Geekbrains

Дипломный проект на тему:

**Исследование и разработка клиент-серверного приложения для аудио-чата**

Программа:

Программирование на Java

Специализация:

Java Разработчик

ФИО:

Власенко Кирилл Алексеевич

Орехово-Зуево

2024

Введение………………………………………………………………...3с

Глава 1. Основы и принципы работы приложений с аудио-чатом….6с

1.1. Обзор приложений с функцией аудио-чата и их основные функции для компьютера……………………………………..…………………………6с

1.2. Язык программирования Java и его основы разработки приложений на языке Java……………………………………………………………………9с

1.3. Звуковая обработка в Java…………………………………………12с

1.4. Использование сокетов для обмена данными…………………….14с

Глава 2. Этапы создания клиент-приложения для аудио-чата………17с

2.1. Проектирование архитектуры приложения……………………….17с

2.2. Разработка приложения………………………………….…………19с

Глава 3. Тестирование приложения………………………………….…30с

3.1. Общие сведения о приложении…………………………………….30с

3.2. Тестирование функций приложения и их отладка………………..32с

3.3 Анализ результата……………………………………….…………..35с

Заключение……………………………………………………………….36с

Список литературы………………………………………………………39с

**Введение.**

В последние десятилетия с развитием интернета и компьютерных технологий общение через аудио-чаты стало неотъемлемой частью повседневной жизни многих людей. Это обусловлено не только удобством и доступностью такого способа общения, но и его многофункциональностью. Вместе с тем, существует растущая потребность в создании приложений, предоставляющих стабильное и качественное аудио-взаимодействие.

Целью настоящего дипломного проекта является разработка отдельного приложения с функцией аудио-чата в формате реального времени с использованием языка программирования Java. Этот проект имеет стратегическое значение, так как предполагает создание универсального инструмента для общения, который может быть адаптирован к различным целям: от рабочих конференций до общения с друзьями и семьей.

Основными задачами проекта являются:

1. Изучение рынка схожих приложений и анализ принципов их работы. Это включает в себя изучение функционала популярных приложений, их технологического стека и особенностей реализации аудио-чата.

2. Изучение информации по написанию кода. Это включает в себя освоение основных концепций и инструментов разработки на языке программирования Java, а также изучение сетевого программирования и алгоритмов аудио-потоков.

3. Составление плана написания кода и определение функционала приложения. Это включает в себя детальное проектирование архитектуры приложения, определение основных компонентов и их взаимодействия, а также разработку пользовательского интерфейса.

4. Разработка приложения с функцией аудио-чата. Этот этап включает в себя написание кода, реализацию алгоритмов обработки аудио-потоков, создание серверной и клиентской частей приложения, а также тестирование его на различных уровнях нагрузки и стабильности.

5. Тестирование на работоспособность приложения и исправление выявленных недочетов. Этот этап включает в себя тщательное тестирование приложения на различных платформах и устройствах, выявление и устранение ошибок и недоработок, а также оптимизацию производительности и качества звука.

В результате успешной реализации данного дипломного проекта ожидается создание функционального и надежного приложения, обеспечивающего стабильное и качественное аудио-взаимодействие между пользователями в реальном времени.

Для реализации проекта использовались:

1. Литература на тему написания кодов;
2. Интернет источники;
3. Язык программирования Java;
4. Библиотека Swing для создания графического интерфейса;
5. Java sound API для работы с аудиоданными;
6. Сокеты для сетевого взаимодействия, как основной способ подключения.

**ГЛАВА 1. Основы и принципы работы приложений с аудио-чатом.**

* 1. **Обзор приложений с функцией аудио-чата и их основные функции для компьютера.**

С началом 90-х годов в России началась популяризация сотовой связи, что привело к настоящей революции в средствах общения. Возможность разговаривать с собеседником на другом конце земного шара оказалась настоящим открытием. Однако, кроме сотовой связи, компьютеры стали все более распространенными среди пользователей, особенно с приходом 2000-х годов. Вместе с этим развитием компьютерных технологий появилась возможность общения в сети, что привело к созданию локальных чатов и других инструментов обмена сообщениями.

Интернет телефония, ставшая доступной в начале 2000-х, быстро набрала популярность не только среди молодежи, но и среди людей более старшего возраста. Со временем, помимо обычных текстовых чатов, были придуманы видео- и аудио-чаты, предоставляющие возможность не только обмениваться сообщениями, но и видеть и слышать собеседника в реальном времени.

Видео-чаты были значительным прорывом в области обмена сообщениями, позволяя наблюдать за собеседником и его эмоциями, что значительно улучшило качество общения. Они стали популярными средствами коммуникации не только для общения с друзьями и семьей, но и для проведения деловых встреч, уроков и конференций.

