Министерство образования Новосибирской области
ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

# Разработка мобильного приложения для видеоконференций

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения

для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.201200.010.000 ПЗ

Разработал:

студент группы ПР-23.106

Сборщиков Е.И.

2025

# Содержание

# Введение

В настоящее время на территории Российской Федерации происходит импортозамещение зарубежных программ с целью обеспечения граждан программными продуктами отечественного производства. На текущий момент проблемой такого решения является то, что разработка отечественных продуктов не настолько быстра, как процесс блокировки зарубежных ресурсов. В социальной сфере этот процесс затрагивает ресурсы, позволяющие обеспечивать надежную, быструю и качественную аудио- и видеосвязь между пользователями. Подобные ресурсы, как правило, обеспечивали связь между людьми и являлись технической составляющей социальных технологий осуществления дистанционного обучения.

Целью курсового проекта является разработка мобильного приложения для создания, управления и участия в видеоконференциях.

Для достижения этой цели поставлены следующие задачи:

– изучить предметную область;

– спроектировать UI- и UX-дизайн приложения;

– выбрать технологию разработки, язык и среду программирования;

– разработать базу данных, мобильное приложение, а также API для реализации запросов к REST и WebSocket контроллерам;

– протестировать UI-компоненты и программные модули приложения;

– настроить безопасность конфиденциальных данных пользователей;

– опубликовать в открытом доступе исполняемый файл приложения для установки на мобильные устройства.

# 1 Исследовательский раздел

## 1.1 Описание предметной области

Видеоконференция – это вид прикладного телевидения, обеспечивающий одновременно дуплексный режим обмена, обработки и представления информации на расстоянии в режиме реального времени с помощью вычислительной техники. Такая технология ранее использовалась в телефонии. В настоящее время видеоконференции используются для решения различного рода задач, таких как общение с людьми на расстоянии или осуществление занятий со студентами в дистанционном формате.

Основной особенностью видеоконференции является то, что обмен, обработку и представление информации можно производить не по сотовой связи, а по сети, используя протоколы передачи данных и сервер для маршрутизации потоков данных. Такую систему обслуживать дешевле, нежели осуществлять передачу информации между коммуникаторами по сотовой связи.

Помимо этого, у видеоконференций есть и другие особенности:

– организация мероприятий для дискуссий, без необходимости собрания людей в одном зале. Это может быть полезным для организаций, имеющих филиалы, расположенные в удаленных друг от друга регионах;

– возможность развертывания на мобильном устройстве. Такой подход позволяет пользователям не использовать для видеосвязи сложные технические устройства в виде компьютера или ноутбука, а за короткое время подключиться к комнате конференции с мобильного устройства;

– безопасность передачи данных по сети, осуществляемая встроенной защитой сетевых протоколов передачи данных и программной настройкой безопасности в исходном коде программы. Такой подход гарантирует, что при осуществлении видеоконференции, разговоры, записи с видеокамер

устройств и прочая информация не будет перехвачена и передана третьим лицам.

Подробная информация о требованиях к эксплуатации и безопасности системы представлена в техническом задании (См. Приложении А).

Большинство подобного рода приложений для мобильных устройств имеют клиент-серверную архитектуру. В такой модели имеется клиент, например – мобильное приложение, которое посылает запросы на сервер. Сервер обрабатывает запрос и при корректном запросе, при необходимости обращается к базе данных и возвращает клиенту ответ. В качестве ответа может быть коллекция данных, если посылаемый запрос был корректным, или сообщение об ошибке, если запрос был некорректным или на сервере произошла ошибка во время обработки запроса. Запросы к серверу могут различаться от типа выбранного API (англ. Application Programming Interface – Программный интерфейс приложения), который бывает следующих видов: SOAP, REST, WebSocket, RPC и GraphQL. Использование каждого из этих видов необходимо для решения различных задач:

– SOAP (англ. Simple Object Access Protocol – Простой протокол доступа к объектам) помогает доставлять системные сообщения от одного сервера к другому;

– REST (англ. Representational State Transfer – Передача репрезентативного состояния) оптимален в случаях, когда планируется реализация CRUD-логики (англ. – Create, Read, Update, Delete – Создание, чтение, редактирование, удаление);

– WebSocket чаще всего подходит для передачи потока данных в режиме реального времени, что достигается сохранением соединения открытым;

– RPC (англ. Remote Procedure Calls – Удаленный вызов процедур) позволяет передавать структуры данных, определенных в буфере протоколов, с возможность выбора типа взаимодействия: унарный (запрос-

ответ), потоковая передача данных от сервера, потоковая передача данных от клиента, дуплексная потоковая передача данных.

В проектируемой системе потребуется использования нескольких видов API: REST и WebSocket. Использование REST API позволит эффективно реализовать функционал, связанный с добавлением, редактированием, удалением и выводом данных, а WebSocket – для потоковой двунаправленной передачи данных внутри комнаты конференции.

Серверная сторона приложения включает в себя несколько составляющих:

– хранилище, где в зашифрованном виде будут храниться аутентификационные данные пользователей (email, пароль) и данные о каждой конференции;

– REST-контроллер, необходимый для совершения CRUD-операций пользователями. Этот компонент связан с базой данных;

– сигнальный сервер, регулирующий открытие и закрытие соединения, трансляцию данных, получение и передачу сообщений в бинарном формате;

– конечная точка соединения по WebSocket, необходимая приема полученной на сервер информации и последующая её широковещательная передача в режиме реального времени;

– конфигурация сервисов, определенная внутри Docker Compose, которая необходима для создания контейнера с пакетами и зависимостями, для развертывания на VDS (англ. Virtual Destroyed Server – Виртуальный выделенный сервер).

Всеми компонентами, которые входят в структуру серверной части приложения, необходимо управлять. Для выполнения этой задачи необходимы специалисты в области сетевого и системного администрирования, которые обладают необходимым набором знаний и владением компетенций, позволяющие администрировать работу программного и аппаратного обеспечения, что снижает риск поломки сервера и остановки работы API.

Клиентская часть системы – мобильное приложение, предоставляющее пользователям удобный, простой и минималистический интерфейс, не требующий специальных знаний по эксплуатации, а также функциональные возможности: регистрация, авторизация, создание, редактирование и удаление конференции, просмотр истории звонков, просмотр списка доступных конференций, вход и выход из комнаты конференции, настройка пользовательского интерфейса.

Исходя из вышесказанного, можно сделать вывод, что проектируемая система использует клиент-серверное взаимодействие, позволяющее работать с данными по сети, используя протоколы передачи информации и API в качестве связующего звена между клиентом и сервером.

В результате должно быть разработано мобильное приложение, позволяющее пользователям обмениваться информацией внутри комнаты конференции по сети, используя оптимальное количество функционала, для обеспечения качественной связи и снижения риска больших задержек передачи данных.

### 1.2 Образ клиента

Клиентами мобильного приложения являются зарегистрированные пользователи системы. Приложение предназначено для различных возрастов, но основной возраст пользователей приложения варьируется от 12 до 30 лет. Приложение будет удобно как для пользователей, которые общаются в группах, объединенных внутри комнаты конференции.

### 1.3 Сценарии

Предположим, что два друга детства находятся в различных городах в связи с обучением в лучших вузах Российской Федерации. Расстояние между городами достаточно большое, поэтому частые встречи довольно затратные в финансовом плане. Для решения такой социальной проблемы как дефицит общения оба друга могут воспользоваться приложением для звонков и

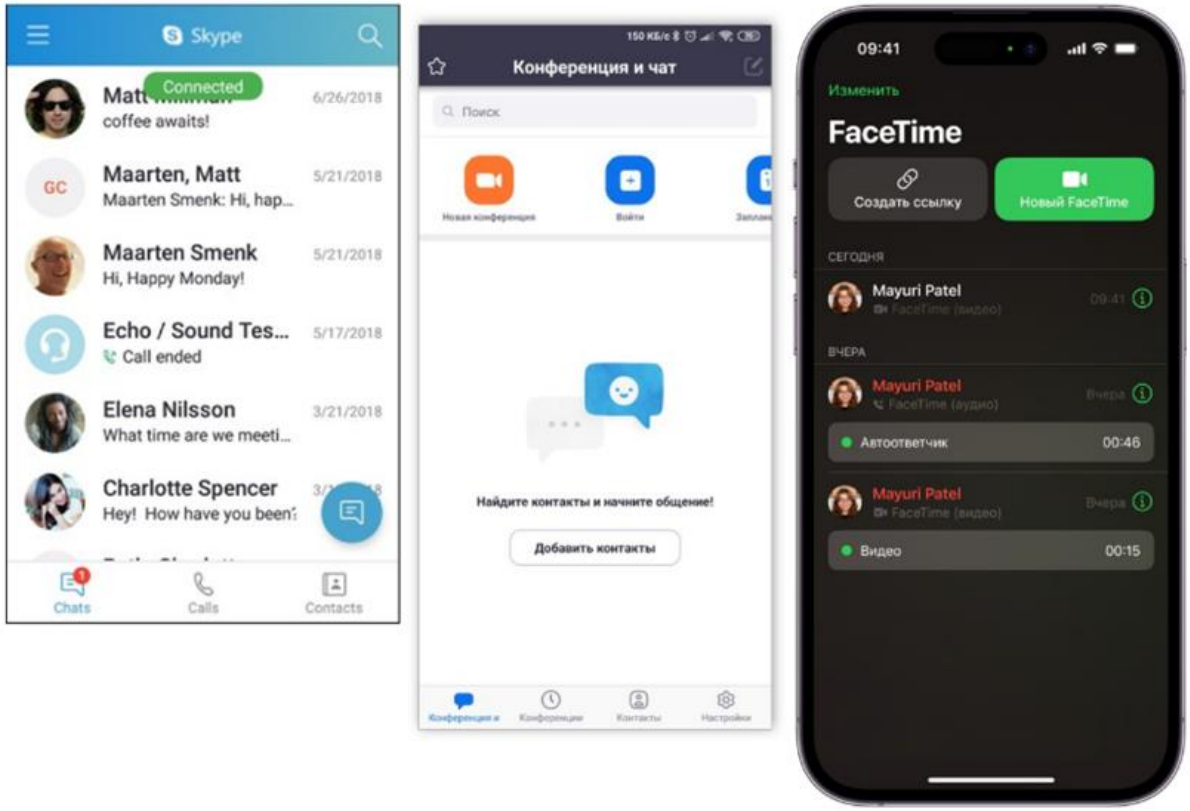
видеоконференций. С помощью приложения они могут поддерживать связь между собой и видеть лица друг друга в режиме реального времени. Для них также доступна опция приглашения других пользователей в конференцию, чтобы поддерживать общение и с ними, в независимости от местоположения или установленной операционной системы на их устройствах, поскольку приложение распространяется на мобильные операционные системы типа Android и IOS или WEB.

Можно привести и такой пример, что в здании учебного заведения идет ремонт и во время ремонта в помещении люди не должны находиться в здании. В таком случае преподавателям поставлена задача - организовать занятия дистанционно. Чтобы централизовать процесс дистанционного обучения в учебном заведении преподаватели могут создавать видеоконференции в приложении и приглашать в нее учеников или студентов для проведения лекционных, практических и лабораторных занятий.

### 1.4 Сбор и анализ прототипов

В магазинах приложений или на устройствах имеется различное программное обеспечение, позволяющее осуществлять аудио- и видео-звонки и видеоконференции для различных операционных систем. Примерами таких приложений могут служить Skype, Zoom, FaceTime.

