Содержание

[**Введение** 4](#_Toc311448933)

[**1.** **Анализ исходных данных** 5](#_Toc311448934)

[**1.1 API функции сетевого взаимодействия (sockets API)** 5](#_Toc311448935)

[**1.2 Запуск программы на выполнение в виде службы операционной системы Windows** 8](#_Toc311448937)

[**1.3**  **Выбор и обоснование среды разработки** 9](#_Toc311448939)

[**2.1 Проектирование алгоритма работы программы** 10](#_Toc311448940)

[**2.2 Проектирование интерфейса** 11](#_Toc311448941)

[**3. Реализация и тестирование** 13](#_Toc311448942)

[**3.1 Реализация программы** 13](#_Toc311448943)

[**3.2 Тестирование работоспособности программы** 15](#_Toc311448944)

[**Заключение** 16](#_Toc311448945)

[**Литература** 17](#_Toc311448946)

[**Приложение А** 18](#_Toc311448951)

[**Приложение В** 20](#_Toc311448952)

[**Приложение Г** 21](#_Toc311448953)

[**Приложение Д** 22](#_Toc311448954)

**Введение**

Целью данной курсовой работы является разработка приложения «Видео чат».

Видео чат – это приложение, обеспечивающее голосовое общение между людьми с поддержкой видео режима. Для передачи видео и аудио потоков часто используется специальные протоколы с шифрованием или без. Данное приложение позволяет возможности регистрации новых пользователей, добавлять новые контакты, а также совершать звонки и участвовать в видео конференциях. На текущий момент существует немного приложений предоставляющих такие возможности. Одно из самых известных приложений «Skype» пользуется своей популярностью благодаря поддержи протокола с шифрованием. «Видео чат» работает через протокол RTP, а так же использует современные технологии обработки видео и аудио.

**Анализ исходных данных**

**1.1 Характеристики требований на разработку**

В данном курсовом проекте необходимо разработать приложение «Видео чат». Для реализации поставленной задачи, нужно создать сервер, который будет принимать запросы от пользователей, и обрабатывать их, а так же клиентское приложение, которое будет предоставлять удобный интерфейс для работы с приложением. Для хранения информации о контактах, пользователях, подключениях нужно разработать специальный модуль, который будет делаться на две части: веб-аппликация и REST – сервисы.

**1.2 Анализ предметной области**

Поскольку необходимо разработать приложение «Видео чат», необходимо выделить все основные задачи и сгруппировать их по типам. Выделим два основных типа задач: клиентские и серверные. В клиентские задачи входит работа с устройствами для захвата видео и аудио, возможность авторизироваться в системе, добавлять новые контакты и совершать звонки. Так же должна быть возможноть для выбора и настройке устройств. Исходя из выше перечисленных задач можно проанализировать существующий пути решения и выбрать наиболее подходящие. Для работы с устройствами выбран «JMF framework». Такой выбор обоснован тем, что JMF полностью написан на java и не использует других библиотек, а так же встроенная поддержка RTP позволяет отсылать видео и аудио потоки по сети. Пользвательский интерфейс будет разрабатываться с использование Swing.

Swing — библиотека для создания графического интерфейса для программ на языке Java. Он содержит ряд графических компонентов, таких как кнопки, поля ввода, таблицы и т. д. Swing реализует известный шаблон проектирования c основными принципами UI. Этот шаблон проектирования называется Модель-Представление-Контроллер (Model-View-Controller - MVC) и стремится "разделить роли". MVC хранит код, ответственный за внешний вид чего-нибудь, отдельно от кода, обрабатывающего данные, и отдельно от кода, реагирующего на взаимодействие и выполняющего изменения. Для использования данного фреймворка не требуется сторонних библиотек, т.к. swing поставляется вместе с jdk.

Теперь рассмотрим задачи, которые относятся к задачам серверного типа. Для хранения информации о пользователях, контактах и подключениях, нужно хранилище. Для этой задачи будет использоваться Mongodb.

