Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____Информатика и системы управления_____

КАФЕДРА ____Программное обеспечение ЭВМ и информационные технологии» (ИУ7)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Система онлайн-трансляций.

Студент ____ИУ7-73Б_____ _____ ____**Ильясов И. М.**____
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент ____ИУ7-73Б_____ _____ ____**Сиденко А. Г.**____
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент ____ИУ7-73Б_____ _____ ____**Степанов А. О.**____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ ____**Рогозин Н. О.**___
(Подпись, дата) (И.О.Фамилия)

2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ7__
(Индекс)
_____ __И.В.Рудаков___
(И.О.Фамилия)
« _____ » _____ 20_ __ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _Компьютерные сети_____

Студенты группы __ИУ7-73Б_
_____Ильясов Идрис Магомет-Салиевич,_____
_____Сиденко Анастасия Генадьевна,_____
_____Степанов Александр Олегович_____
(Фамилия, имя, отчество)

Тема курсового проекта _Система онлайн трансляций_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_учебный_____
Источник тематики (кафедра, предприятие, НИР) ___кафедра_____

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание _Разработать клиент-серверное приложение системы онлайн трансляций. Запись_____
происходит с камеры телефона на Андроид. Просмотр с десктопного приложения._____

Оформление курсового проекта:

Расчетно-пояснительная записка на _20-25__ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
К защите должны быть подготовлены презентация и доклад, отражающие суть выполненной работы, содержание и методы решения основных задач, а также полученные результаты.

Дата выдачи задания « _5_ » __октября__ 2020 г.

Руководитель курсового проекта	05.10.20 (Подпись, дата)	__Н.О. Рогозин_____ (И.О.Фамилия)
Студент	05.10.20 (Подпись, дата)	_И. М. Ильясов_____ (И.О.Фамилия)
Студент	05.10.20 (Подпись, дата)	_А. Г. Сиденко_____ (И.О.Фамилия)
Студент	05.10.20 (Подпись, дата)	_А. О. Степанов____ (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

РЕФЕРАТ

Данная работа посвящена разработке системы онлайн-трансляций, предназначенной для транслирования звука и изображения с камеры устройства на операционной системе Android на Desktop-приложение для платформы Windows.

ВВЕДЕНИЕ

В настоящее время большую популярность обретают видеостриминговые площадки. Большинство производителей видео-контента (журналисты, блогеры, телекомпании, интернет-телевидение и т.д.) ведут прямые трансляции в социальных сетях или на своих сайтах поскольку они наиболее привлекательны для пользователя. Также в последнее время повсеместно начинают использоваться дроны для различных задач (доставить посылку, снять красивые виды). Управление дроном невозможно без использования видеотрансляции. Поэтому в наше время очень важно иметь возможность транслировать видеоизображение с камеры устройства под управлением Android, потому что это популярная мобильная операционная система с открытым исходным кодом. Рассматриваемый в данной работе сценарий использования предполагает, что пользователь запускает прямую трансляцию с мобильного устройства под управлением Android, используя приложение, передающее трансляцию на рестример, с которого можно получить и отобразить трансляцию на десктопном приложении Windows.

Целью данной работы является разработка программного комплекса для трансляции видеоизображения с камеры мобильного устройства на платформе Android на десктопное приложение компьютера под управлением Windows.

Для достижения данной цели ставятся следующие задачи:

1) выбрать протокол потокового видео для организации трансляции;
2) выбрать средства для разработки;
3) разработать мобильное Android-приложение;
4) разработать рестример;
5) разработать десктопное приложения.

1 Аналитический раздел

В данном разделе производится постановка задачи, анализируется предметная область, рассматриваются существующие на данный момент протоколы.

1.1 Постановка задачи

Требуется разработать клиент-серверное приложение, позволяющее запускать и воспроизводить онлайн-трансляции. Запись производится с камеры телефона с установленной ОС Android. Онлайн-трансляция просматривается через десктопный клиент на ПК с установленной системой Windows. Данный комплекс приложений должен обладать следующим функционалом.

1. Передача видео и аудио с телефона Android.
2. Остановка и запуск трансляции.
3. Воспроизведение прямой трансляции на десктопном приложении.

1.2 Анализ предметной области (про стриминг)

Стриминговый сервис – это источник стриминговых медиа, включающих в себя ТВ-шоу, фильмы и другие потоковые мультимедиа.

Стриминговые медиа – это видео или аудио контент, передаваемый в сжатом виде через Интернет и воспроизводимый немедленно, а не сохраняемый на жесткий диск [1].

В настоящее время потоковая передача основана на подключении к Интернету для воспроизведения медиафайлов и прямых трансляций. При потоковой передаче мультимедиа пользователю не нужно ждать загрузки файла для его воспроизведения. Поскольку носитель отправляется в непрерывном

потоке данных, он может воспроизводиться по мере поступления. Пользователи могут приостановить, перемотать назад или перемотать вперед, как они могли бы с загруженным файлом, если контент не транслируется в прямом эфире.

Эта технология появилась в начале 1990-х годов и была переведена на новый уровень в последние годы такими компаниями, как YouTube, Netflix, Spotify, Pandora и Hulu. Эти службы доставляют телевизионные шоу, фильмы, музыку и игры на компьютеры пользователей. Также данные услуги доступными и на мобильных устройствах. Благодаря тому, что смартфоны и планшеты становятся жизнеспособными вариантами вычислений, пользователи имеют больше возможностей выбирать, где и когда они могут смотреть потоковое мультимедиа. Этот вид потребления в пути создает благоприятную ситуацию и для бизнеса, поскольку вся мультимедийная информация может быть передана и доступна простым прикосновением пальца пользователя [1].