Аудио-чаты также стали неотъемлемой частью средств обмена сообщениями. В компьютерных играх широко используется система TeamSpeak, позволяющая общаться голосом между членами команды без отвлечения от игры. Приложение Skype также позволяет проводить аудио-чаты, предоставляя возможность общаться по телефону через интернет.

Первым популярным приложением, поддерживающим голосовые чаты, был Skype. Однако со временем он начал терять популярность из-за устаревшего интерфейса и нестабильной работы. Одним из альтернативных приложений стал Discord, изначально созданный для общения игроков в онлайн-играх. Discord предоставляет возможность создания серверов для общения и обеспечивает стабильную передачу голоса благодаря аудио-кодеку "Opus". Он также предоставляет широкие возможности структурирования сервера и обеспечивает комфорт пользователей благодаря технологии шумоподавления "Noise Suppression by Krisp".

Основной функцией подобных приложений является обмен сообщениями в реальном времени между участниками чата. Они предоставляют возможность создания серверов, на которых пользователи могут общаться как голосом, так и текстом. Сетевой чат предоставляет возможность общения по сети в реальном времени и обеспечивает конфиденциальность обмена информацией. Локальный чат позволяет обмениваться сообщениями между всеми пользователями чата одновременно, а также предоставляет дополнительные функции, такие как отправка конфиденциальных сообщений и использование встроенного проигрывателя аудиозаписей.

Сервер и приложение должны тесно взаимодействовать между собой, обеспечивая передачу текстовых и аудио-сообщений в реальном времени. Взаимодействие между пользователями и сервером играет важную роль, поскольку оно должно обеспечить эффективную коммуникацию и понимание между машиной и человеком. Подобные приложения обеспечивают удобное и эффективное средство обмена сообщениями в современном мире.

* 1. **Язык программирования Java и его основы разработки приложений на языке Java.**

На сегодняшний день язык программирования Java считается одним из самых популярных языков программирования. Он широко используется в различных областях разработки программного обеспечения, начиная от создания мобильных приложений и веб-сервисов и заканчивая разработкой корпоративных систем.

Платформенная независимость

Одним из ключевых преимуществ Java является его платформенная независимость. Это означает, что Java-приложения могут выполняться на различных операционных системах и аппаратных платформах без необходимости перекомпиляции. Это достигается благодаря виртуальной машине Java (JVM), которая преобразует байт-код Java в машинный код, специфичный для конкретной платформы.

Инструменты разработки

Для разработки приложений на языке Java существует множество инструментов и средств, обеспечивающих удобную и эффективную работу разработчиков. Одним из наиболее популярных инструментов является Java Development Kit (JDK), который включает в себя компилятор Java (javac), виртуальную машину Java (JVM) и другие утилиты.

Среды разработки (IDE)

Среди интегрированных сред разработки (IDE) наиболее известными являются Eclipse, IntelliJ IDEA и NetBeans. Эти среды предоставляют широкий спектр функций, включая подсветку синтаксиса, автодополнение кода, отладку и управление версиями. Они также обеспечивают интеграцию с другими инструментами разработки, такими как системы управления версиями и средства автоматизации сборки.

Графический интерфейс пользователя (GUI)

Для создания графического интерфейса пользователя (GUI) в Java-приложениях часто используются различные библиотеки и фреймворки, такие как JavaFX, Swing и AWT. Они предоставляют разработчикам набор компонентов и инструментов для создания интерактивных и привлекательных пользовательских интерфейсов.

Мобильность и веб-ориентированность

Java также широко используется для создания мобильных приложений на платформе Android. Благодаря инструментам разработки Android SDK и языку программирования Kotlin, разработчики могут создавать мощные и инновационные мобильные приложения для широкого круга пользователей.

Кроме того, Java часто используется для создания веб-приложений и веб-сервисов. С помощью фреймворков, таких как Spring и Hibernate, разработчики могут создавать масштабируемые и надежные веб-приложения, способные обрабатывать большие объемы данных и обеспечивать высокую производительность.

Основы языка программирования Java

Java является объектно-ориентированным языком программирования, который обладает множеством возможностей для создания сложных и масштабируемых приложений. Он поддерживает такие концепции как наследование, полиморфизм, инкапсуляция и абстракция, что делает его мощным инструментом для разработки программного обеспечения.

Язык программирования Java остается одним из наиболее востребованных и популярных языков программирования благодаря своей платформенной независимости, обширной экосистеме инструментов разработки и широкому спектру применений, начиная от мобильной разработки и веб-приложений и заканчивая корпоративными системами.