MongoDB — документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++ и распространяется в рамках лицензии Creative Commons.

Для предоставления возможности регистрации новых пользователей, нужно разработать веб-аппликацию, которая будет развернута в каком либо контейнере (для данной задачи выбран TomCat).

Для взаимодействия пользователя с сервером была выбрана архитектура REST.

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами.

В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.иц. Написана на языке C++ и распространяется в рамках лицензии Creative Commons.

Одной из самых сложных и важных задач является разработка архитектуры модуля организации звонков между пользователями. Вся архитектура базируется на уже существующих паттернах взаимодействия. Для данной задачи подходит паттерн «Request Reply» или так же известный «Exchange Pattern». Реализация данного паттерна позволит организовать процесс звонка – ответа. В качестве брокера будет выбран ActiveMQ.

ActiveMQ — это message broker с открытым исходным кодом (распространяется под лицензией Apache 2.0), который полностью реализует Java Message Service 1.1 (JMS). Он обеспечивает «Enterprise Features», такие как кластеризация, хранение сообщений, с возможностью использовать различные базы данных, кэширование и ведение журналов.

Исходя из выше перечисленных задач и методов их решения можно выделить основные модули:

|  |  |
| --- | --- |
| Название модуля | Опсиание |
| phantombox-domain | Содержит все необходимые модели |
| phantombox-server | Веб-аппликация и REST |
| phantombox-broker | Модуль звонков |
| phantombox-client | Клиент |

Таблица 1 – Основные модули приложения

**1.3 Специальное и общесистемное программное обеспечение**

Для разработки проекта использовалась IntelliJ IDEA, как среда разработки и Gradle для сборки проекта. После компиляции и сборки проекта создается исполняемый «jar» файл, для запуска которого нужна установка jdk 1.7. В качестве хранилища используется Mongodb, ActiveMQ –брокер, TomCat – веб-контейнер.

Для сборки модулей и загрузки артефактов в локальный репозиторий нужно выполнить команду «gradle clean build UA», ниже перечислен порядок сборки.

1. phantombox-domain
2. phantombox-server
3. phantombox-broker
4. phantombox-client

После сборки «phantombox-server» будет создано два основных артефакта: «phantombox-server-web» и «phantombox-server-rest-web», которые нужно развернуть в TomCat. Для этого нужно скопировать данные jar файлы в папку webapps TomCat. После этого запустить сервер.

Для запуска MongoDB ниже описаны основные шаги:

1. Создать новый файл в папке bin MongoDB и назовать его mongodb.config
2. Добавить в mongodb.config одну строку: dbpath=ПУТЬ\_КУДА\_НУЖНО\_СОХРАНЯТЬ\_ФАЙЛЫ\_БАЗЫ\_ДАННЫХ. Например, в Windows можно написать dbpath=c:\mongodb\data а в Linux — dbpath=/etc/mongodb/data.
3. Запустить mongod с параметром --config /path/to/your/mongodb.config.

Для запуска ActiveMQ нужно запустить activemq.bat без всяких дополнительных действий.

После нужно собрать клиента и запустить.

# **2. Программное проектирование**

**2.1 Информационные структуры**

Разрабатываемое приложение осуществляет процессы по обработке некой информации. Исходя из предыдущей главы, уже известны базовые процессы и взаимодействия, теперь нужно выяснить какие модели будут участвовать в процессах. Ниже в таблице перечислены основные модели:

|  |  |
| --- | --- |
| Модель | Описание |
| User | Информация о пользователе |
| Contact | Информация о контакте |
| Connection | Информация о соединении |
| Device | Информация об устройствах (беб-камеры, микрофоны) |
| Setup | Настройки программы |

Таблица 2 – Информационные структуры

Это основные модели, которые участвуют в взаимодействии разных модулей.