1.3 Выбор и обоснование используемого протокола

На сегодняшний день существует множество протоколов для передачи потокового видео. Их можно разделить на две основные группы: протоколы, использующие HTTP в качестве базового протокола, и протоколы с механизмами передачи данных и управления потоком.

1.3.1 Протоколы, использующие HTTP в качестве базового протокола

К данной группе относятся HTTP Live Streaming (HLS) и MPEG-DASH (Dynamic Adaptive Streaming over HTTP). Эти протоколы используют последовательное скачивание файла на клиентское устройство для воспроизведения потокового видео и HTTP для управления трансляцией. В большинстве своем данная группа протоколов использует TCP, что в свою

очередь вызывает большие задержки при передаче данных из-за механизмов проверки доставки пакета [2].

1.3.2 Протоколы с механизмами передачи данных и управления потоком

В число протоколов, относящихся к этой группе, входят Real Time Messaging Protocol (RTMP) и семейство протоколов, связанных с протоколом Real Time Protocol (RTP) – Real Time Streaming Protocol (RTSP), Real Time Transport Control Protocol (RTCP), Resource Reservation Protocol (RSVP). Каждый из этих протоколов вносит свой вклад при доставке видеотрафика.

Протокол RTP – IP-базированный транспортный протокол, который используется для передачи мультимедиаданных, таких, как видео и аудио, по сетям с коммутацией пакетов, в частности сети Internet. Данный протокол вместе с протоколами RTCP и RSVP способен обеспечить доставку и необходимый уровень Quality of Service (QoS) для мультимедиафайлов. RTP не включает функции маршрутизации, поскольку для этого применяется UDP (User Datagram Protocol) из стека протоколов TCP/IP [3].

Протокол RTP выполняет следующие функции:

1. Идентификация типа полезной нагрузки.
2. Нумерация последовательности пакетов.
3. Присвоение временных меток.

RTP выполняет ряд функций, свойственных уровню приложений, таких как:

1. Упорядочение пакетов во времени.

2. Реконструкция пакетов.
3. Синхронизация пакетов.

Вся необходимая для этого информация хранится в RTP-заголовке [3].

RTCP – протокол управления передачей. Данный протокол выполняет функции управления и поставляет RTP управляющую информацию для диагностики и оптимизации производительности. Среди множества функций основными являются следующие.

1. Мониторинг Quality of Service (QoS), управление перегрузками канала.
2. Идентификация источника.
3. Внутренняя синхронизация аудиовизуальной информации.
4. Управление масштабированием.

Отправитель, основываясь на пакетах обратной связи, генерируемых протоколом RTCP на узле получателя, может настроить скорость передачи данных. Эти пакеты также используются сетевым администратором для оценки производительности сети. RTCP-протокол также производит масштабирование информации, в результате чего ограничивается сетевой трафик.

RSVP – это протокол, резервирующий ресурсы по всему маршруту для своевременного и эффективного транспортирования аудио- и видеоданных. С этой целью он рассылает необходимые запросы на полосу пропускания коммутаторам, маршрутизаторам и другим сетевым устройствам [4].

RTSP – это протокол прикладного уровня. Назначение данного протокола в предоставлении конечному пользователю управления потоком. RTSP обеспечивает интерфейс, позволяющий выполнять такие операции, как

перемотка, прокрутка, пауза и останов. RTSP работает совместно с протоколами нижнего уровня, такими, как RTP, RSVP, IP и TCP/UDP. Протоколом RTSP предусматривается, что управление состоянием или связью должен осуществлять сервер. Данные в RTSP могут передаваться вне основной полосы другими протоколами, например, RTP [3].

1.4 Вывод

Так как низкие задержки при передаче данных являются более предпочтительными, чем механизмы проверки доставки пакетов, в качестве используемых протоколов были выбраны протоколы с механизмами передачи данных и управления потоком, а именно, RTP и RTSP.

2 Конструкторский раздел

В данном разделе рассматривается структура комплекса приложений, состоящего из Android-приложения, рестримера и десктопного клиента Windows.

2.1 Функциональная модель

На рисунке 1 изображена функциональная модель IDEF0 нулевого уровня.