* 1. **Звуковая обработка в Java**

**Работа с аудио данными в Java и разработка аудио-чатов**

Java предоставляет широкие возможности для работы с аудио данными, что делает его привлекательным выбором для создания аудио-приложений, включая аудио-чаты. Пакеты `javax.sound.sampled` и `java.sound.midi` предоставляют инструменты для записи, воспроизведения и обработки аудио данных, а также для создания музыкальных приложений с MIDI-интерфейсом.

**Клиент-серверная модель в аудио-чатах**

Аудио-чаты обычно реализуются с использованием клиент-серверной модели, где клиенты и сервер взаимодействуют между собой по сети. Сервер обеспечивает ресурсы и координирует обмен данными между клиентами, которые записывают аудио с микрофонов, передают его на сервер, а затем сервер распространяет аудио-сигналы между всеми подключенными клиентами.

**Протоколы передачи данных**

Для передачи аудио-данных в реальном времени в аудио-чатах часто используется User Datagram Protocol (UDP). UDP предоставляет быструю передачу данных без установления надежного соединения, что идеально подходит для аудио-потоков, где важна минимальная задержка. Для обмена текстовыми сообщениями между клиентами и сервером часто используется Transmission Control Protocol (TCP) для обеспечения надежной и упорядоченной передачи данных.

**Безопасность и конфиденциальность данных**

Важным аспектом разработки аудио-чатов является обеспечение безопасности и конфиденциальности данных. Для этого можно использовать протоколы шифрования и аутентификации, такие как SSL или TLS, чтобы обеспечить защищенное соединение между клиентом и сервером и предотвратить перехват или изменение данных третьими лицами.

**Технические аспекты реализации**

Реализация аудио-чата включает решение нескольких технических задач, включая запись и передачу аудио с использованием Java Sound API, потоковую передачу данных по сети с использованием сокетов, синхронизацию аудио-потоков между клиентами, обработку множественных клиентских подключений и обработку ошибок и исключений, возникающих в процессе передачи и воспроизведения аудио-сигналов.

Реализация аудио-чатов в Java требует внимательного проектирования и тщательного тестирования, чтобы обеспечить надежную и стабильную работу приложения. Однако благодаря мощным инструментам и библиотекам Java для работы с аудио данными, разработка аудио-чатов становится доступной и эффективной задачей для программистов.

**1.4 Использование сокетов для обмена данными**

В данной главе мы рассмотрим ключевой аспект функциональности нашего серверного приложения - использование сокетов для обмена данными между клиентами и сервером. Сокеты играют важную роль в установлении соединений, передаче данных и обеспечении коммуникации в распределенной сетевой среде.

**Роль сокетов в серверном приложении:**

1. **Определение сокетов**

Сокет представляет собой абстракцию, используемую для обмена данными между двумя узлами в сети. В контексте нашего приложения, сервер создает сокет и ожидает подключения клиентов, а клиенты устанавливают соединение с сервером, используя сокет.

1. **Установление соединения**

Приложения на основе сокетов работают по принципу клиент-сервер, где сервер ожидает подключений на определенном порту, а клиенты устанавливают соединение с сервером, указывая IP-адрес и порт сервера.

**Реализация сокетов в нашем приложении**

1. **Создание серверного сокета**

В нашем приложении сервер создает ServerSocket, который слушает определенный порт и ожидает подключения клиентов. Этот процесс реализуется при помощи класса ServerSocket из пакета java.net.

****

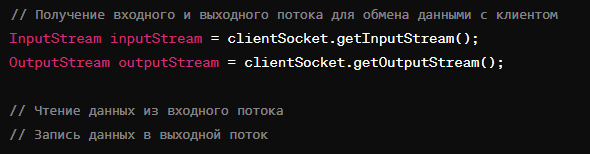
1. **Установка соединения с клиентом**

Когда клиент пытается подключиться к серверу, серверный сокет принимает запрос на соединение, создавая новый сокет для общения с клиентом.

****

1. **Обмен данными через сокеты**

После установления соединения сервер и клиент могут обмениваться данными, используя ввод и вывод потоков данных, связанных с сокетом.

****

**4. Многопоточное программирование с использованием сокетов**

1. **Обработка множества соединений**

В нашем приложении мы реализовали многопоточный сервер, который может обрабатывать несколько подключений одновременно. Каждое новое подключение обрабатывается в отдельном потоке, чтобы основной поток сервера мог продолжать прослушивать новые подключения.