Выбранные приложения имеют главный экран, на котором имеется некоторое исправление. Как правило, это может быть история звонков и кнопки, определяющие часть функциональности приложения. Это могут быть кнопки навигации по нижней панели или выдвижному меню, кнопки, предназначенные для администрирования конференций – создание, редактирование или удаление. Не исключена возможность использования кнопок, показывающих более подробную информацию о звонке. Такие кнопки часто располагаются в крайней правой части элемента списка выводной информации. Примеры прототипов приведены на рисунке 1.




Skype Zoom FaceTime

Рисунок 1 – Сравнение главных экранов приложений-прототипов

В верхней части у приложений расположены различные элементы навигации. В Skype располагается строка меню; в Zoom расположены три основные кнопки: новая конференция, войти, запланированные конференции; в FaceTime расположены кнопки: изменить, создать ссылку, новый FaceTime. Видно, что у всех приложений практически идентичны верхние панели управления.

Нижняя панель у приложений имеет различия. В FaceTime нижняя панель полностью отсутствует, а у Skype и Zoom они практически идентичны. В Skype на нижней панели расположены три кнопки: чаты, звонки, контакты. В Zoom — конференция, конференции, контакты и настройки.

Центральная часть у приложений практически схожа, там располагаются чаты или информация о звонках с собеседниками. Звонки в приложениях реализованы похожим образом. На экран выводиться

изображение аватара пользователя (аудио-звонок) или трансляция с камеры (видео-звонок). Сами экраны звонка выполнены в похожем стиле, но различия все же есть и они касаются больше пользовательского интерфейса, нежели функционала. Сравнение экранов звонка в приложениях представлено на рисунке 2.

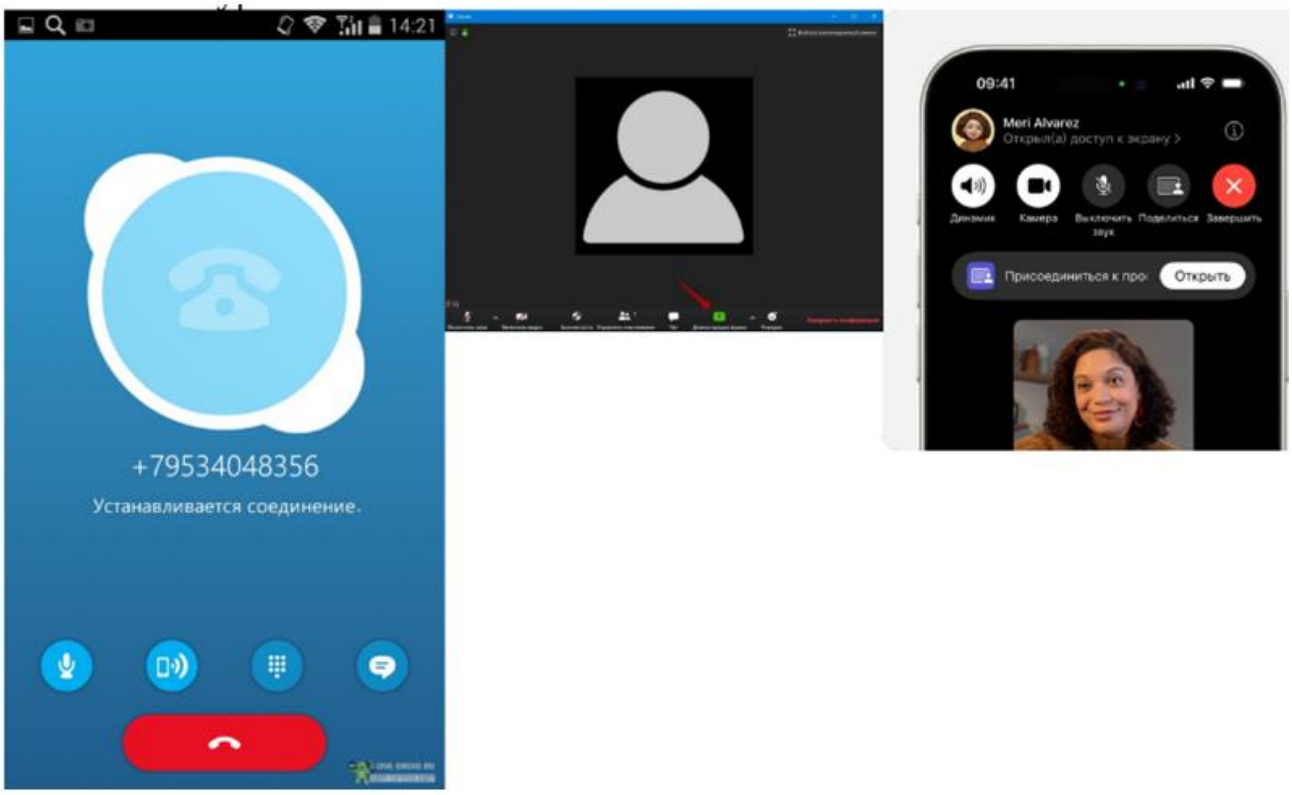



Рисунок 2 – Сравнение экранов звонка приложений-прототипов

Подводя итоги анализа прототипов, можно сделать вывод, что приложения схожи, но все же имеются различия как в пользовательском интерфейсе, так и в функциональности приложений. Более подробная информация о сравнении приложений приведена в таблице 1.

Таблица 1 – Сравнение приложений по основным критериям

| Критерии/Название | Skype | Zoom | FaceTime |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Стоимость | Бесплатное | Бесплатная базовая версия | Бесплатное |
| Платные услуги | Звонки на телефонные номера | Профессиональный и бизнес тарифы | Отсутствуют |

Продолжение таблицы 1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Основной язык | Английский | Английский | Английский |
| Аудио-звонки | Есть | Есть | Есть |
| Видео-звонки | Есть | Есть | Есть |
| Демонстрация экрана | Есть | Есть | Есть |
| Чаты с собеседниками | Есть | Есть | Отсутствуют |
| Расширение количества участников | Есть | Есть | Есть |
| Доступность | Кроссплатформенное | Кроссплатформенное | Проприетарное |

Рассмотрев несколько приложений в качестве прототипов, было решено разработать мобильное приложение для совершения аудио-, звонков и видеоконференций, которые будет содержать основные функции, иметь простой и понятный пользовательский интерфейс, а также иметь шифрование вызовов и обеспечивать не сильные затраты ресурсов физических компонентов мобильного устройства.

# 2 Проектирование приложения

## 2.1 UI/UX дизайн проекта

Для разработки дизайна приложения выбрано веб-приложение Figma, предоставляющее обширный функционал и множество плагинов для разработки дизайн-макета.

Для проекта определены экраны, позволяющие в мобильном приложении обеспечить визуальный интерфейс функционала регистрации, авторизации, просмотра истории звонков, поиска доступных конференций,, настроек интерфейса и комнаты видеоконференции. Каждый из экранов можно сгруппировать по характерным особенностям:

- аутентификация:
  1) регистрация,
  2) авторизация;
- основные экраны:
  1) история звонков,
  2) поиск доступных конференций,
  3) настройки интерфейса;
- комната конференции.

Для групп экранов созданы концепты компоновки интерактивных элементов, которые расположены на рисунке 3. Использование концептов дизайна экранов позволяет качественно выполнить разметку дерева виждетов, или интерактивных элементов, в коде приложения. Такой подход часто используется в современной разработке мобильных приложений, в основном в случаях, когда над проектом работает один или несколько, порядка трех или четырех, разработчиков. Это позволяет сделать дизайн масштабируемым и посредством наследования эффективно использовать для построения других, схожих экранов.

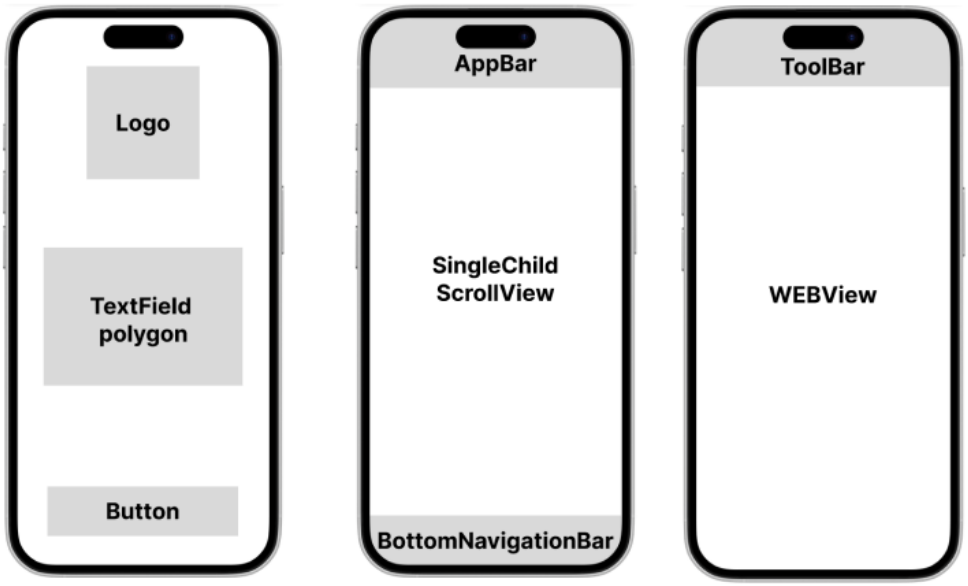



Рисунок 3 – Концепты построения элементов для каждой группы экранов

Слева расположен концепт для экранов регистрации и авторизации. Они имеют логотип сверху (Logo), элемент для определения текстовых полей ввода (TextField polygon), и кнопку в самом низу (Button). В середине расположен концепт, предназначенный для экранов истории звонков, поиска доступных конференций и настроек. В этом случае сверху расположена панель приложения (AppBar), где традиционно располагается название страницы, снизу расположена нижняя панель навигации (BottomNavigationBar), благодаря которой можно перемещаться между экранами. Основную часть занимает элемент просмотра прокрутки с одним дочерним элементом (SingleChildScrollView). Этот виджет эффективен, так как подходит для представления информации в виде списка, обеспечивая качественный пользовательский опыт и не уменьшая производительность. Справа расположен концепт экрана комнаты конференции. Основную часть экрана занимает элемент представления из WEB (WEBView), на котором располагается динамическая информация в режиме реального времени. Для управления состоянием вверху находится элемент панели инструментов (ToolBar).

Приложение имеет две темы оформления: светлую и темную. Цветовые схемы обеих тем представлены на рисунках 4 и 5 соответственно.


Рисунок 4 – Цветовая схема светлой темы

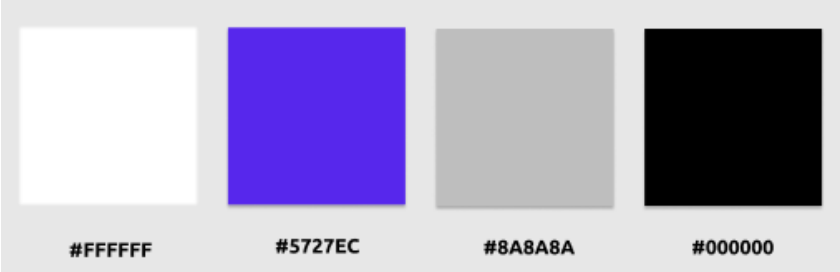



Рисунок 5 – Цветовая схема тёмной темы

Цветовая схема логотипа и логотип приложения представлены на рисунках 6 и 7 соответственно.

