без искажений, а интерфейс приложения обеспечивает интуитивно понятное взаимодействие с пользователем, снижая порог вхождения и повышая пользовательский комфорт.

Разработанное решение представляет собой законченный продукт, который может быть интегрирован в мобильные социальные сети, корпоративные платформы и сервисы удалённого взаимодействия.

Таким образом, поставленные цели работы достигнуты: создано программное обеспечение, обеспечивающее высокое качество видеозвонков и голосовых сообщений, безопасность данных и удобство использования. Результаты проекта могут быть применены для совершенствования коммуникационных возможностей мобильных приложений.

Список используемой литературы

1. Андиева Е. Ю., Сидоренко В. С. Анализ Критериев Выбора Case–Средств //Редакционная коллегия. – 2015. – 164 с.
2. Баланов, А. Н. Комплексное руководство по разработке: от мобильных приложений до веб–технологий : учебное пособие для вузов / А. Н. Баланов. – Санкт–Петербург : Лань, 2024. – ISBN 978–5–507–48841–4. – Текст : электронный // Лань : электронно–библиотечная система. – URL: https://e.lanbook.com/book/394577 (дата обращения: 06.06.2024). – Режим доступа: для авториз. пользователей. – С. 97.
3. Гриффитс Р. Д. Head First. Программирование для Android [Текст] / Р. Д. Гриффитс – Санкт–Петербург: Питер, 2016. – 704 с
4. Гусев, К. В. Технология разработки программных приложений : учебное пособие / К. В. Гусев, М. Б. Туманова, Е. А. Чернов. – Москва : РТУ МИРЭА, 2023. – ISBN 978–5–7339–1938–6. – Текст : электронный // Лань : электронно–библиотечная система. – URL: https://e.lanbook.com/book/382706 (дата обращения: 07.09.2024). – Режим доступа: для авториз. пользователей. – С. 4.
5. Дэрси Л. Разработка приложений для Android–устройств. Базовые принципы [Текст] /Л. Дэрси, Ш. Кондер – Том 1. – Москва: Эксмо, 2014. – 598 с.
6. Ескендир, М. Введение в разработку мобильных приложений [Текст]: научная статья. – Вестник магистратуры, 2019 – 4с.
7. Ефремова, О.С. Требования охраны труда при работе на персональных электронно–вычислительных машинах (ПК) [Текст]: пособие. – М.: Лабиринт, 2019 – 60 с.
8. Игошев, Б. ЭВТ: знакомимся, делаем, играем. / М. Галагузова, Д. Комский – М.:Молодая гвардия, 1989. – 156 с
9. Нейл, Т. Мобильная разработка [Текст]: справочник. – СПБ. 2013 – 208с.

10. Пак, Т. Разработка мобильных приложений [Текст]: учебное пособие. – СПБ: Санкт–Петербургский государственный университет аэрокосмического приборостроения, 2021 – 79с.

11. Руководства для разработчиков [Электронный ресурс]. – Режим доступа: https://developer.android.com/docs (дата обращения: 03.05.2024).

12. Филлипс, Б. Android. Программирование для профессионалов / Филлипс Б. – СПб: Питер, 2021. – 704 с.

13. Якоб Нильсен, Ралука Будиу. Как создавать идеально удобные приложения для мобильных устройств. 2013г.

14. Android Application Development All–in–One For Dummies / by John Wiley and Sons, Inc. – Canada. 2012 – 655 с.

15. Android Programming: The Big Nerd Ranch Guide / by Bill Phillips, Chris Stewart, Brian Hardy and Kristin Marsicano – 200 Arizona Ave NE. 2015 – 618 с.

16. Dart Apprentioce First Edition / by Jonathan Sade & Matt Gallowat, Copyright Razeware LLC. 2020 – 308 с.

17. Meier R. Professional Android 4 Application Development. Wrox Press, 2012. – 864 с.

18. Nash, J. Flutter for Beginners: An introductory guide to building cross–platform mobile applications with Flutter and Dart 2. – Packt Publishing, 2020. – 398 с.

19. Smaran, S. Flutter in Action. – Manning Publications, 2020. – 408 с.

20. Zammetti, F. Learn Flutter and Dart to Build iOS and Android Apps. – Apress, 2020. – 369 с.

21. ZEGOCLOUD. Get started with Video Call SDK – ZEGOCLOUD Doc [Электронный ресурс]. URL: https://www.zegocloud.com/docs/video–call/quickstart?platform=android&language=java (дата обращения: 05.06.2024).

Приложение А

Легенда к блок-схеме последовательности обработки голосового сообщения

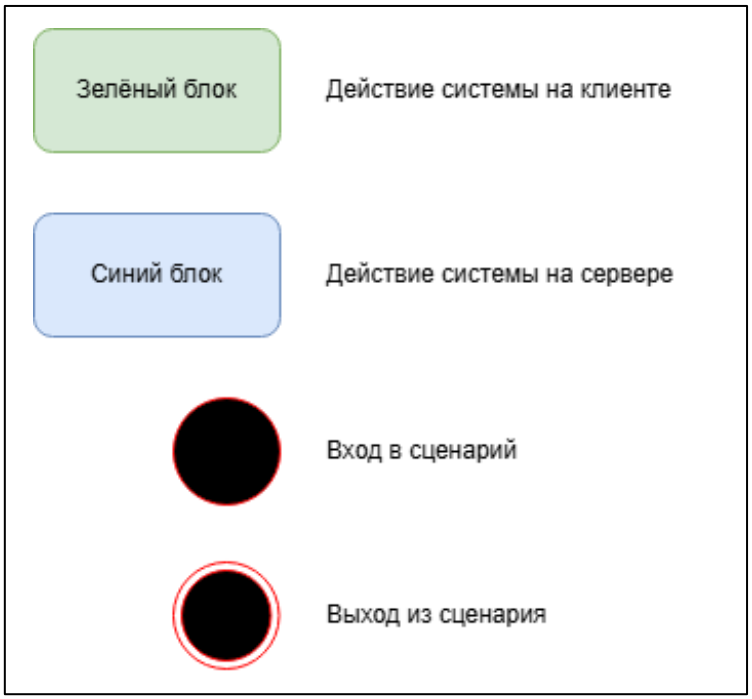


Рисунок А1 – Легенда к блок-схеме последовательности обработки голосового сообщения