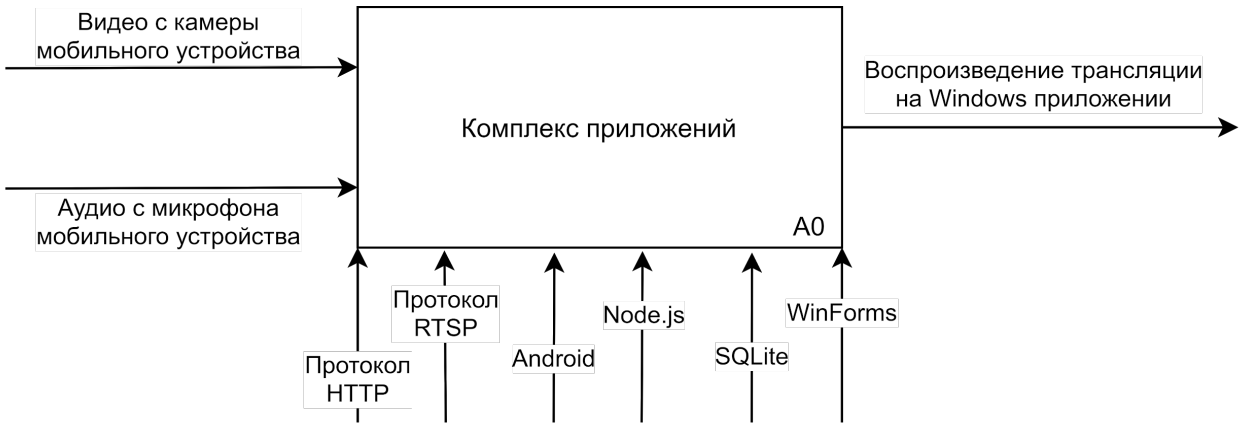

Рисунок 1 – IDEF0 нулевого уровня

На рисунке 2 изображена функциональная модель IDEF0 первого уровня.

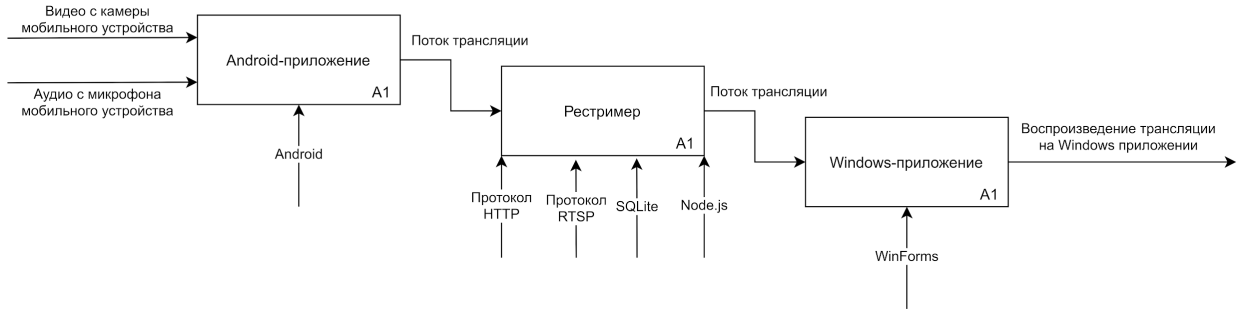

Рисунок 2 – IDEF0 первого уровня

2.2 Сценарий использования

На рисунке 3 представлена диаграмма взаимодействия пользователя с десктопным приложением.

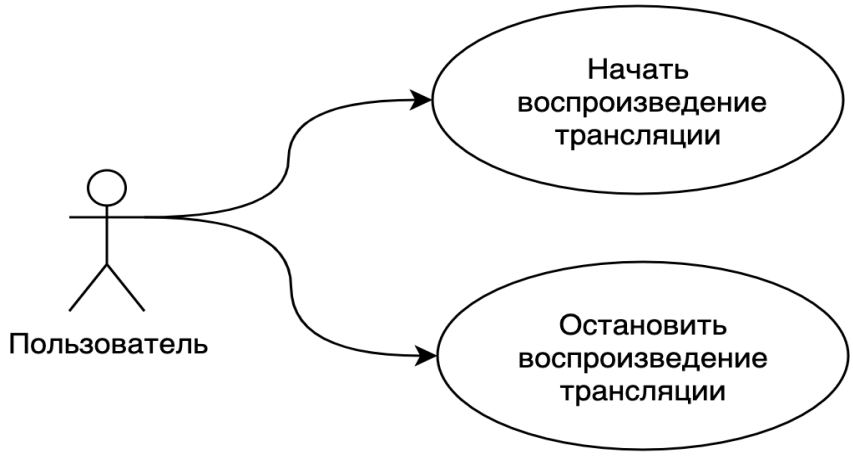

Рисунок 3 – UseCase диаграмма взаимодействия с десктопным приложением

На рисунке 4 представлена диаграмма взаимодействия пользователя с андроид приложением.

Рисунок 4 – UseCase диаграмма взаимодействия с андроид приложением.

2.3 Android-приложение

Необходимо провести обзор требуемых для реализации технологий и сравнение их между собой для выбора наиболее подходящих.

2.3.1 ОС Android

Android – операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнес-браслетов игровых приставок, ноутбков, нетбуков, смартбуков, очков Google Glass, телевизоров и других устройств.

Операционная система Android выбрана в качестве среды функционирования программно-аппаратного комплекса с целью удовлетворения потребностей целевой аудитории, в качестве главных критериев выбора приняты следующие.

- Возможность принимать видео и аудиоданные с видеоисточника (камера и микрофон смартфона).
- Возможность прямой трансляции видеопотока в Интернет.
- Доступность. Согласно данным портала StatCounter Global Stats, операционная система Android является самой распространённой, а значит, и самой доступной в 2020 году. Статистика распределения операционных систем для мобильных платформ в мире приведена на рисунке.