## 2.2 Математическое обеспечение

Под математическим обеспечением часто понимают аппаратное обеспечение. Для работы с устройствами используется JMF, его можно загрузить с официального сайта. После чего нужно установить продукт и запустить «jmfinit.exe», JMF найдет все устройства в системе и сконфигурирует файл «jmf.properties» который будет находится в папке lib. Нужно скопировать этот файл в папку «phantombox-client\phantombox-client-ui\src\main\resources\», после чего можно собрать и запустить приложение.

Наличие установленного «JMF» в системе обязательно.

Для сборки используется Gradle build tool. Хорошая и удобная утилита для сборки проекта. TomCat, ActiveMQ и MongoDB так же должны быть установлены в системе. Все эти приложения можно найти на официальных ресурсах.

## 2.3 Функциональное обеспечение

Каждый модуль приложения предоставляет некую функциональность. В таблице ниже перечислены основные модули и их функции.

|  |  |
| --- | --- |
| Модуль | Функции |
| phantombox-server | Сохранение и получение пользователей. Создание нового пользователя и авторизация, получение ip адреса, добавление новых контактов. |
| phantombox-broker | Отправка запроса на видео звонок и получение ответа |
| phantombox-client | Авторизация, , добавление и удаление контактов, настройка устройств, звонки, информация о приложении. |

Таблица 3 – Функциональное обеспечение

Функциональные возможности модулей зависят от типов задач, которые они решают.

## 2.4 Алгоритмическое обеспечение

В данной главе будет рассмотрен алгоритм захвата потока с устройств и отправка его по протоколу RTP. Как уже упоминалось JMF поддерживает протокол RTP, что позволяет пользоваться уже готовой реализацией протокола RTP.

Протокол RTP (англ. Real-time Transport Protocol) работает на транспортном уровне и используется при передаче трафика реального времени.

Протокол RTP переносит в своём заголовке данные, необходимые для восстановления голоса или видеоизображения в приёмном узле, а также данные о типе кодирования информации (JPEG, MPEG и т. п.). В заголовке данного протокола, в частности, передаются временная метка и номер пакета. Эти параметры позволяют при минимальных задержках определить порядок и момент декодирования каждого пакета, а также интерполировать потерянные пакеты.

Спецификация RTP описывает два под-протокола:

* Протокол передачи данных, RTP, который взаимодействует с передачей данных реального времени. Информация, предоставляемая посредством этого протокола включает тайм-стемп (для синхронизации), последовательный номер (для детектирования потери и дублирования пакетов) и формат полезной нагрузки, который определяет формат кодирования данных.
* Протокол контроля, RTCP, используемый для определения качества обслуживания ([QOS](http://ru.wikipedia.org/wiki/QOS)), обратной связи и синхронизации между медиа-потоками. Занимаемая полоса пропускания RTCP — мала в сравнении с RTP, обычно около 5 %.
* Управляющий сигнальный протокол, такой как [SIP](http://ru.wikipedia.org/wiki/SIP), [H.323](http://ru.wikipedia.org/wiki/H.323), [MGCP](http://ru.wikipedia.org/wiki/MGCP) или [H.248](http://ru.wikipedia.org/wiki/H.248). Сигнальные протоколы управляют открытием, модификацией и закрытием RTP-сессий между устройствами и приложениями реального времени.
* Управляющий протокол описания медиа, такой как [Session Description Protocol](http://ru.wikipedia.org/wiki/Session_Description_Protocol" \o "Session Description Protocol).

На рисунке 1 изображена архитектура RTP протокола.

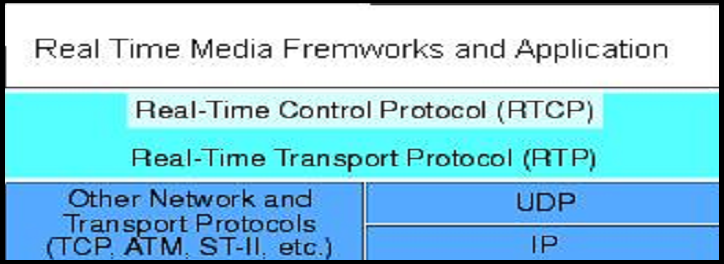


Рис. 1 – Архитектура RTP протокола

JMF 1.0 API (Java Media Player API) позволили программистам разрабатывать Java программы, которые работают с медиа-данными. JMF 2.0 API расширяет возможности получения и хранения медиа-данных, управления типом обработки, которая выполняется во время воспроизведения, а также выполнение кастомизированных обработок над медиа-потоками данных. Кроме того, JMF 2.0 определяет плагин API, который позволяет опытным разработчикам и поставщикам технологий с легкостью настраивать и расширять функциональность JMF.