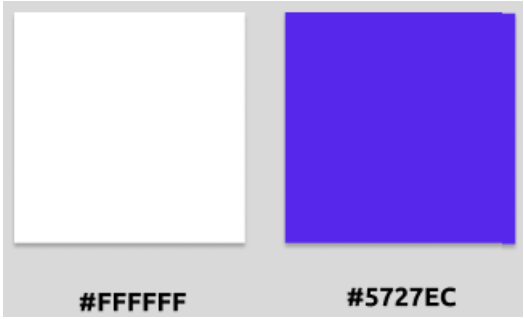



Рисунок 6 – Цветовая схема логотипа приложения

Рисунок 7 – Логотип приложения

Дизайн основных экранов приложения в светлой теме оформления и нативного интерфейса AndroidOS представлен на рисунке 8.

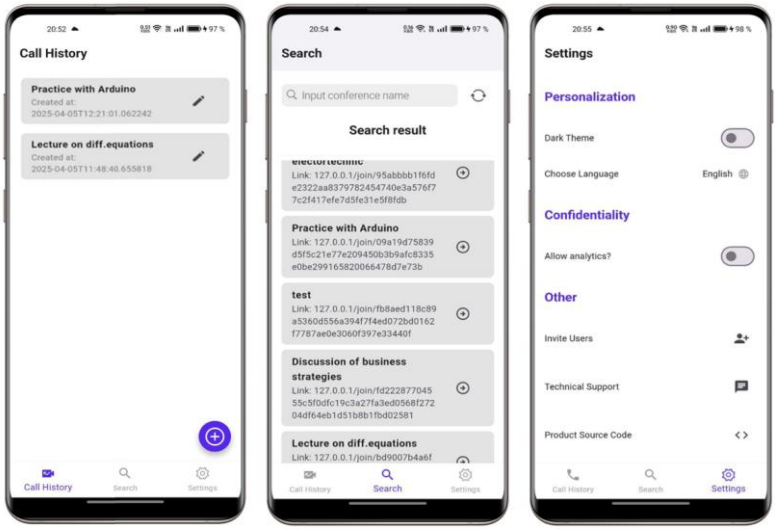



Рисунок 8 – Дизайн-макет приложения в светлой теме

Дизайн основных экранов приложения в темной теме оформления и нативного интерфейса iOS представлен на рисунке 9.

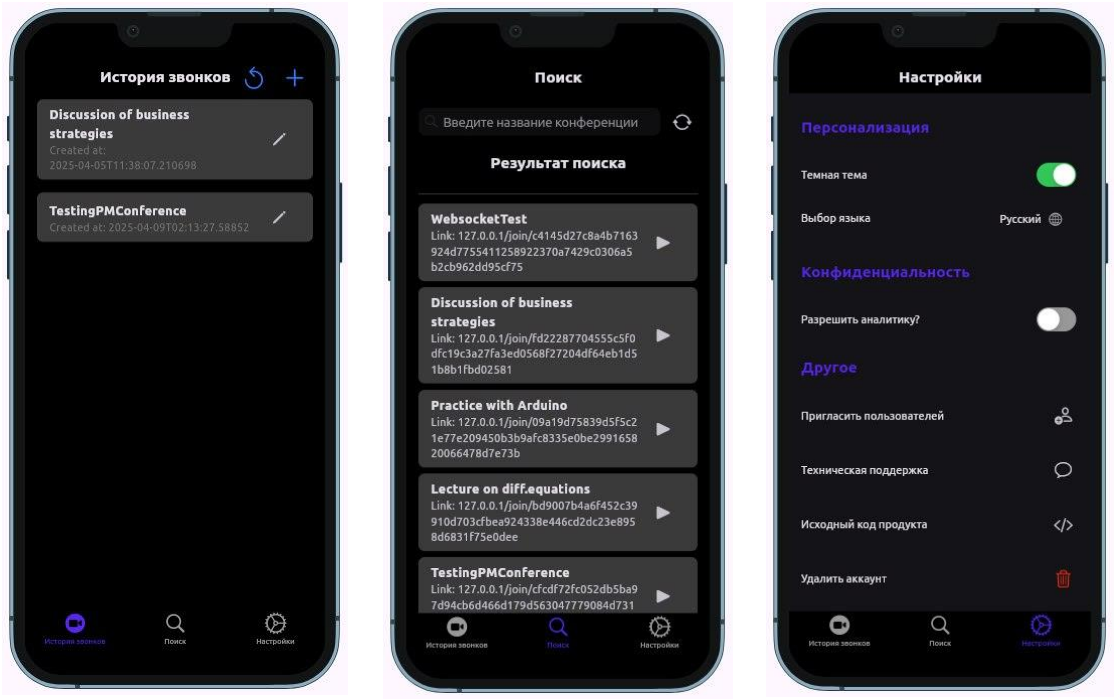



Рисунок 9 – Дизайн приложения в темной теме оформления

При запуске приложения пользователь видит экран регистрации (См. рисунок 10). На этом экране пользователь может ввести данные адреса электронной почты (по шаблону example@<...>.<...>) и пароль, который

нужно также подтвердить в соответствующем поле ввода. На экране авторизации подтверждение пароля не требуется. Оба экрана идентичны для платформ Android и iOS. Внешний вид расположен на рисунке 10.

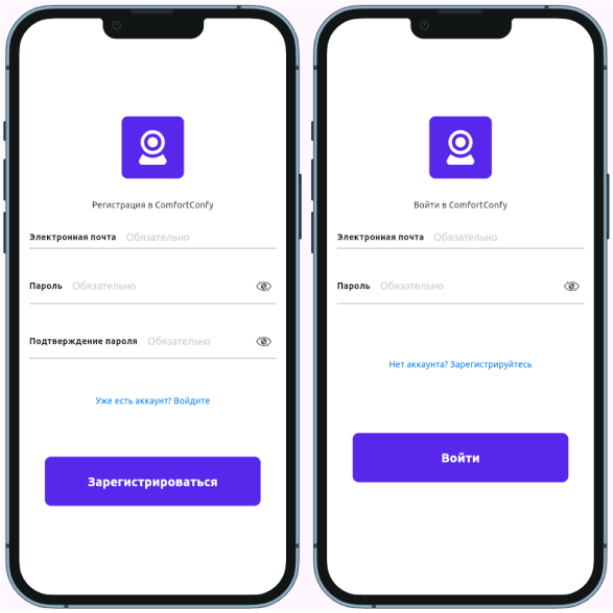



Рисунок 10 – Экраны регистрации и авторизации

После прохождения процедуры регистрации или авторизации, пользователь попадает на экран истории звонков, расположенный на рисунке 11.

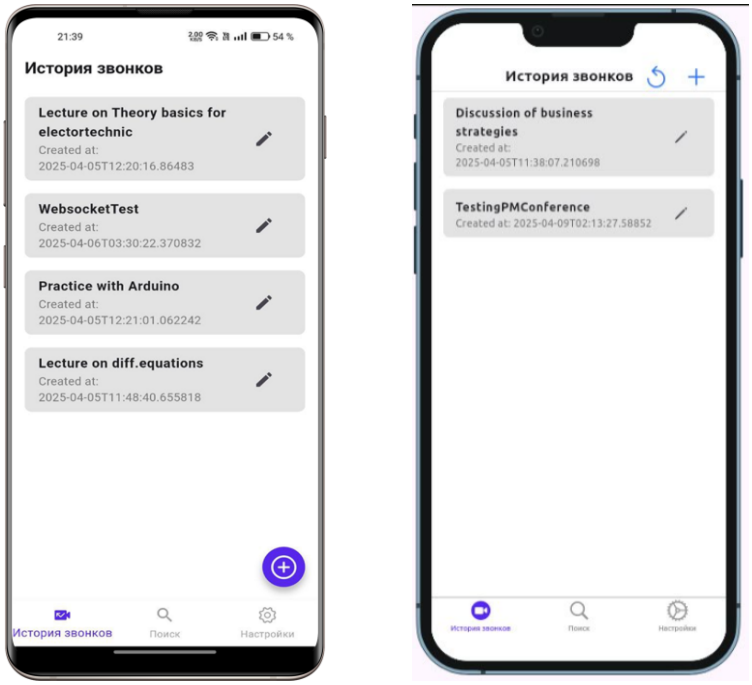



Рисунок 11 – Экран истории звонков для платформ Android и iOS

Навигация между другими экранами происходит по нижней навигационной панели (для Android – BottomNavigtionBar, для iOS – TabBar). Внешний вид представлен на рисунке 12.

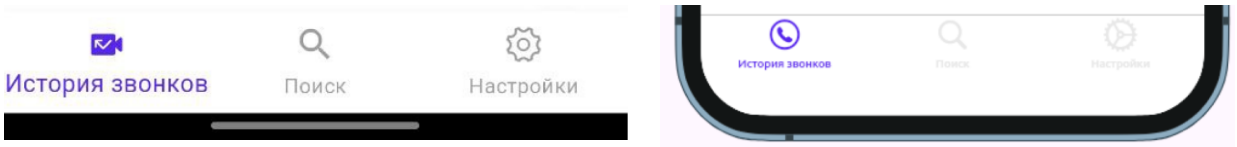



Рисунок 12 – Нижняя навигационная панель для платформ Android и iOS

Этот виджет находится на странице, которая служит каркасом для перехода к другим экранам как по кнопкам на панели, так и с помощью свайпа.

На экране истории звонков отображается информация о ранее созданных пользователем конференциях. Также на этом экране можно создать конференцию. На Android – с помощью FloatActionButton, расположенной внизу экрана, на iOS – с помощью кнопки «+», расположенной на верхней панели справа от названия экрана. Внешний вид представлен на рисунке 13.

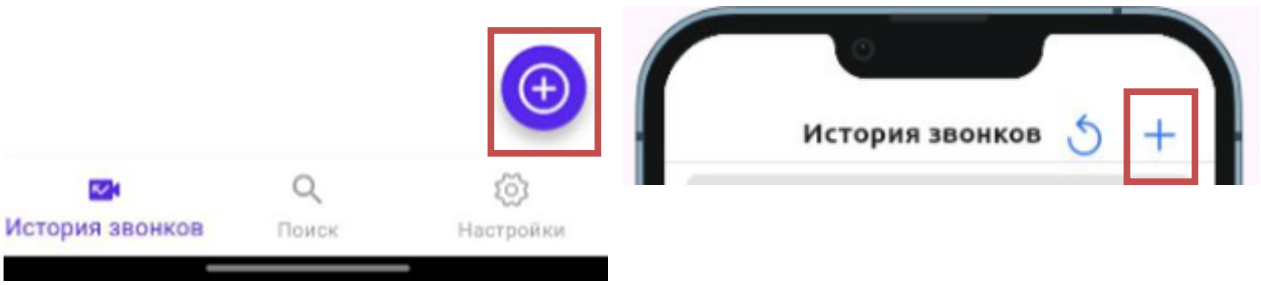



Рисунок 13 – Интерфейс для создания конференции

После создания конференции сформируется карточка на экране истории звонков, и конференция будет доступна на экране поиска. На карточке конференции отображена информация о названии конференции и дате её создания, а также расположена кнопка в виде иконки карандаша, при нажатии на которую можно выбрать опцию: изменить название конференции или удалить конференцию. Внешний вид графической модели карточки расположен на следующей странице (См. рисунок 14).