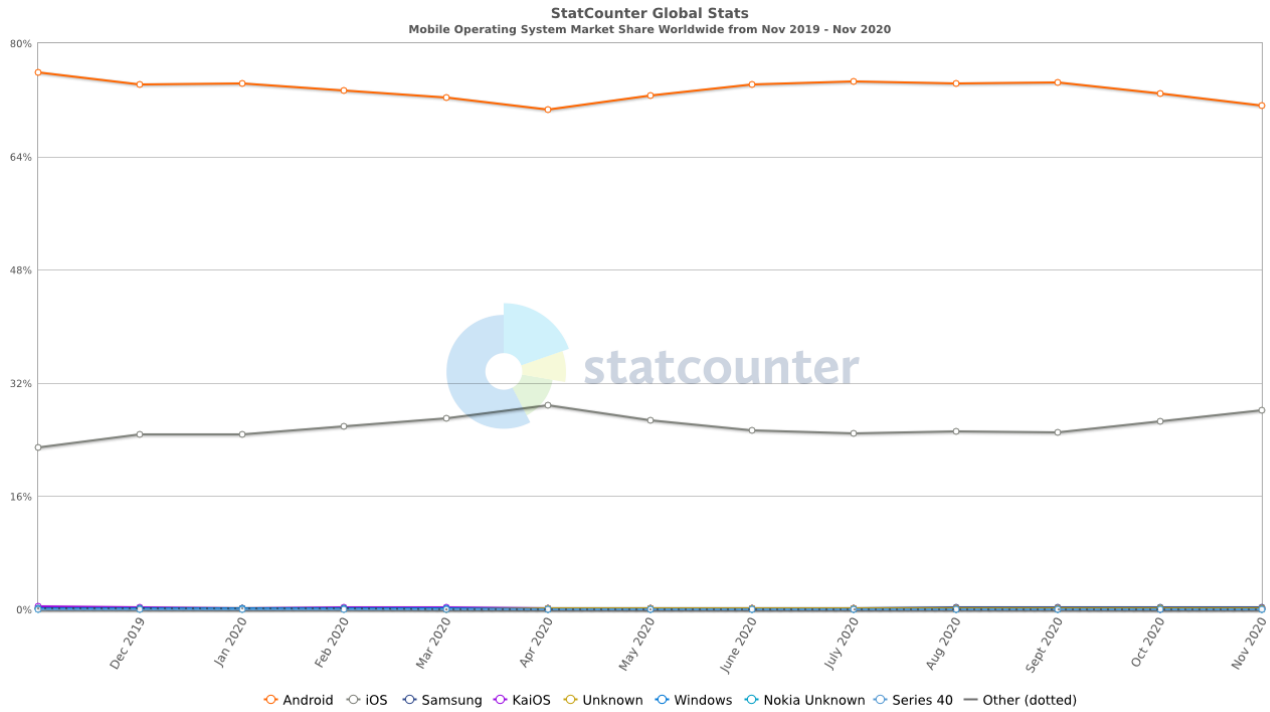

Рисунок 5 – Распределение рынка мобильных операционных систем по всему миру с декабря 2019 г. по ноябрь 2020 г. [5]

2.3.2 Формат видеоданных

Видео – это по существу трёхмерный массив цветных пикселей. Два измерения означают вертикальное и горизонтальное разрешение кадра, а третье измерение – это время. Кадр – это массив всех пикселей, видимых камерой в данный момент времени, или просто изображение.

Выбран формат MPEG-4 Part 14 (MP4) способный хранить мультимедиа – например, аудио, видео и субтитры. MPEG-4 – это формат, часто используемый при онлайн-потоковой передаче видео. Формат MP4/MPEG-4, выпущенный в 2001 году в соответствии со стандартом ISO/IEX 14496-1:2001, использует технологию кодирования AAC, которая предусматривает сжатие с потерями для недопущения копирования данных пользователем. [6]

2.3.3 Процесс работы приложения

Процесс.

1. Пользователь открывает приложение
2. Через дисплей пользователь управляет приложением – запускает и останавливает съёмку.
3. Система обрабатывает видео и аудио данные.
4. На дисплей выводится изображение.
5. Видеоизображение передаётся на сервер.

2.4 Ретранслятор

Ретранслятор должен передавать клиенту поток трансляции, полученной с Android-устройства.

2.4.1 REST API

На сервере рестримера используется REST API, написанное на Node.js. Данное API поддерживает следующие запросы:

- POST /stream – добавление трансляции
  В теле запроса отправляются параметры name (название трансляции) и url (ссылка на приходящий поток).

- GET /streams – получение списка трансляций
  Данный запрос возвращает все зарегистрированные трансляции с параметрами: id, name, url, дата создания, дата последнего обновления.

- DELETE /streams/:name – удаление трансляции
  Удаляет трансляцию с названием name.

2.4.2 Передача потока клиенту

Для передачи потока трансляции клиенту используется тип передачи данных unicast, который предполагает подключение одного клиента и отправки ему одностороннего потока данных. Если будет подключено больше одного клиента, то нагрузка на рестример увеличивается.

2.4.3 Процесс работы рестримера

Процесс работы рестримера должен выглядит следующим образом:

1. На рестример принимается RTSP-поток трансляции.
2. При помощи библиотеки FFmpeg поток делится на видео файлы, которые сохраняются на диск.
3. Используя HTTP API, клиент может получить ссылку на запущенную трансляцию.
4. При подключении клиента к потоку происходит сборка видео файлов в RTSP-пакет.
5. Рестример передает клиенту unicast пакеты по протоколу UDP, если по UDP не получается, то используется протокол TCP.