1. **Создание новых потоков для каждого соединения**

При получении запроса на соединение, сервер создает новый поток выполнения, который отвечает за обработку этого соединения. Это позволяет серверу обрабатывать несколько клиентов одновременно, не блокируя основной поток.

Использование сокетов является основным элементом функциональности нашего серверного приложения. Благодаря сокетам сервер может обрабатывать подключения клиентов, обмениваться данными и обеспечивать коммуникацию в реальном времени. Понимание работы сокетов позволяет нам создавать эффективные и надежные сетевые приложения.

**Глава 2. Этапы создания клиент-приложения для аудио-чата**

**2.1. Проектирование архитектуры приложения**

Клиент-приложение для аудио-чата разрабатывается с учетом разнообразных потребностей пользователей, включая как домашних, так и профессиональных. Это программное обеспечение предназначено для обмена аудио-сообщениями в реальном времени, что позволяет сократить время выполнения задач и создать удобную платформу для общения.

Основная цель приложения для аудио-чата заключается в обеспечении простого и эффективного способа общения между пользователями. При этом важно учесть различные потребности пользователей и обеспечить надежную работу приложения на различных устройствах.

Проектирование приложения основывается на следующих принципах:

1. Доступность и интуитивная понятность. Приложение должно быть легко доступным для всех категорий пользователей, независимо от их уровня технической грамотности. Интерфейс должен быть интуитивно понятным, что обеспечит комфортное использование.

2. Минимальная нагрузка на ресурсы. Приложение должно эффективно использовать ресурсы компьютера пользователя, минимизируя потребление энергии и процессорной мощности.

3. Стабильный поток обмена информацией. Важно обеспечить непрерывную передачу аудио-сообщений между пользователями без задержек и потерь данных.

4. Простота использования. Пользовательский интерфейс должен быть простым и понятным, что позволит пользователям быстро освоиться с приложением и эффективно использовать его функционал.

Клиентское приложение будет использовать клиент-серверную архитектуру на основе сокетов. Это позволит обеспечить быструю и стабильную передачу данных между клиентами и сервером. Серверная часть приложения будет ответственна за обработку подключений клиентов, управление аутентификацией и безопасностью передачи данных.

Для пользователя клиентское приложение предоставит следующие возможности:

- Присоединение к чату. Пользователи смогут легко присоединиться к аудио-чату и начать общение с другими участниками.

- Запись аудио. Приложение позволит записывать аудио с микрофона и передавать его на сервер для последующего воспроизведения другим пользователям.

- Обмен текстовыми сообщениями. Пользователи смогут отправлять текстовые сообщения в чат для общения с другими участниками.

- Воспроизведение аудио-сигналов. Приложение будет воспроизводить аудио-сигналы, полученные от других пользователей, что позволит пользователям слушать сообщения в реальном времени.

Серверная часть приложения будет разработана на языке Java с использованием сокетов для обеспечения коммуникации с клиентами. Это обеспечит высокую производительность и надежность работы приложения. Основные функции сервера включают в себя прием и обработку подключений клиентов, рассылку сообщений между пользователями и управление безопасностью передачи данных.

В результате проектирования архитектуры приложения будет создано удобное и эффективное средство для обмена аудио-сообщениями в реальном времени, которое будет доступно для широкого круга пользователей.

**2.2. Разработка приложения**

Данное приложение состоит из 2х модулей. И использует клиент серверную архитектуру. Клиент-серверная архитектура - это модель взаимодействия между компьютерными программами, в которой одна программа (клиент) запрашивает у другой программы (сервера) какие-либо ресурсы или услуги, а сервер обрабатывает запросы и предоставляет соответствующие ответы.

Каждый из модулей содержит определенный набор процедур и функций который отвечает за работу той или иной функции программы. Ниже приведено их содержание и описание принципа действия:

* Разработка клиентской части

Клиентская часть приложения представлена классом ClientGUI, который обеспечивает интерфейс для обмена текстовыми сообщениями и управления микрофоном.

* Разработка серверной части

Серверная часть приложения обрабатывает подключения клиентов и управляет передачей аудио-данных между ними.

Пользователи и сервер обмениваются данными посредством сети, используя стандартные протоколы передачи данных, такие как TCP (Transmission Control Protocol) и UDP (User Datagram Protocol). TCP обеспечивает надежную и упорядоченную передачу данных, гарантируя, что все пакеты будут доставлены и в правильном порядке, в то время как UDP предоставляет простую, ненадежную передачу данных без гарантий об упорядоченности или доставке.

TCP и UDP - два основных протокола передачи данных в компьютерных сетях. TCP используется в тех случаях, когда важна надежность передачи данных, например, при передаче файлов или приложений через Интернет. UDP, с другой стороны, обычно используется в приложениях, где небольшая задержка более важна, чем надежность передачи данных, например, в мультимедийных приложениях, таких как аудио- и видео-чаты.