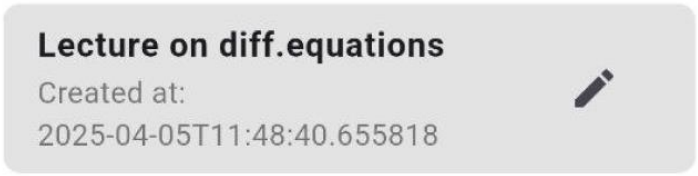



Рисунок 14 – Карточка конференции и кнопка опций

На экране поиска конференций пользователь в текстовое поле может ввести название конференции, или часть названия. В зависимости от переданного слова пользователь увидит список конференций, с названиями, содержащие переданную часть слова. При нажатии на элемент списка, пользователь может перейти в эту комнату и принять участие в конференции. Чтобы получить данные о доступных конференциях, необходимо нажать на кнопку с иконкой перезапуска, расположенной вверху рядом с полем для ввода названия конференции. Дизайн экрана на всех платформах идентичен. Внешний вид расположен на рисунке 15.

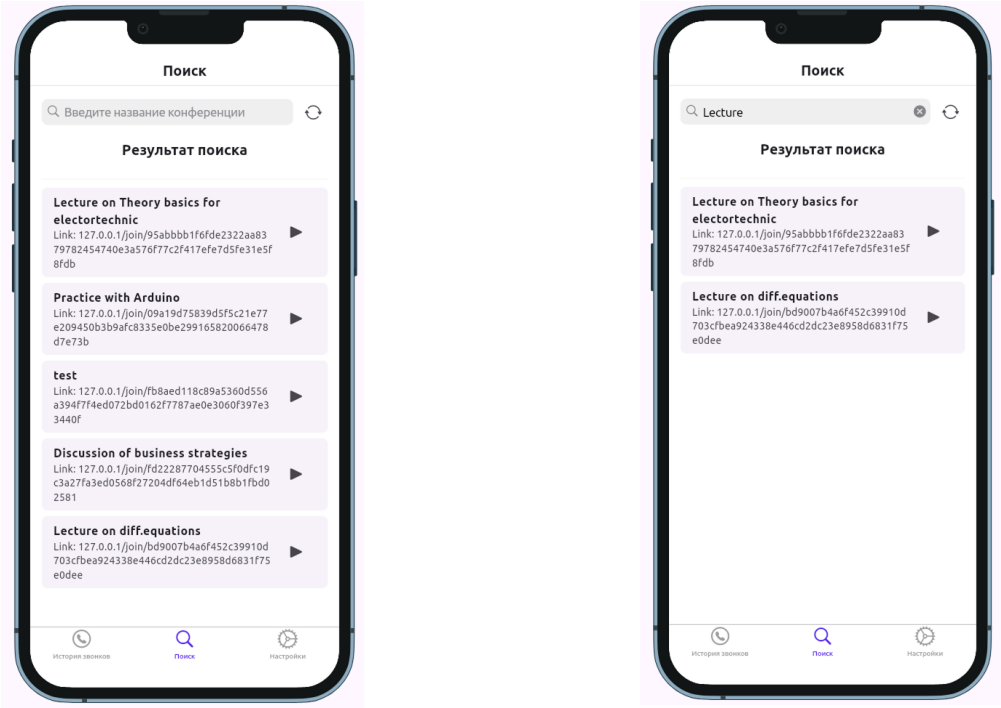



Рисунок 15 – Вывод всех доступных конференций на экране поиска и вывод доступных конференций по переданной части слова

На экране настроек пользователь может персонализировать интерфейс, настроить конфиденциальность и использовать прочие опции. Здесь

присутствуют нативные виджеты и их отображение зависит от выбранной платформы. На следующей странице представлены рисунки 16 и 17 с внешним видом экрана настроек и отображением темной темы для платформ Android и iOS соответственно.

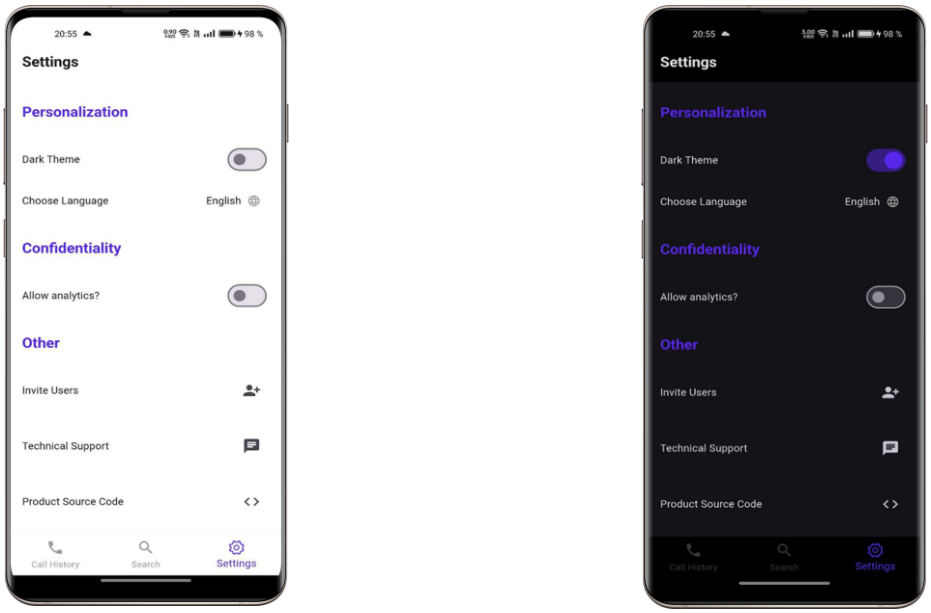



Рисунок 16 – Опции на экране настроек и включенная темная тема для платформы Android

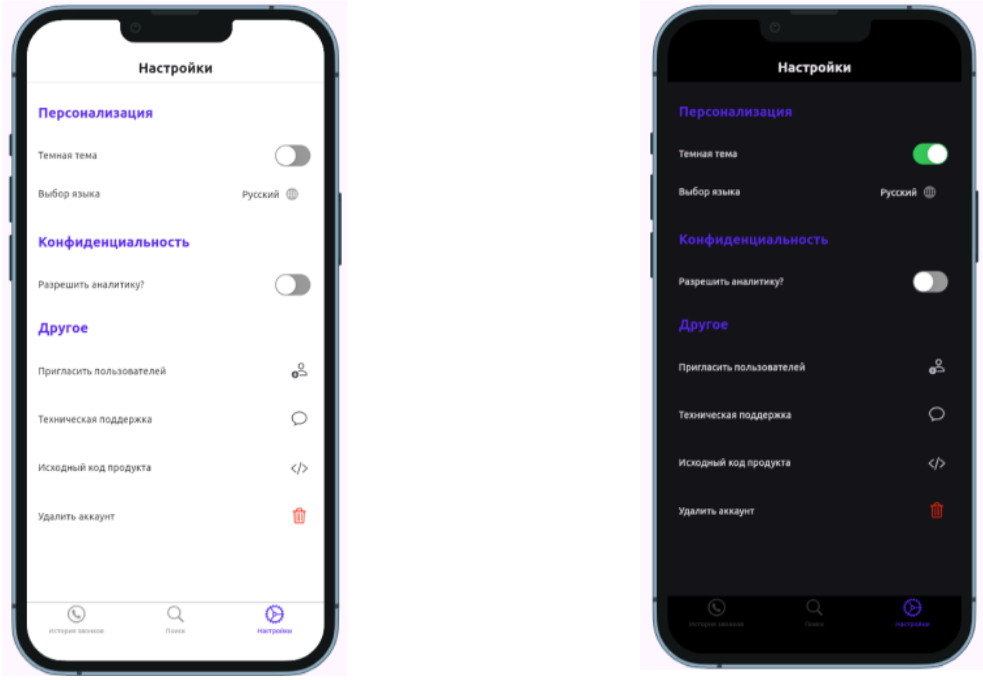



Рисунок 17 – Опции на экране настроек и включенная темная тема для платформы iOS

На рисунке 18 изображен внешний вид панели инструментов, который идентичен как для Android, так и для iOS.

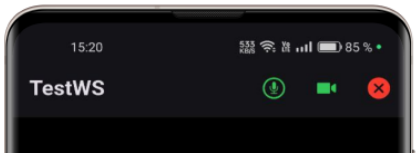



Рисунок 18 – Панель инструментов

На остальной части экрана комнаты конференции отведено место под отображение всех участников.

### 2.2 Выбор технологии, языка и среды программирования

Перед началом разработки программного продукта необходимо определится с технологиями и программными ресурсами, поскольку эти аспекты реализации играют очень важную роль. Исходя из предметной области, приложение имеет две основные составляющие - мобильное приложение, серверное приложение.

Для разработки мобильного приложения выбрана технология кроссплатформенной разработки Flutter. Она примечательна тем, что позволяет запустить разработанный программный продукт на таких системах как Android, IOS, Windows, MacOS (OS X), дистрибутивах Linux и в веб-браузере. При сборке проекта исполняемый файл переводится с языка исходного кода в Assembly Code. При создании проекта, автоматически генерируются конфигурационные файлы и для каждой операционной системы, обращаются к переведённому исполняемому файлу программы и генерируют исполняемые файлы для каждой системы (.apk/.aab – Android, .ipa – IOS, .exe – Windows, .exfat – MacOS, .sh – дистрибутивы Linux). Это облегчает процесс публикации проекта в магазины приложений, так как с помощью одной команды в терминале можно сгенерировать исполняемые файлы для каждой системы.

В качестве языка программирования для реализации мобильного приложения был выбран язык dart, который поставляется совместно с Flutter. Dart имеет следующие преимущества:

– схожий синтаксис с языками Java и C#;

– использует объектно-ориентированный подход к программированию и асинхронность;

– является проектом с открытым исходным кодом;

– поддерживается браузерами;

– обеспечивает безопасность типов данных, гибкую и быструю компиляцию;

Для разработки серверного приложения был выбран язык программирования Python, а также технологии FastAPI и WebSockets. Этот выбор обоснован тем, что данный язык программирования имеет очень простой синтаксис, что позволяет затрачивать меньше времени на разработку сервера. Выбор технологий обоснован в связи с тем, что в мобильном приложении необходимо использовать оба контроллера: REST API(FastAPI) – создание, просмотр списка конференций, истории звонков, вход и выход из конференции; WebSocket – потоковая передача информации на сервер для дальнейшей обработки направление трансляции обработанной информации пользователям в комнату конференции.

Для работы с аутентификацией пользователей и хранилищем данных был выбран сервис Supabase. Этот выбор обоснован тем, что при создании проекта можно выбрать сервер, где будет располагаться проект. Также, SupabaseAPI поддерживает шаблоны методов для аутентификации пользователей, которые легко интегрируются в проект мобильного приложения и хранилище данных, с которым также можно взаимодействовать из мобильного приложения на основе CRUD-логики.