2.5 Desktop-клиент

Desktop-клиент должен воспроизводить онлайн-трансляцию, запущенную на смартфоне с установленной системой Android. Для этого ему необходима rtsp-ссылка на трансляцию. Процесс работы клиента должен выглядеть следующим образом.

1. Пользователь запускает клиент.

2. Пользователь выбирает rtsp-ссылку и нажимает на воспроизведение трансляции.

3. Клиент принимает данные с рестримера и выводит на экран видеоизображение трансляции, воспроизводя звук.

4. Клиент может остановить воспроизведение трансляции, нажав на специальную кнопку останова.

2.6 Вывод

В данном разделе были приведены функциональная модель, сценарий использования комплекса приложений, рассмотрена структура комплекса приложений, состоящего из Android-приложения, рестримера и десктопного клиента Windows.

3 Технологический раздел

В данном разделе производится выбор средств программной реализации, приводятся UML-диаграммы классов и интерфейс пользователя.

3.1 Выбор технологий

Начинать разработку необходимо с подбора инструментов, которые потребуются для реализации системы.

3.1.1 Java и Kotlin

Лучший способ разработки приложение для Android – это Android Studio.

Для разработки мобильных приложений на Android предпочтительным языком программирования является Java, так он поддерживается компанией Google и большинство приложений в Google Play построены именно на нем.

Сама Java была разработана компанией Sun Microsystems в 1995 году, и она до сих пор используется для широкого спектра программных приложений. Код Java выполняется виртуальной машиной, которая работает на устройствах Android и интерпретирует код.

Kotlin появился в качестве официального языка для разработки Android в 2017 году. Как и Java, Kotlin работает на виртуальной машине Java. Он полностью совместим с Java и не вызывает никаких препятствий или увеличения размера файлов.

Основное отличие заключается в том, что Kotlin требует меньше «шаблонного» кода, т.е. более простая для чтения система. Он также устраняет

такие ошибки, как исключение нулевого указателя, и даже освобождает от необходимости заканчивать каждую строку точкой с запятой. [7]

Рекомендованный современный подход к сборке программных проектов на базе ОС Android – это использовать системы сборки, самая удобная для использования из них – система Gradle. Файлы конфигурации Gradle написаны на языке Groovy, значит, необходимо ознакомиться также и с этим языком, и с самой системой сборки для корректного конфигурирования сборки реализуемой системы.

3.1.2 C#, WinForms и VLC Media Player

Для разработки Desktop-клиента, позволяющего просматривать онлайн-трансляцию, был выбран язык C#**.** C# – это активно развивающийся объектно-ориентированный язык со строгой типизацией, позволяющий создавать безопасные и надежные приложения.

Windows Forms («WinForms») – это библиотека классов GUI, включенная в .NET Framework. Это сложная объектно-ориентированная оболочка вокруг Win32 API, позволяющая разрабатывать настольные и мобильные приложения Windows, ориентированные на .NET Framework.

Для проигрывания онлайн-трансляции использовалась сторонняя библиотека от компании VideoLAN, позволяющая интегрировать медиапроигрыватель VLC Media Player в десктопные приложения. VLC Media Player – это свободный кроссплатформенный медиапроигрыватель, работающий на большинстве современных операционных систем и мобильных платформ, в частности Android, iOS, Tizen и Windows 10 Mobile.

3.1.3 Node.js + CoffeeScript

Node.js представляет среду выполнения кода на JavaScript, которая построена на основе движка JavaScript Chrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код. Node.js прежде всего предназначен для создания серверных приложений на языке JavaScript. Node.js использует событийно-ориентированную модель и неблокирующую ввод / вывод архитектуру, что делает его легковесным и эффективным. Node.js не является библиотекой. Это среда для выполнения JavaScript.

Для написания ретранслятора был использован язык программирования CoffeeScript. Это язык, который добавляет синтаксический сахар в JavaScript, который улучшает читаемость кода и уменьшает его размер. Код, написанный на CoffeeScript транслируется в JavaScript. Транслируемый код соответствует всем рекомендациям по написанию кода на JavaScript.

3.1.4 Git

Современная командная работа в среде программирования невозможна без системы управления версиями кода. Версионностью исходного кода проще управлять с помощью системы управления версий Git.

3.2 UML-диаграммы классов

На рисунке 5 представлена UML-диаграмма классов, где INetworkService – интерфейс, предоставляющий функциональность для нашего приложения.

- postVideoAsync(String, String) – метод, принимающий имя и url видео, публикует видео на сервер

- getVideoAsync() – метод, получающий все запущенные стримы.

- deleteVideoByNameAsync(String) – метод, удаляющий видео с сервера по имени.

NetworkService – класс, реализующий интерфейс INetworkService.

RestreamerApi – класс, выполняющий get, post, delete запросы на сервер.