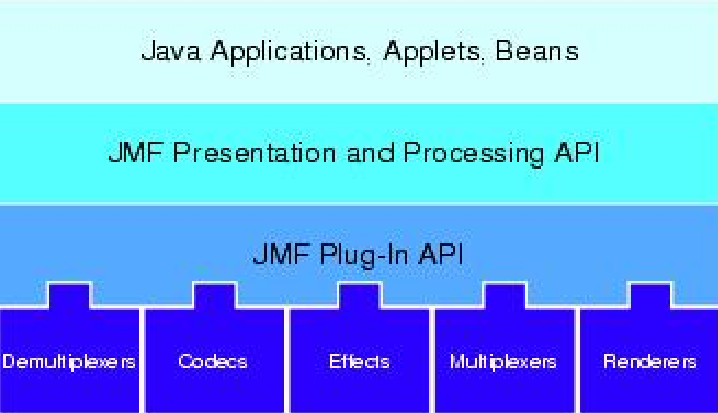


Рис. 2 – Архитектура JMF

Изучив документацию и API «JMF» можно выделить основные компоненты:

DataSource – источник данных. Может быть создан из MediaLocator, который в свою очередь является метаинформацией об устройстве, и создается на основе CaptureDeviceInfo.

Processor – обработчик медиа-данных. Этот компонет выполняет самую основную работу. Он получает данные из DataSource и обрабатывает их в зависимости от типа данных.

Transmitter – передачик. Существует два типа передатчиков: DataSink и RTPM.

Ниже на рисунке изображена общая схема взаимодействия компонентов.

Исходя из выше перечисленного можно выделить следующие шаги к конфигурации компонентов.

1. Получить все устройсва в сети. Для этого можно воспользоваться CaptureDeviceManager.
2. Выбрать нужное устройство и создать MediaLocator
3. На основе MediaLocator создать DataSource
4. Передать DataSource в Processor и обработать медиа данные.
5. Получить DataOutput из Processor и начать передечу данных.

Ниже изображена схема отправителя медия-контента.

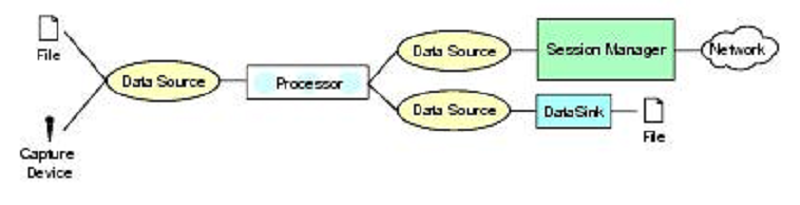


Рис. 3 – Отправитель

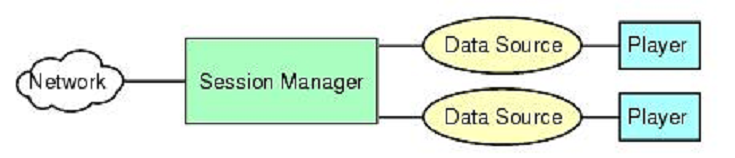


Рис. 4 – Получатель

JMF предоставляет удобный API, который позволяет управлять процессами максимально эффективно. В приложении были реализованы специальные менеджеры, которые отвечают за настройку и конфигурацию компонентов.

## 2.5 Пользовательский интерфейс

Пользовательский интерфейс должен отвечать следующим требованиям:

1. Простота.
2. Удобство.
3. Прямолинейность.
4. Надежность.