# 3 Разработка мобильного приложения

## 3.1 Взаимодействие с API сервисом

В мобильном приложении взаимодействие с API сервисом происходит с использованием Future-методов. Это способ описание методов класса с использованием асинхронного программирования, являющимся экземпляром класса Future<T>, который в свою очередь является объектом, представляющим отложенное вычисление. С помощью таких методов можно описать и интегрировать с сервера в мобильное приложение все типы запросов. Пример такого метода расположен ниже:

```
/// <Summary>
/// SignIn with email & password
/// </Summary>
Future<AuthResponse> signInWithEmailPassword(String email, String password)
async{
  return await _supabase.auth.signInWithPassword(
    email: email,
    password: password
  );
}
```

В этом методе используется ссылка на тип данных в виде Response-модели для авторизации пользователей. Метод является асинхронным, поскольку указано ключевое слово async. В качестве возвращаемого значения (на это указывает молификатор return await) ожидаются параметы метода auth.signInWithPassword класса SupabaseClient, к которому идет обращение через экземлпяр _supabase этого класса.

## 3.2 Разработка базы данных

Для разработки базы данных был использован сервис Supabase. В проекте хранилище организовано следующим образом: создаются таблицы, содержащие информацию о пользователях и о видеоконференциях. Также создается связующая таблица, которая содержит внешние ключи на основные

таблицы. Для хранения данных в защищенном виде используется хэширование данных по стандарту SHA256. Диаграмма хранилища представлена на рисунке 19.

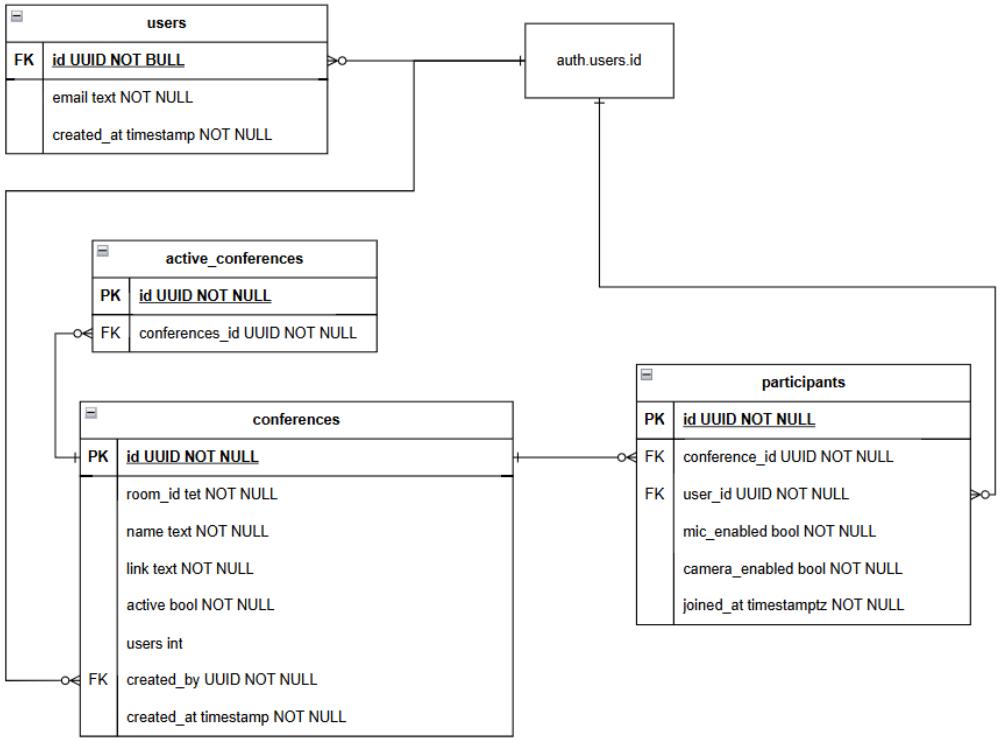



Рисунок 19 – Диаграмма базы данных

Такая модель базы данных позволяет оптимально организовать хранение данных и доступ к ним. В ней соединены как реляционные отношения (conferences, active_conferences, participants), так и документы данных (auth.users.id).

### 3.3 Разработка мультимедийного контента

Клиентский компонент (мобильное приложение) разработано с использованием фреймворка Flutter, поддерживающий язык программирования Dart. Также создан и серверный компонент, который позволяет обеспечить безопасное хранение данных и доступ к ним.

Вёрстка экранов мобильного приложения проводилась с использованием виджетов, поскольку такой подход предписан выбранной технологией разработки программного обеспечения. Собственные виджеты, как самостоятельные, так и являющиеся частями экранов были созданы с

использованием архитектурного паттерна MVC (англ. Model-View-Controller – Модель-Представление-Контроллер). Такой подход полезен для адаптации элементов вёрстки и масштабирования приложения. Шаблон MVC представляет собой модель данных, реализуемой в виде публичных или абстрактных классов данных, представления – импортированных на экран виджетов и самого экрана и контроллеров – инструменты связки клиента и сервера посредством виджетов.

Мультимедийный контент приложения представляет собой кнопки, иконки, темы приложения, тексты и картинки. В проекте все картинки хранятся в каталоге /assets, который находится на уровне корневого каталога. Это обеспечивает правильный и мгновенный доступ к элементам каталога, к которым существует обращение при вёрстке экранов приложения.

Общие интерактивные элементы, такие как кнопки, текстовые элементы, панель приложения и нижняя навигационная панель и многие другие, хранятся в каталоге с общих виджетов, а виджеты, применяемые к конкретному экрану и не более – расположены в соответствующих подкаталогах внутри /features.

В каталоге /themes хранятся конфигурационный файл с цветами тем приложения, а также провайдер тем. Провайдер необходим для переключения со светлой темы на тёмную и наоборот.

Иконки, используемые в приложении, не требуют хранения в каталоге /assets или /components, поскольку Flutter предоставляет доступ к репозиторию, где хранится большое количество иконок для различных платформ.

### 3.4 Описание используемых плагинов

Для разработки системы были использованы некоторые сторонние плагины как на клиентской, так и на серверной стороне. В таблицах 2 и 3 указаны названия и описания каждого используемого плагина соответственно.

Таблица 2 – Описание используемых плагинов для разработки клиентской части приложения

| Название | Описание |
|---|---|
| flutter: ^3.6.0 | Используемая версия фреймворка для разработки кроссплатформенных приложений |
| supabase_flutter: ^2.8.4 | Плагин для интеграции Supabase с Flutter, позволяет работать с базой данных Supabase из приложения Flutter. |
| provider: ^6.0.0 | Плагин для управления состоянием приложения, упрощает обмен данными между виджетами. |
| url_launcher: ^6.0.20 | Позволяет запускать внешние приложения по гиперссылкам из приложения Flutter. |
| clipboard: ^0.1.3 | Плагин для работы с буфером обмена, позволяет копировать и вставлять текст. |
| cupertino_icons: ^1.0.6 | Набор иконок в стиле Cupertino. |
| flutter_svg: ^2.0.9 | Плагин для отображения SVG-графики в приложениях Flutter. |
| device_preview: ^1.2.0 | Позволяет предварительно просматривать макет приложения на разных устройствах и ориентациях внутри редактора кода. |
| web_socket_channel: ^2.1.0 | Плагин для работы с веб-сокетами, позволяет устанавливать двустороннюю связь между клиентом и сервером. |
| camera: ^0.11.1 | Плагин для работы с камерой устройства, позволяет делать фотографии и записывать видео. |
| image: ^3.0.1 | Плагин для обработки изображений, позволяет выполнять различные операции над изображениями. |
| permission_handler: ^10.2.0 | Плагин для управления разрешениями приложения, позволяет запрашивать и проверять права доступа к различным функциям устройства. |
| flutter_localizations | Встроенный пакет Flutter для локализации приложений. |
| intl: any | Плагин для поддержки интернационализации и локализации, позволяет форматировать даты, числа и текст в соответствии с языком и регионом пользователя. |
| flutter_test | Встроенный пакет для тестирования приложений Flutter. |
| flutter_lints: ^3.0.0 | Плагин для анализа кода и проверки его качества, помогает выявлять потенциальные ошибки и улучшать стиль кода. |
| intl_utils: ^2.0.0 | Плагин для расширения возможностей пакета intl, добавляет дополнительные функции для работы с датами и числами. |
| build_runner: ^2.3.0 | Плагин используемый пакетами, требующими генерации кода на основе аннотаций или других метаданных. |

Таблица 3 – Описание используемых плагинов для разработки серверной части приложения.

| Название | Описание |
|---|---|
| 1 | 2 |
| python = "^3.10" | Версия Python, используемая в проекте. Python 3.10 включает такие нововведения, как структурное сопоставление шаблонов, улучшения в производительности и новые функции для работы с типами данных. |

Продолжение таблицы 3

| 1 | 2 |
|---|---|
| supabase = "^2.13.0" | Библиотека для взаимодействия с Supabase, которая предоставляет функции для работы с базой данных, аутентификацией и другими сервисами Supabase в приложениях Python. |
| fastapi = "^0.115.11" | Фреймворк для создания веб-API, известный своей скоростью и простотой использования. FastAPI поддерживает асинхронные запросы и автоматическую генерацию документации API. |
| uuid = "^1.30" | Библиотека для генерации уникальных идентификаторов (UUID), которые часто используются для идентификации объектов в базе данных или других системах. |
| asyncio = "^3.4.3" | Встроенная библиотека Python для работы с асинхронным программированием. Asyncio позволяет писать однопоточные коды, которые могут выполнять несколько задач одновременно. |
| uvicorn = "^0.34.0" | ASGI-совместимый веб-сервер, используемый для запуска приложений, написанных на FastAPI или других фреймворках, поддерживающих ASGI. Uvicorn обеспечивает высокую производительность и поддержку WebSocket. |
| pydantic-settings = "^2.8.1" | Библиотека, основанная на Pydantic, для управления настройками приложения. Она позволяет легко загружать и валидировать конфигурационные данные из различных источников. |
| python-dotenv = "^1.1.0" | Библиотека для загрузки переменных окружения из файла .env. Это позволяет сохранять конфиденциальные данные вне кода. |

Использование вышеперечисленных плагинов позволяет качественно настроить работу REST- и WebSocket контроллеров, обеспечить безопасность передачи, получения по сети и хранения данных в базе, а также эффективно использовать пространство имён.

### 3.5 Описание разработанных процедур и функций

В системе разработанные процедуры и функции присутствуют как на серверной стороне, где определены и описаны маршруты REST API- и WebSocket контроллеров, так и на клиентской стороне, где присутствуют методы для связи с API, а также отображения на экране нативных UI-элементов.

В таблице 4 приведено описание разработанных процедур и функций, определённых в исходном коде серверной части.