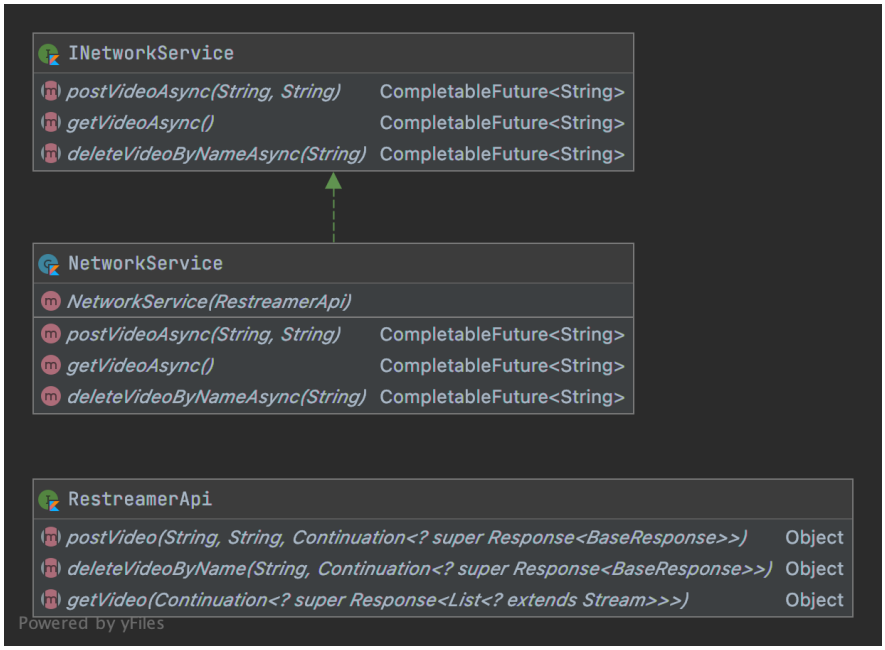

Рисунок 5 – UML диаграмма классов

В листинге 2 представлен код RestreamerApi.

На рисунке 6 представлена диаграмма классов, SelfVideoStreamPresenter – предоставляет методы для MainActivity.

- startStream() – начать стрим.
- stopStream() – остановить стрим.
- setCameraSurface(SurfaceTexture) – устанавливает текстуру для отображения изображения с камеры.
- isStreamStarted() – проверка начался ли стрим.

- openCamera(SurfaceTexture) – открывает камеру для использования и устанавливает текстуру.

SelfVideoStreamPresenter вызывает методы SelfVideoStreamService, который является реализацией интерфейса ISelfVideoStreamService. SelfVideoStreamService вызывает методы StreamingForegroundService.

Все зависимости внедряются с помощью Dependency injection – DI, процесс предоставления внешней зависимости программному компоненту.

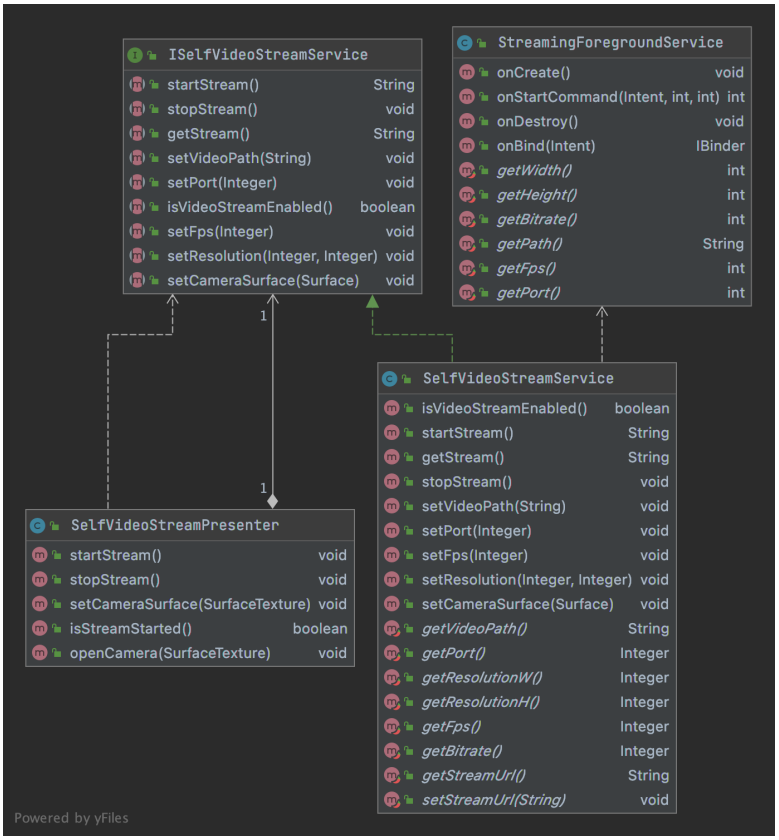

Рисунок 6 – UML диаграмма классов

В листингах 3 и 4 представлены реализации startStream и stopStream для StreamingForegroundService.

3.3 Интерфейс пользователя

При первом запуске приложения, необходимо спросить разрешения, представлены на рисунках 7, 8, 9.