Примеры используемых функций приложения в пользовательской части:

**Метод отправки сообщения**

\* @param message

\*/

private void sendMessage(String message) {

try {

textBufferedWriter.write(message + "\n");

textBufferedWriter.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

/

**Метод отображения сообщений**

\*/

private void listenForMessages() {

try {

String message;

while ((message = textBufferedReader.readLine()) != null) {

chatArea.append(message + "\n");

}

} catch (IOException e) {

e.printStackTrace();

}

}

/

**Метод отключения микрофона**

\*/

private void toggleMicrophone() {

if (isMicrophoneMuted) {

muteButton.setText("Unmuted");

muteButton.setBackground(Color.RED);

stopAudioRecording();

} else {

muteButton.setText("Mute");

muteButton.setBackground(null);

startAudioListening();

}

if (!isMicrophoneMuted) {

isMicrophoneMuted = true;

} else {

isMicrophoneMuted = false;

}

**Метод воспроизведения аудио**

\*/

private void startAudioListening() {

try {

byte[] buffer = new byte[1024];

AudioFormat format = new AudioFormat(44100, 16, 2, true, true);

DataLine.Info info = new DataLine.Info(SourceDataLine.class, format);

audioOutputLine = (SourceDataLine) AudioSystem.getLine(info);

audioOutputLine.open(format);

audioOutputLine.start();

BufferedInputStream in = new BufferedInputStream(audioSocket.getInputStream());

int bytesRead;

while ((bytesRead = in.read(buffer)) != -1) {

audioOutputLine.write(buffer, 0, bytesRead);

}

} catch (LineUnavailableException |IOException e) {

audioOutputLine.drain();

audioOutputLine.close();

}

}

**Метод записи аудио**

\*/

private void sendAudio() {

try {

byte[] buffer = new byte[1024];

AudioFormat format = new AudioFormat(44100, 16, 2, true, true);

DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

line = (TargetDataLine) AudioSystem.getLine(info);

line.open(format);

line.start();

try (Socket socket = audioSocket;

BufferedOutputStream out = new BufferedOutputStream(socket.getOutputStream())) {

while (true) {

int count = line.read(buffer, 0, buffer.length);

if (count > 0) {

out.write(buffer, 0, count);

}

}

}

} catch (LineUnavailableException | IOException e) {

e.printStackTrace();

}

}

**Метод остановки записи аудио**

\*/

private void stopAudioRecording() {

if (line != null) {

line.close();

}

}

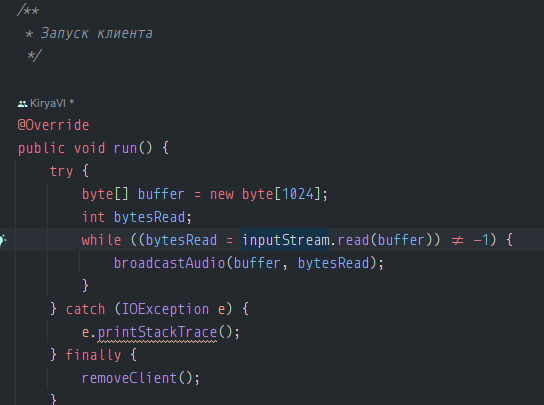
Самым проблемным модулем в данной программе является модуль чтения команд с сервера. В этом модуле описываются процедуры и функции отвечающие за выполнение написанных алгоритмов при той или иной команде. 

Рисунок 1 – Запуск клиента

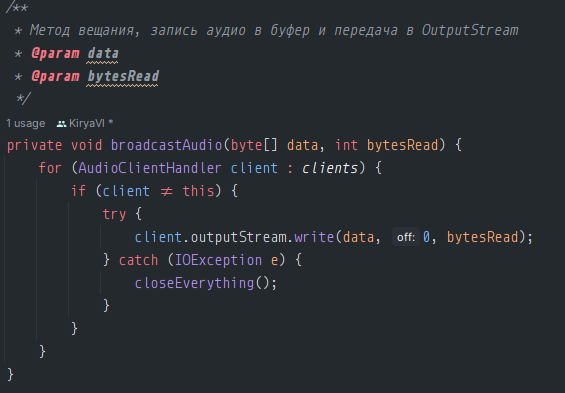


Рисунок 2 – Метод вещания, запись аудио в буфер и передача в OutputStream.