Таблица 4 – Описание разработанных процедур и функций для серверной части

| Название | Описание |
|---|---|
| 1 | 2 |
| get_supabase | Функция подключения к хранилищу данных Supabase. Для подключения использует параметры SUPABASE_URL и SUPABASE_KEY, записанные в .env |
| hashed_room_id | Функция для хеширования идентификатора комнаты конференции. Принимает как параметр сгенерированный идентификатор комнаты в формате UUID и возвращает хэш идентификатора |
| create_conference | Асинхронная функция для выполнения POST запроса на создание конференции. В качестве входных параметров использует модель запроса (название конференции и идентификатор создателя) и экземпляр класса Supabase. Функция предназначена для создании записи о новой конференции и сохранения записи в базу данных |
| update_conference_name | Асинхронная функция для изменения названия выбранной конференции. В качестве входных параметров принимает модель запроса (идентификатор комнаты и новое название) и экземпляр Supabase. Функция предназначена для обновления данных в поле названия конференции существующей записи |
| delete_conference | Асинхронная функция для удаления выбранной конференции. В качестве входных параметров принимает идентификатор комнаты конференции и экземпляр Supabase. Функция предназначена для каскадного удаления данных о конференции из базы данных |
| join_conference | Асинхронная функция для присоединения в конференцию. В качестве входных параметров принимает идентификатор комнаты конференции и экземпляр Supabase. Функция является счетчиком, которая увеличивает количество пользователей в выбранной конференции |
| leave_conference | Асинхронная функция для выхода из конференции. В качестве входных параметров идентификатор комнаты конференции и экземпляр Supabase. Функция является счетчиком, которая уменьшает количество пользователей в выбранной конференции и, если пользователей не осталось – конференция становиться не активной. |
| list_conferences | Асинхронная функция для вывода данных о доступных конференциях. Также эта функция используется для вывода данных на экран истории звонков, используя необязательный параметр с идентификатором создателя |
| connect | Метод, предназначенный для добавления соединения по WebSocket-протоколу и создания задачи комнате конференции |
| disconnect | Метод, предназначенный для остановки соединения |
| broadcast | Асинхронный метод доставки сообщения в очередь, а не отправки их непосредственно |
| send_message | Асинхронный метод является отдельной задачей и запускается для каждой комнаты для обработки отправки сообщений с задержкой |

Продолжение таблицы 4

| 1 | 2 |
|---|---|
| send_message | Асинхронный метод является отдельной задачей и запускается для каждой комнаты для обработки отправки сообщений с задержкой |
| _safe_send | Асинхронный метод с приватным модификатором доступа, предназначенный для обработки ошибок отправки сообщений |
| websocket_endpoint | Асинхронная функция, являющаяся конечной точкой входа для обмена сообщениями в реальном времени |

Все эти разработанные процедуры на серверной стороне системы обеспечивают качественную работу REST- и WebSocket-контроллеров.

В таблице 5 приведено описание разработанных процедур и функций, использующихся на клиентской стороне системы – в мобильном приложении.

Таблица 5 - Описание разработанных процедур и функций для клиентской части

| Название | Описание |
|---|---|
| 1 | 2 |
| signInWithEmailPassword | Асинхронный метод, предназначенный для авторизации в приложении по email и паролю. Возвращает токен пользователю, посредством которого можно пользоваться приложением в рамках сессии |
| signUpWithEmailPassword | Асинхронный метод, предназначенный для регистрации в приложении по email и паролю. Корректные данные записываются в базу данных, и возвращается токен, посредством которого можно пользоваться приложением в рамках сессии |
| signOut() | Асинхронный метод для выхода из приложения и завершения сессии |
| getCurrentUserEmail | Метод для получения email выбранного пользователя, необходимый для соответствия пользователя внутри сессии |
| androidCreateConference | Асинхронная функция для демонстрации нативного модального окна Android с виджетами для создания конференции |
| iosCreateConference | Асинхронная функция для демонстрации нативного модального окна iOS с виджетами для создания конференции |
| showConferenceOptions | Асинхронная функция, которая содержит в своем определении обращения к процедурам, демонстрирующим нативные элементы управления свойствами конференции, таким как изменение названия конференции или удаление конференции |

Продолжение таблицы 5

| 1 | 2 |
|---|---|
| platformDeleteAccount | Асинхронная функция, демонстрирующая нативные виджеты с функционалом для удаления аккаунта пользователя |
| _showAndroidDialog | Приватный метод для демонстрации модального окна с нативными виджетами Android для смены языка интерфейса |
| _showCupertinoDialog | Приватный метод для демонстрации модального окна с нативными виджетами iOS для смены языка интерфейса |
| _onLanguageSelected | Приватный метод для смены языка интерфейса по выбранному языку (русский или английский) |
| _initializePermissions | Приватный асинхронный метод для инициализации предоставленных разрешений к датчикам микрофона и камеры |
| _initializeWebsocket | Приватный метод инициализации соединения по протоколу WebSocket |
| _handlePermissionDenied | Приватный метод обработки запроса разрешений |
| _handleWebSocketError | Приватный метод обработки ошибок соединения по протоколу WebSocket |
| _initializeCamera | Приватный асинхронный метод для инициализации датчика камеры |
| _initializeMicrophone | Приватный асинхронный метод инициализации датчика микрофона |
| _parseParticipants | Вывод карточек участников видеоконференции |
| _toggleMic | Приватный метод управления активностью микрофона |
| _toggleCamera | Приватный метод управления активностью камеры |
| _copyToClipboard | Приватный метод для копирования в буфер обмена ссылки для приглашения пользователей в приложение |
| _launchURL | Приватный метод для открытия ссылки в браузере. Этот метод необходим, чтобы пользователи смогли просмотреть открытый исходный код приложения |
| _loadTheme | Приватный асинхронный метод для загрузки коллористики темы |
| ToggleTheme | Асинхронный метод для переключения между темами |
| ToggleAnalytics | Асинхронный метод включения/отключения опции аналитики использования приложения |

Все эти методы разработаны с целью взаимодействия с API сервисом, управления датчиками мобильного устройства, а также корректным отображением нативных UI-элементов в приложении.

# 4 Тестирование

## 4.1 Протокол тестирования дизайна приложения

Для проведения тестирования нативного для Android дизайна приложения было выбрано устройство Realme 11 с разрешением 2400x1080 пикселей.

Проверка происходила с целью определения корректности отображения виджетов и экранов по следующим параметрам:

– оптимальный размер иконок под различные виды тем оформления на уровне операционный системы;

– читабельный размер и цвет шрифта;

– корректное отображение элементов.

Тестирования дизайна было пройдено успешно.

## 4.2 Протокол тестирования функционала приложения

Тестирования функционала приложения распределено на несколько этапов. Сначала тестируется маршрутизация на REST-контроллере, а затем – функционал передачи и получения информации по WebSocket. Для тестирования API используется утилита Postman, которая предполагает обширный функционал для тестирования каждого вида API. Для проверки работоспособности API, проведены следующие тесты:

– создание конференции;

– вывод списка истории звонков;

– изменение названия конференции;

– вывод списка доступных конференций;

– присоединение к комнате конференции;

– соединение по WebSocket и передача данных;

– выход из комнаты конференции;

– удаление конференции из базы данных.

Результаты каждого теста продемонстрированы на рисунках 20-26, расположенных в вышеупомянутой последовательности.