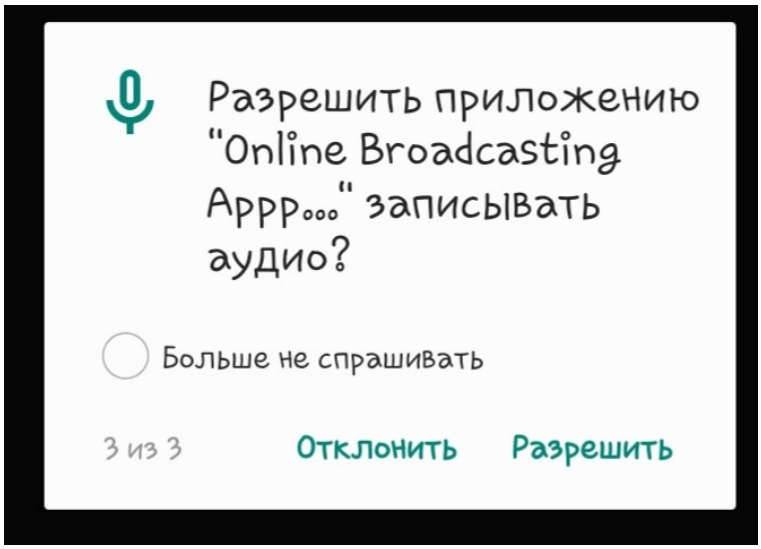

Рисунок 7 – Разрешение на запись аудио

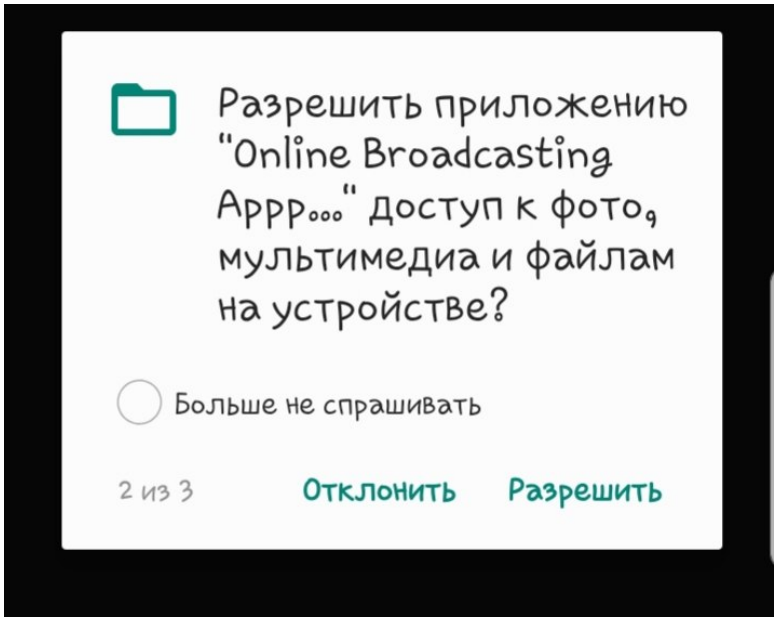

Рисунок 8 – Разрешение на доступ к файлам на устройстве

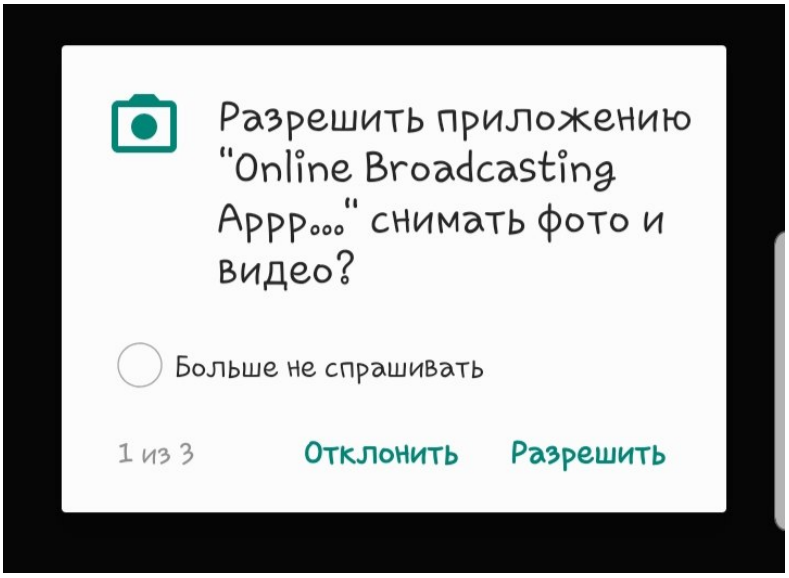

Рисунок 9 – Разрешение на съемку фото и видео

После разрешения виден начальный экран приложения, рисунок 10.