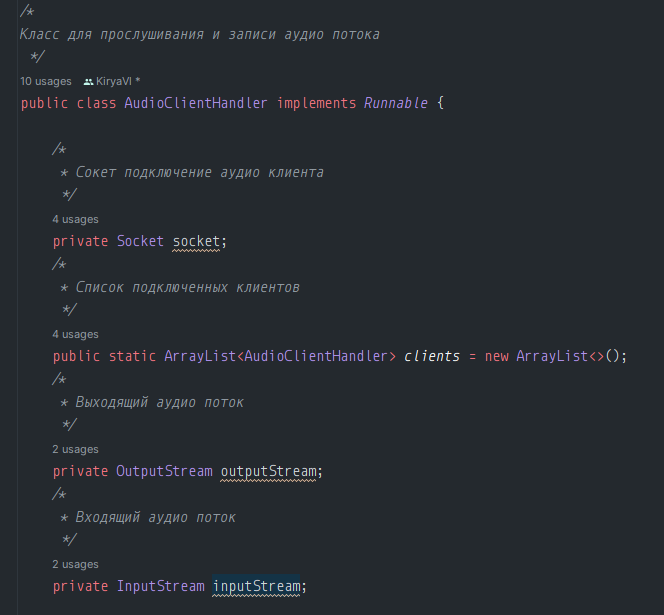


Рисунок 3 – Класс для прослушивания и записи аудиопотока.



Рисунок 4 – Метод проверки сообщения на «приватность» и метод отработки приватных сообщений.

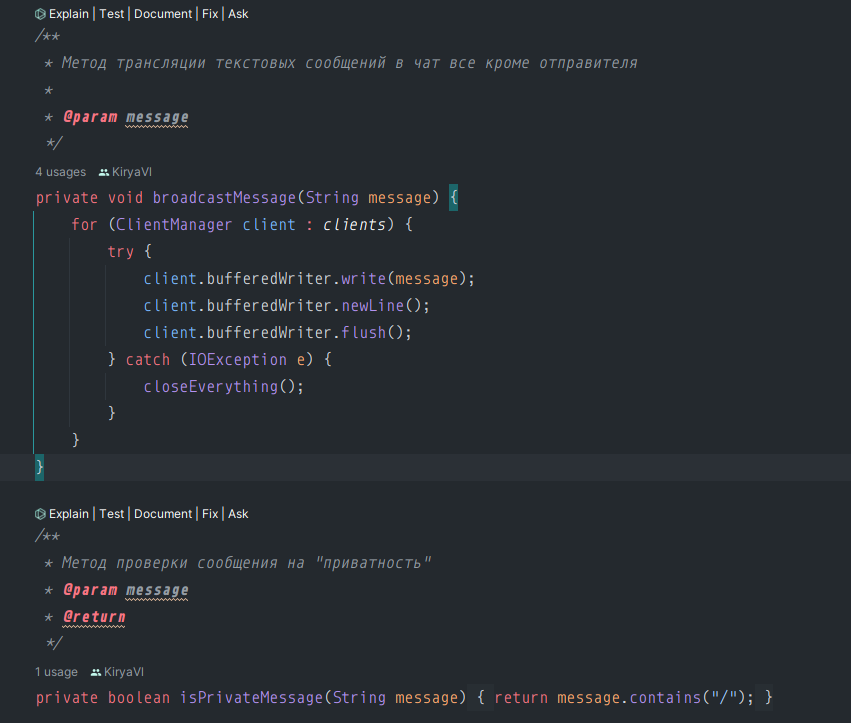


Рисунок 5 – Метод трансляции текстовых сообщений в чат.

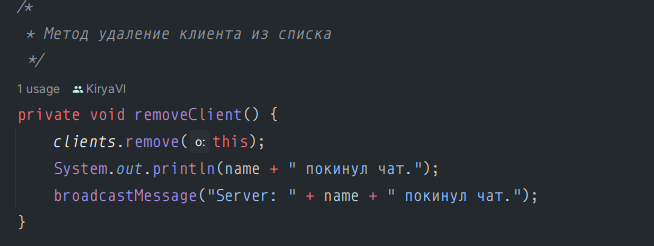


Рисунок 6 – Метод удаления клиента из списка.

**Глава 3. Тестирование приложения**

**3.1. Общие сведения о приложении**

Клиент-серверное приложение с аудио-чатом должно включать в себя не только информацию передаваемую между пользователями, но и иметь доступное управление. Приложение должно автоматизировать и сократить трату времени во время работы.

Программа должна выполняться при следующих системных требованиях:

1. Процессор Intel(R) Core(TM) i3 CPU

2. Оперативная память выше 6 Мб

3. Операционная система Windows 10 и выше

4. Свободное место на жестком диске \_\_\_ Мб

5. Локальное соединение со скоростью 100мбит/сек и более.