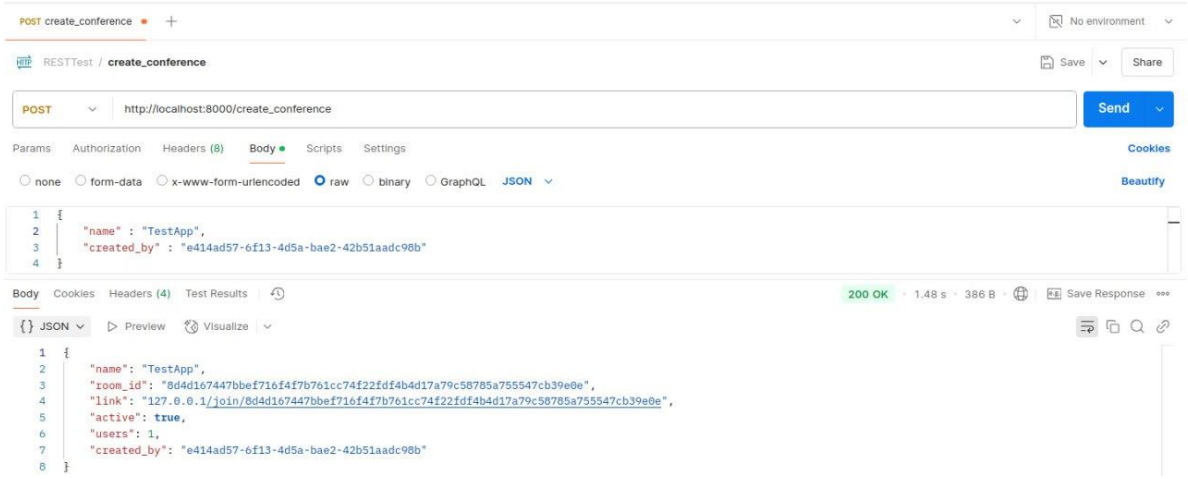



Рисунок 20 – Тестирование запроса на создание конференции

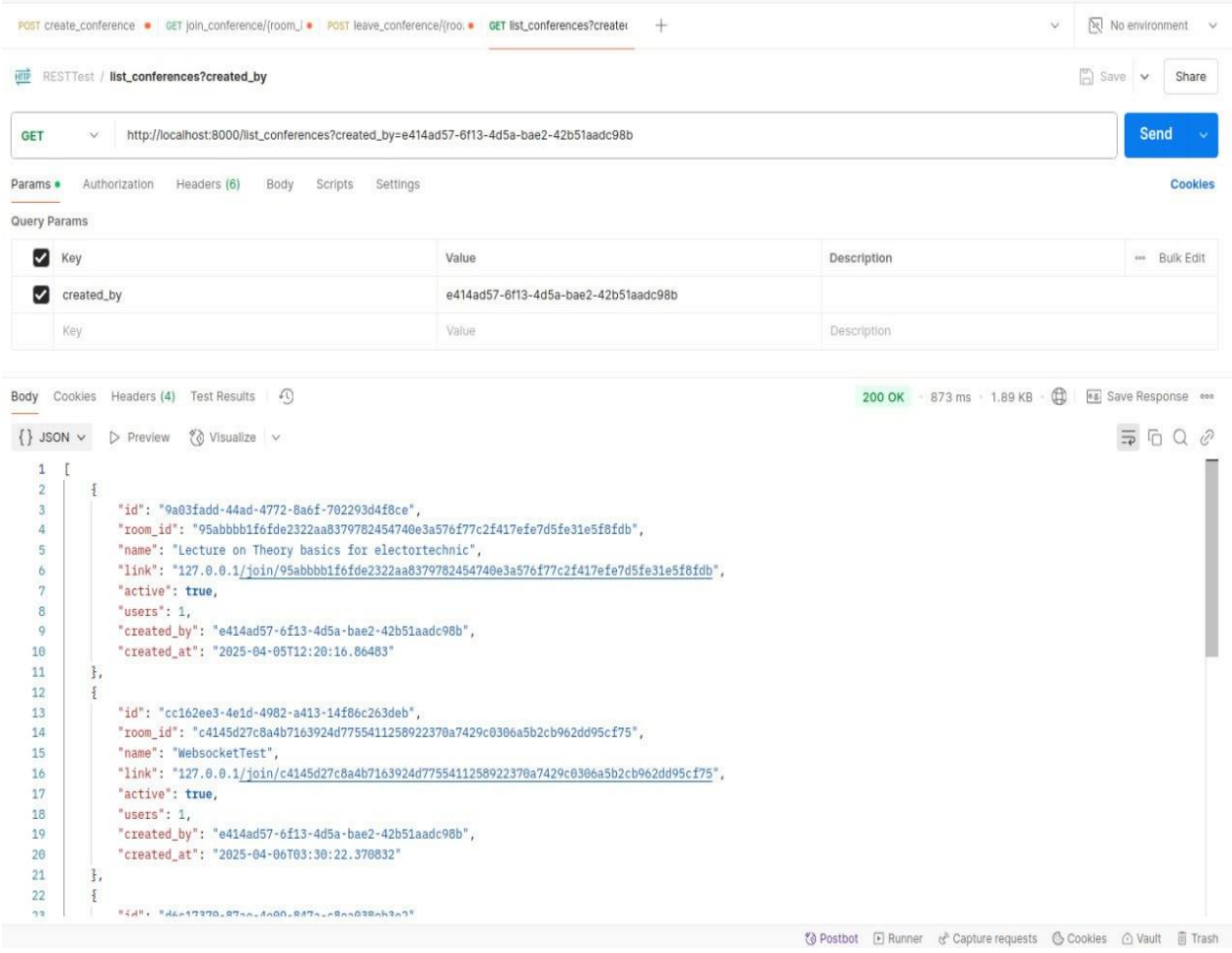



Рисунок 21 – Тестирование запроса на вывод истории звонков

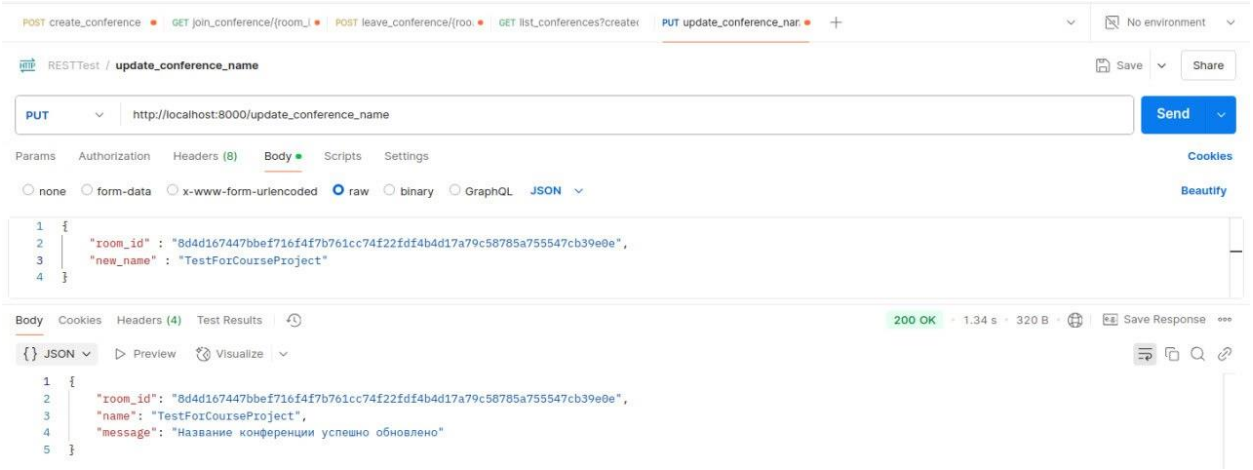



Рисунок 22 – Тестирование запроса на изменение названия конференции

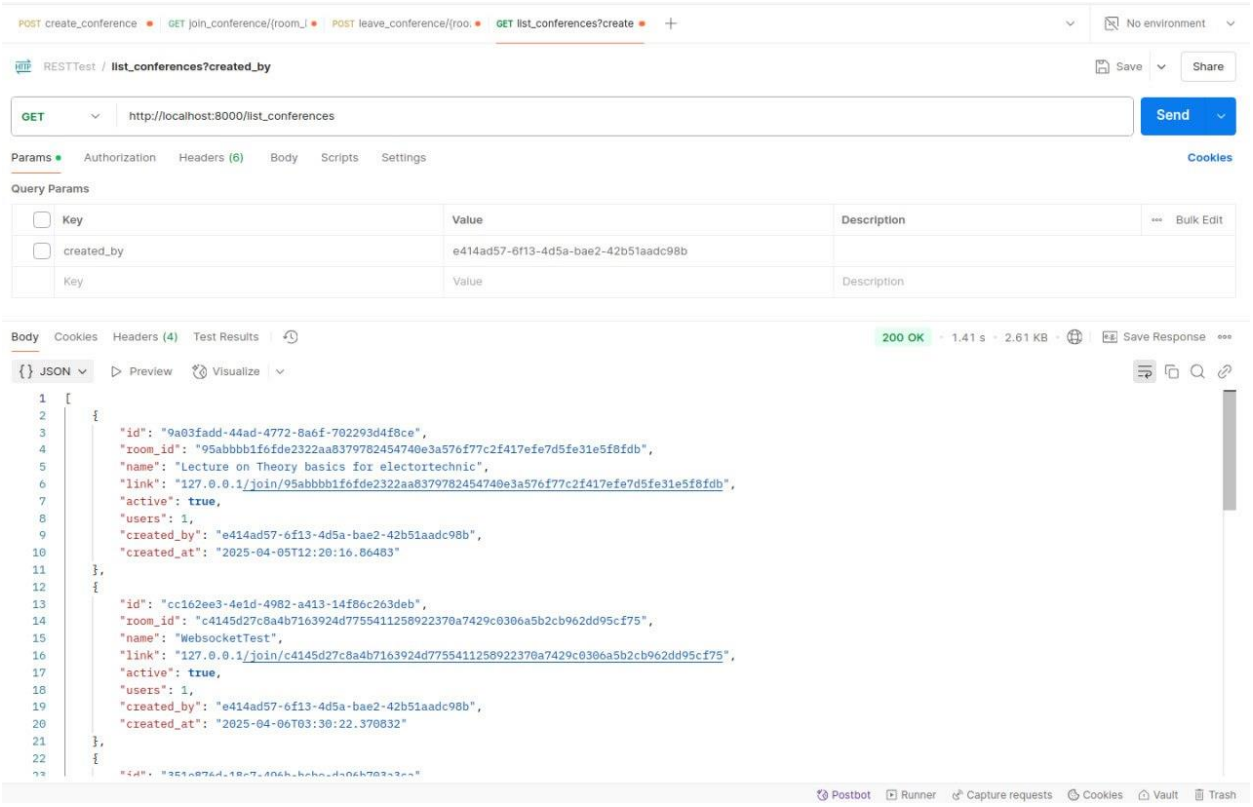



Рисунок 23 – Тестирование запроса на вывод списка доступных конференций

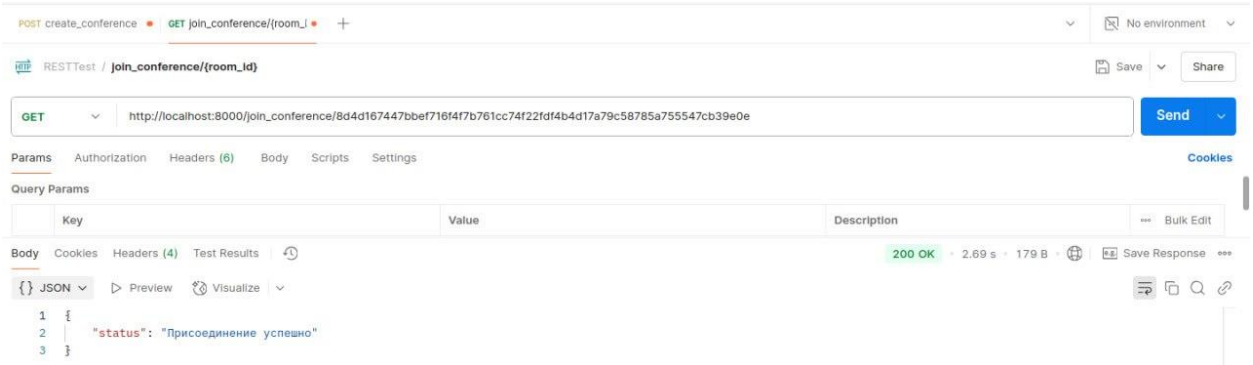



Рисунок 24 – Тестирование запроса на присоединение к комнате конференции

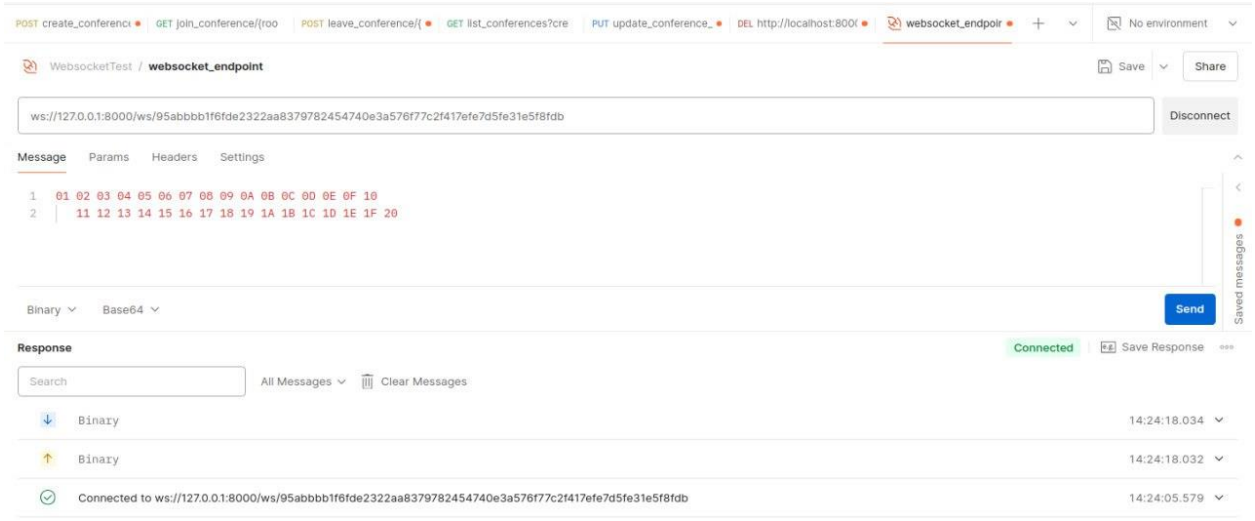



Рисунок 25 – Тестирование соединения по WebSocket и передачи данных

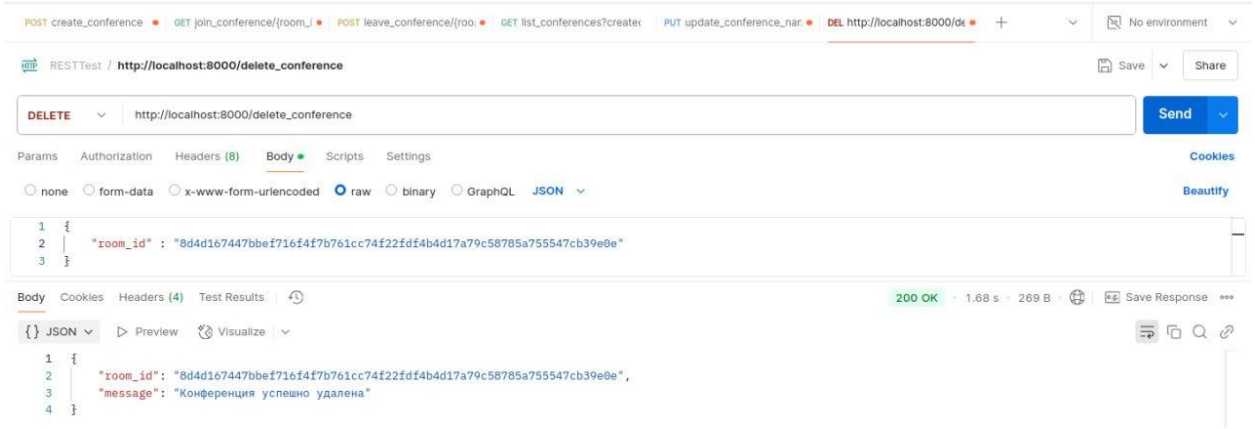



Рисунок 26 – Тестирование запроса на выход из комнаты конференции

В дополнение к документации по тестированию программных модулей также были сформированы тест-кейсы, расположенные в Приложении Б. Благодаря им всегда известно, как и что протестировать оптимальным количеством проверок, и не забыть о нюансах, так как записан каждый шаг.