По функциональным требованиям интерфейс должен отвечать следующим критериям:

1. Авторизация.
2. Добавление и удаление новых контактов.
3. Выбор нужных устройств (веб-камера и микрофон).
4. Функции звонка и ответа.
5. Выход и приложения.
6. Информация.

# **3 Программная реализация**

## 3.1 Детальная реализация функциональных частей программного обеспечения

В данной главе будут рассмотрены основные аспекты реализации функциональных частей. Так же будут описаны способы взаимодействия с функциональными частям приложения.

#### **3.1.1 Реализация программной части**

Приложение состоит из трех основных модулей: client, sever, broker, каждый из которых так же разбит на подмодули. Ниже на диаграмме изображены основные компоненты сервера.

**SERVER**

**Storage**

**server-rest-call**

**server-storage**

**server-service**

**TomCat**

web application server

**server-rest-web**

**server-web**

Клиент отправляет REST запросы на сервер

**CLIENT**

Рис. 5 – Диаграмма сервера

**Модуль broker реализует паттерн** «Request Reply» для предоставления функции звонка. Ниже на рисунке изображена общая диаграмма паттерна и схематичное изображение процесса звонка.

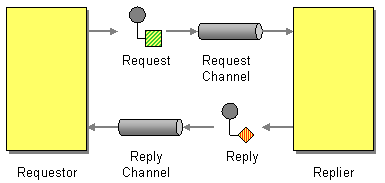


Рис. 6 – Паттерн «Request Reply»

Request queue

Request queue

REQUEST

RESPONSE

Рис. 7 – Схематичное отображение процесса звонка

#### **3.1.2 Реализация пользовательского интерфейса**

**Заключение**

В результате выполнения курсового проекта была разработана сетевая игра «Шахматы».

Согласно задания приложение работает под операционной системой Windows. Запуск сервера осуществляется в виде службы операционной системы.

При выполнении работы были получены новые знания и навыки программирования сокетов и системных служб операционной системы Windows, а также создания сетевых приложений типа «клиент-сервер» в среде разработки Delphi версии 7 средствами сокетов.

Использование данной игры не требует особых знаний и усилий со стороны пользователя и больших затрат ресурсов компьютера. Поэтому она может использоваться широким кругом пользователей.

**Литература**

1. Архангельский А.Я. Программирование в Delphi. Учебник по классическим версиям Delphi. – М.:ООО «Бином-Пресс», 2006. – 1152 с.
2. Дарахвелидзе П.Г., Марков Е.П. Программирование в Delphi 7. – СПб.:

БВХ-Петербург, 2003. – 784 с.: ил.

1. Джамса К., Коуп К. Программирование для Интернет в среде Windows/Пер. с англ. — СПб.: Питер, 1996.
2. Оланд Д., Джонс Э. Программирование в сетях Microsoft Windows , ISBN: 5-318-00725-2 издательство "Питер", 2002
3. Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. – СПб: Питер, 2001 ISBN 5-318-00453-9

**Приложение А**

(обязательное)

 Диаграмма вариантов использования **Приложение Б**

(обязательное)

Диаграмма деятельности.



**Приложение В**

(обязательное)

Диаграмма развертывания



**Приложение Г**

(обязательное)

Диаграмма компонентов

**Приложение Д**

(обязательное)

Протокол тестирования

|  |  |  |
| --- | --- | --- |
| Функция | Результат тестирования | Примечание |
| Регистрация сервиса сервера | работает | Только администратор |
| Старт сервера как сервис | работает | Только администратор |
| Останов сервиса | работает | Только администратор |
| Удаление сервиса | работает | Только администратор |
| Подключение к серверу | работает | Только пользователь |
| Старт игры | работает | Только пользователь |
| Выбор вариантов хода | работает | Только пользователь |
| Переход хода | работает | Только пользователь |
| Завершение игры | работает | Только пользователь |
| Отключение от сервера | работает | Только пользователь |

Таблица Д.1 – Протокол тестирования