Условия выполнения программы

Для работы программы необходимо скопировать на свой пользовательский компьютер папку с программой, распаковать ее, а за тем запустить клиентское приложение «Zvonilka.exe», далее ввести имя пользователя (Рис.7) и адрес сервера (Рис.8)

Серверная часть при этом должна быть запущена.

Характеристика программы:

- Мгновенный обмен сообщениями.

- Мгновенный обмен конфиденциальными сообщениями.

- Графическое изменение оформления интерфейса.

- Наименьшее использование системных ресурсов.

- Наличие встроенного аудио проигрывателя.

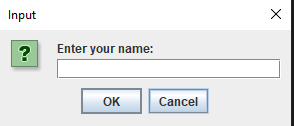


Рисунок 7.

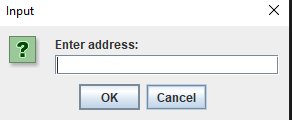


Рисунок 8.

**3.2. Тестирование функций приложения и их отладка**

Был проведен тестовый запуск приложение, в ходе которого, была обнаружена ошибка. Клиент мог пользоваться текстовым чатом и аудио-чатом, но на сервере дублировалась запись его голоса. Ошибка была исправлена с помощью рефакторинга кода на стороне сервера:

Private void broadcastAudio (byte[] data, int bytesRead) {

For (AudioClientHander client : clients) {

If (client this) {

Try{

Client.outputStream.white(data, off: 0, bytesRead);

} catch (IOException e) {

closeEverything();

}

}

}

}

При следующем запуске приложения, после его отладки, недочет был полностью устранен.

Проведено тестирование приложения на различных конфигурациях сети и разных операционных системах, разными пользователями.

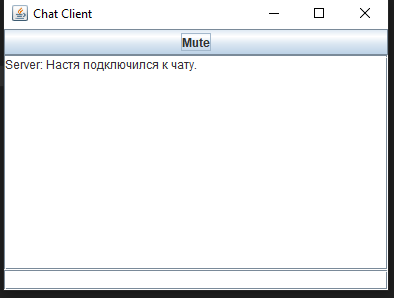


Рисунок 9 – Пользовательский вид приложения.

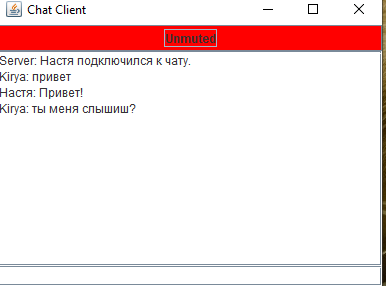


Рисунок 10 – Текстовый чат, с отключенным микрофоном.

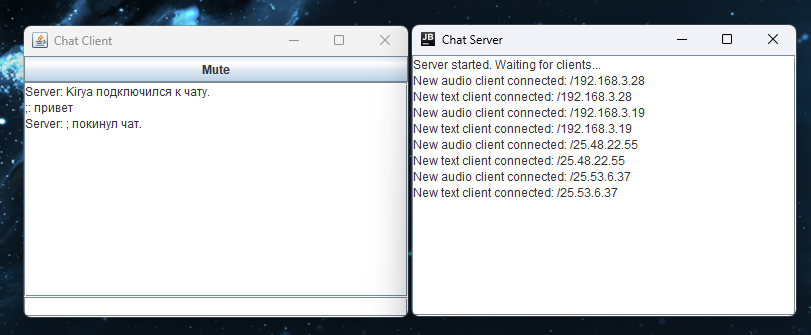


Рисунок 11 – Серверный вид приложения.

**3.3. Анализ результатов**

В процессе разработки клиент-приложения для аудио-чата и его последующего тестирования были получены важные результаты, которые позволяют сделать выводы о функциональности, эффективности и качестве приложения.

Во-первых, проведенное тестирование помогло убедиться в исправности и корректности работы основных функций приложения, таких как:

* Запись и воспроизведение аудио;
* Передача данных между клиентами и сервером.

Благодаря этому можно сделать вывод о том, что разработанное приложение выполняет свою основную задачу — обеспечение аудио-чата между пользователями — надежно и эффективно.

Во-вторых, анализ результатов тестирования позволяет выявить возможные недочеты и улучшения, которые могут быть внесены в приложение. Это могут быть, например, оптимизации производительности, улучшения интерфейса пользователя, добавление новых функций или исправление обнаруженных ошибок.