# Заключение

По итогам курсового проекта разработано мобильное приложение, позволяющее создавать, управлять и участвовать в видеоконференциях. В ходе работы были выполнены исключительно важные задачи, которые позволили создать интуитивно понятное, кроссплатформенное мобильное приложение.

Первоначально была проведена исследовательская работа, которая позволила понять основные принципы и концепции организации видеоконференций, изучить и сделать выбор в направлении используемых технологий, продумать образ клиентов, являющихся целевой аудиторией, определить сценарии использования и выяснить концепции существующих на рынке приложений-прототипов.

На основе собранных данных был разработан макет, который позднее перешел в интерфейс мобильного приложения, где использовались цвета в соответствии с выбранной предметной областью, анимации, нативные виджеты для платформ Android и iOS. Для работы с данными в мобильном приложении был разработан собственный API, который позволяет осуществлять CRUD-логику, а также передачу данных в режиме реального времени. После разработки всех компонентов системы и связки их было проведено тестирование, которые позволило выявить недостатки для их дальнейшего исправления.

Архитектурные шаблоны, используемые в разработке мобильного приложения и API, позволяют обеспечить читаемый и понятный исходный код, а также масштабируемость проекта. Поскольку проект является открытым, то любой разработчик может просмотреть исходный код GitHub и внести изменения в собственную ветку, что впоследствии может войти в новую версию приложения.

# Библиография

Нормативно-правовые акты:

1 ГОСТ ЕСКД Р 2.105-2019. ЕСКД. Общие требования к текстовым документам. – Москва: Стандартинформ, 2019. – 36 с

Электронные ресурсы:

1 pub.dev [электронный ресурс]. – Репозиторий библиотек и плагинов для разработки flutter-приложений. – URL: https://pub.dev/ (дата обращения: 07.02.2024)

2 metanit.flutter [электронный ресурс]. – Руководство по фреймворку Flutter. – URL: https://metanit.com/dart/flutter/ (дата обращения: 10.03.2024)

3 supabase.com [электронный ресурс]. – Документация Supabase. – URL: https://supabase.com/docs (дата обращения: 24.02.2025)

4 habr.com [электронный ресурс]. – Исчерпывающее руководство по различным типам API. – URL: https://habr.com/ru/companies/otus/articles/737610/ (дата обращения: 26.02.2025)

# Приложение А

(обязательное)

Министерство образования Новосибирской области
ГБПОУ НСО «Новосибирский авиационный технический колледж
имени Б.С. Галущака»

Разработка мобильного приложения для видеоконференций

Техническое задание

НАТКиГ.201200.010.000ПЗ

Разработал:
Студент группы ПР-23.106
Сборщиков Е.И.

2025

# Содержание

## Введение

Настоящее техническое задание распространяется на разработку мобильного приложения для видеоконференций, используемого для создания и проведения видеоконференций между зарегистрированными пользователями.

Наименование приложения «ComfortConfy»

Область применения: мобильное приложение для звонков и видеоконференций предназначено для непосредственных пользователей приложения, установивших программный продукт на свое устройство.

Приложение ComfortConfy предполагает возможность пользователям осуществлять аудио- и видеозвонки, создавать, приглашать пользователей и проводить видеоконференции, редактировать персонализацию и настройки конфиденциальности учетной записи в приложении.

Основанием для разработки является Протокол № Уч-018/4 от «06» февраля 2025 года.

Наименование темы разработки - «Разработка мобильного приложения для звонков и видеоконференций».

Условное обозначение темы разработки - «ComfortConfy».

# 1 Назначение разработки

Основное назначение разработки программного продукта:

– обеспечение развертывания приложения на различных видах операционных систем, таких как Android и IOS;

– предоставление пользователям возможностей для совершения и видеозвонков, а также создания и проведения видеоконференций;

Лица, работающие с системой, представляют себя в качестве:

– пользователя, основные возможности которого заключаются в организации аудио- и видеозвонков, создании и проведении видеоконференций;

– администратора, основные задачи которого заключается в обеспечение качества функционирования системы и информационной безопасности.

# 2 Требования к мобильному приложению

## 2.1 Требования к функциональным характеристикам

– осуществление аудио- или видео-звонка между двумя пользователями;

– создание видеоконференции;

– подключение к видеоконференции пользователей посредством перехода по реферальной ссылке.

## 2.2 Требования к надёжности

– организация стабильного Интернет-соединения;

– соблюдение типов данных при заполнении полей;

– соблюдение норм безопасности, регламентируемых законами Российской Федерации.

### 2.3 Условия эксплуатации

Пользователь должен иметь практические навыки использования мобильных устройств под управлением операционных систем типа Android или IOS;

### 2.4 Требования к составу и параметрам технических средств

– мобильное устройство с установленной операционной системой Android не ниже версии 10

– мобильное устройство с установленной операционной системой IOS не ниже версии 12.

### 2.5 Требования к информационной и программной совместимости

Требования к информационным структурам (файлам) не предъявляются.

### 2.6 Требования к защите информации

Доступ к данным имеется только у системных администраторов приложения ComfortConfy.

### 2.7 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

## 3 Требования к программной документации

Состав программной документации включает в себя:

– пояснительную записку;

– техническое задание;

– тексты программных модулей с комментариями.

# 4 Технико-экономические показатели

Технико-экономические показатели разработки программного продукта не предъявляются.

# 5 Стадии и этапы разработки

Таблица А.1 – Стадии разработки

| № | Этапы разработки КП | Сроки выполнения | Отчётность |
|---|---|---|---|
| 1 | Определение цели и задач, объекта и предмета исследования | 08.02.2025 | Пояснительная записка |
| 2 | Описание предметной области | 26.02.2025 | Пояснительная записка |
| 3 | Выбор технологии, языка и среды программирования | 01.03.2025 | Пояснительная записка |
| 4 | Оформление технического задания | 03.03.2025 | Техническое задание |
| 5 | Проектирование UI/UXдизайна | 05.03.2025 | Спецификации программного обеспечения |
| 6 | Разработка мобильного приложения | 10.04.2025 | Схема структурная системы и спецификации компонентов |
| 7 | Отладка и тестирование приложения | 20.04.2025 | Тексты программных компонентов |
| 8 | Оформление документации | 21.04.2025 | Программная документация |
| 9 | Защита | 28.04.2025 | |

# 6 Порядок контроля и приёмки

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

– техническое задание;

– пояснительная записка;

– презентация;

– программный продукт.

# Приложение Б

(справочное)

## Документация к тестированию API

Таблица Б.1 – Создание конференции

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Создание конференции с корректными данными |
| Шаги тестирования | 1. Отправить POST-запрос на /create_conference с валидными данными. |
| | 2. Проверить, что конференция создана в базе данных. |
| | 3. Проверить, что возвращается корректный ответ с данными конференции. |
| Данные тестирования | name : String<br>create_by: UUID |
| Ожидаемый результат | Конференция создана в базе данных, возвращается ответ с данными конференции, статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.2 – Изменение названия конференции

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Изменение названия существующей конференции |
| Шаги тестирования | 1. Создать конференцию. |
| | 2. Отправить PUT-запрос на /update_conference_name с новым именем. |
| | 3. Проверить, что название обновлено в базе данных. |
| | 4. Проверить корректность ответа. |
| Данные тестирования | room_id: SHA256<br>name: String |
| Ожидаемый результат | Название конференции обновлено в базе данных, возвращается ответ с обновленным именем, статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.3 – Удаление конференции

| Наименование | Описание |
|---|---|
| 1 | 2 |
| Приоритет | Высокий |
| Название тестирования | Удаление существующей конференции |
| Шаги тестирования | 1. Создать конференцию. |

Продолжение таблицы Б.3

| 1 | 2 |
|---|---|
| Шаги тестирования | 2. Отправить DELETE-запрос на /delete_conference с корректным room_id. |
| | 3. Проверить, что конференция удалена из базы данных. |
| | 4. Проверить корректность ответа. |
| Данные тестирования | room_id: SHA256 |
| Ожидаемый результат | Конференция удалена из базы данных, возвращается подтверждение удаления, статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.4 – Вход в комнату конференции

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Присоединение пользователя к активной конференции |
| Шаги тестирования | 1. Создать конференцию. |
| | 2. Отправить GET-запрос на /join_conference/{room_id} с корректным room_id. |
| | 3. Проверить, что количество пользователей увеличилось. |
| | 4. Проверить корректность ответа. |
| Данные тестирования | room_id: SHA256 |
| Ожидаемый результат | Количество пользователей увеличилось, возвращается статус 200 OK. |

Таблица Б.5 – Выход из комнаты коныеренции

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Выход пользователя из активной конференции |
| Шаги тестирования | 1. Создать конференцию и присоединить пользователя. |
| | 2. Отправить POST-запрос на /leave_conference/{room_id} с корректным room_id. |
| | 3. Проверить, что количество пользователей уменьшилось. |
| | 4. Проверить корректность ответа. |
| Данные тестирования | room_id: SHA256 |
| Ожидаемый результат | Количество пользователей уменьшилось, возвращается статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.6 – Получение списка истории звонков

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Получение списка активных конференций с фильтрацией по пользователю |
| Шаги тестирования | 1. Создать несколько конференций. |
| | 2. Отправить GET-запрос на /list_conferences с параметром created_by. |
| | 3. Проверить, что возвращаются только конференции, созданные указанным пользователем. |
| Данные тестирования | created_by: UUID |
| Ожидаемый результат | Возвращаются только конференции, созданные указанным пользователем, статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.7 – Получение списка доступных конференций

| Наименование | Описание |
|---|---|
| Приоритет | Средний |
| Название тестирования | Получение списка всех активных конференций |
| Шаги тестирования | 1. Создать несколько конференций. |
| | 2. Отправить GET-запрос на /list_conferences без параметра created_by. |
| | 3. Проверить, что возвращаются все активные конференции. |
| Данные тестирования | Нет |
| Ожидаемый результат | Возвращаются все активные конференции, статус 200 OK. |
| Фактический результат | Статус 200 OK |

Таблица Б.8 – WebSocket-соединение

| Наименование | Описание |
|---|---|
| Приоритет | Высокий |
| Название тестирования | Установка WebSocket-соединения с существующей конференцией |
| Шаги тестирования | 1. Создать конференцию. |
| | 2. Установить WebSocket-соединение с /ws/{room_id}. |
| | 3. Проверить, что соединение установлено. |
| Данные тестирования | room_id="valid_room_id" |
| Ожидаемый результат | Соединение установлено, данные передаются корректно. |
| Фактический результат | Соединение установлено, данные передаются корректно. |