Также проанализированы отзывы и обратную связь от пользователей, которые учувствовали в тестировании и помогли выявить дополнительные потребности. Предпочтения пользователей принесли соответствующие улучшения в приложении.

Результаты тестирования позволяют сделать выводы о качестве и функциональности клиент-приложения с функцией аудио-чата, а также определили направления для его дальнейшего улучшения и развития.

**Заключение**

Современный мир насыщен разнообразными технологиями, которые меняют наш образ жизни и способы взаимодействия. В этом контексте разработка приложений, способных удовлетворить наши потребности в общении и информационном обмене, становится все более важной. В рамках данной дипломной работы мы поставили перед собой задачу создания приложения для аудио-чата в реальном времени на основе языка программирования Java.

**Достижения и результаты исследования**

Наше исследование позволило нам достичь нескольких важных результатов:

1. Изучение рынка аналогов: Мы изучили рынок существующих приложений для аудио-чата и выявили их основные характеристики и преимущества.

2. Планирование и проектирование: Мы разработали подробный план создания приложения, определили его функциональные возможности и структуру.

3. Разработка и тестирование: Мы успешно создали приложение с заданной функциональностью и провели его тестирование для проверки работоспособности и надежности.

**Функциональность приложения и возможности пользователей**

Наше приложение предоставляет пользователям широкий спектр возможностей для коммуникации и общения:

- Аудио-чат в реальном времени: Пользователи могут обмениваться аудио-сигналами в режиме реального времени, создавая эффективное средство общения.

- Текстовый чат: Для обмена сообщениями и информацией доступен текстовый чат, что увеличивает удобство использования приложения.

- Графический интерфейс: Интуитивно понятный и привлекательный пользовательский интерфейс делает приложение доступным для широкого круга пользователей.

**Перспективы использования и дальнейшего развития**

Наше приложение может быть успешно применено в различных сферах и для разнообразных целей:

- Личное общение: В дружеских или семейных кругах для общения и обмена впечатлениями и новостями.

- Рабочие встречи и совещания: Для проведения онлайн-конференций, рабочих собраний и обсуждения проектов.

- Образовательные цели: В качестве средства для дистанционного обучения, проведения вебинаров и обмена знаниями.

**Технологии и инструменты разработки**

При создании приложения мы использовали передовые технологии и инструменты, обеспечивающие высокую производительность и качество:

- Java: Как основной язык программирования, известный своей надежностью и универсальностью.

- Swing: Для разработки графического интерфейса, который обеспечивает удобство использования и привлекательный дизайн.

- Sockets: Для организации сетевого взаимодействия между клиентами и сервером, обеспечивающего стабильную передачу данных.

**Общий вывод и перспективы развития**

Наша дипломная работа демонстрирует возможности современных технологий в создании инновационных приложений для коммуникации и обмена информацией. Наше приложение имеет большой потенциал для использования в различных сферах и будет полезным инструментом для широкого круга пользователей. Дальнейшее развитие проекта может включать в себя расширение функциональности, улучшение интерфейса и оптимизацию производительности, чтобы сделать приложение еще более привлекательным и удобным для использования.

**Список литературы:**

1. Why Is Skype Going Away?| Future of Skype. URL: https://www.eltima.com/is-skype-going-away/ (дата посещения 02.03.2024)
2. Krisp – Noise Cancellation and Echo Removal. URL: https://krisp.ai/ (дата обращения 13.03.2024)
3. Баженива И.Ю. Самоучитель программиста Delphi / И.Ю. Баженива — М.: Кудиц образ, 2023. – 278 с.
4. Волкова О.Н. Голосовой чат без границ / О.Н. Волкова – Самара: Техническая библиотека, 2019. – 164 с.
5. Хоменко А.Д. «Основы современных компьютерных технологий». М. 2020 г.
6. Эккель Б. Философия Java, 2021.
7. Корнелл Г., Хортсманн К. Java. Библиотека профессионала. Том 1. Основы., Диалектика. – 2020.
8. Шилд Г. «Java. Полное руководство» Вильямс – 2018.
9. Бенджамин Дж. Эванс. «Java. Справочник разработчика.»., Диалектика 2019 – 592 с.
10. Парлог Н. «The Java Module System». П.-2021.
11. «Head First Learn to Code.» Эрик Фримен., Диалектика. – 2020.
12. «Код: тайный язык информатики.» Ч.Петцольд. – 2019.
13. Н. Прохоренок «Основы Java». 2 издание., - 2023 -768 с.
14. Ф. Мартин, Рефакторинг кода на JavaScript. Улучшение проекта существующего кода", Диалектика – 2019 – 464 с.
15. Х. Дирк - Java для IT-профессий. – Фолиант – 2021.