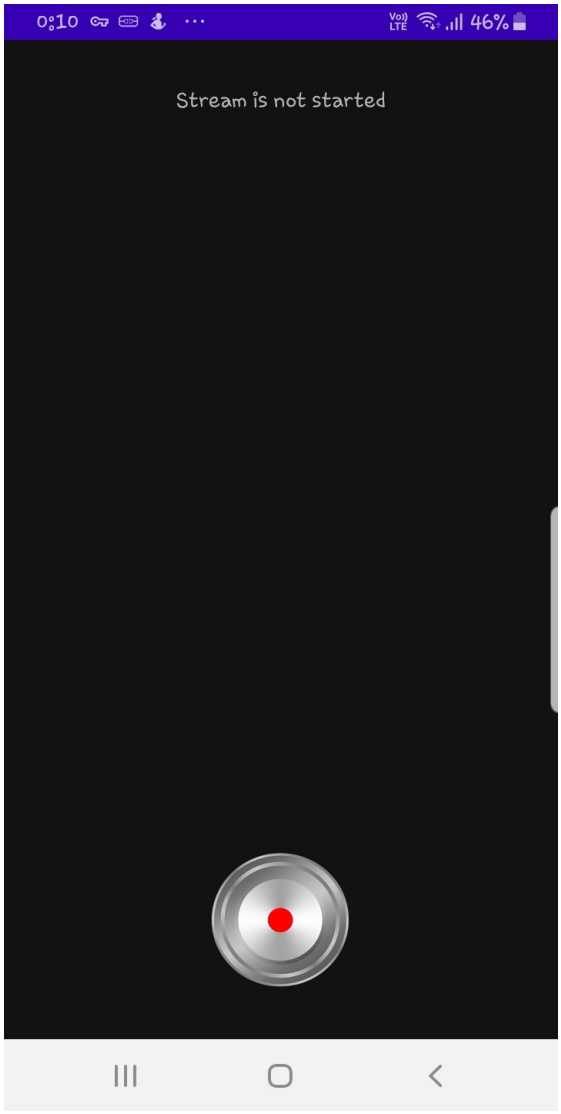

Рисунок 10 – Начальный экран андроид приложения

После начала записи появляется уведомление о работе на фоне и начинает транслироваться картинка с камеры, рисунок 11.

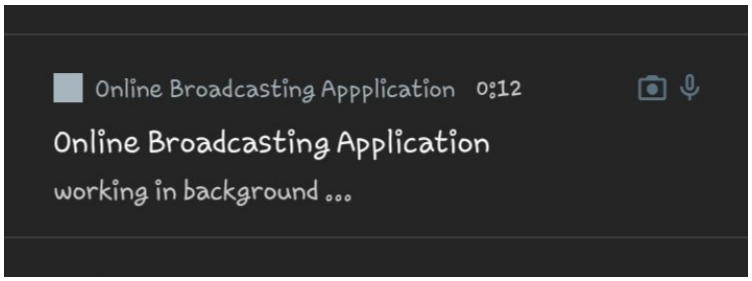

Рисунок 11 – Уведомление о работе на фоне

Запуск трансляции приложения представлен на рисунке 12.

Рисунок 12 – Трансляция

Результат get-запроса к серверу представлен на рисунке 13.

Рисунок 13 – Результат get-запроса

Десктопный клиент, начальный экран приложения представлен на рисунке 14.

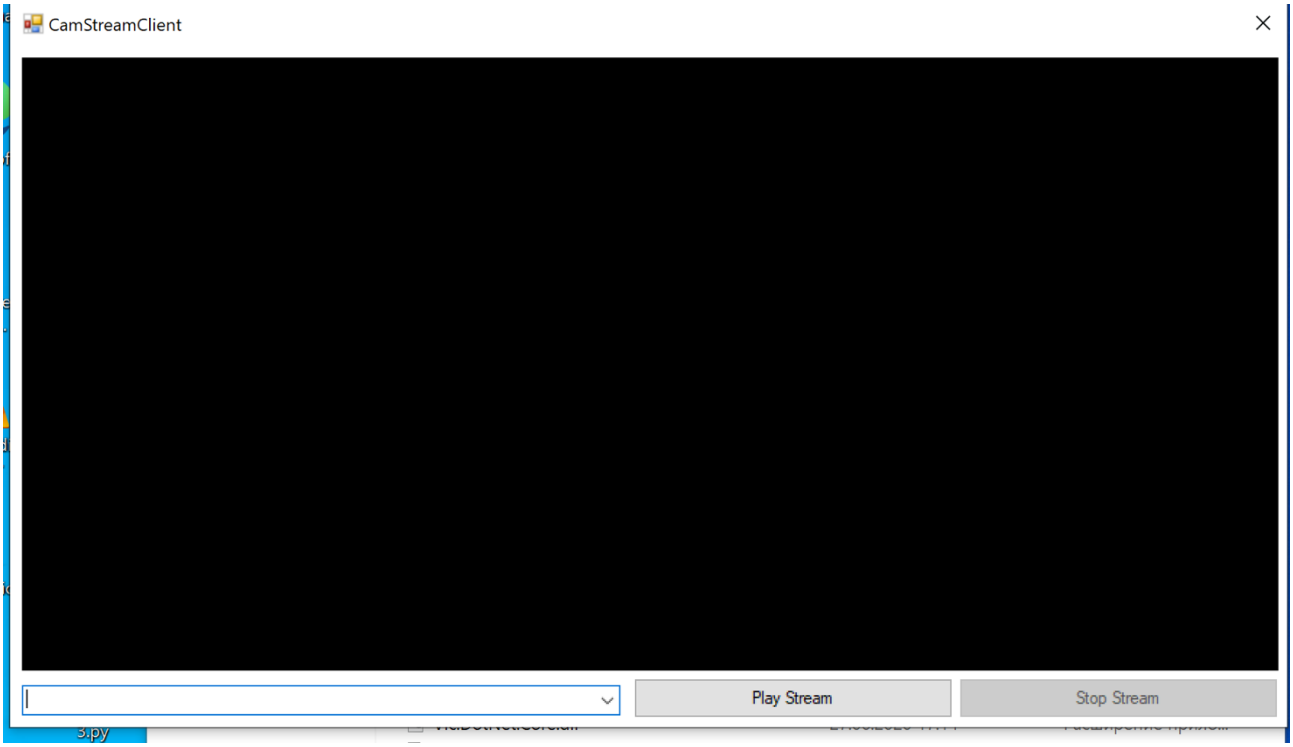

Рисунок 14 – Начальный экран десктопного приложения

После начала трансляции и запуска экран приложения представлен на рисунке 15.

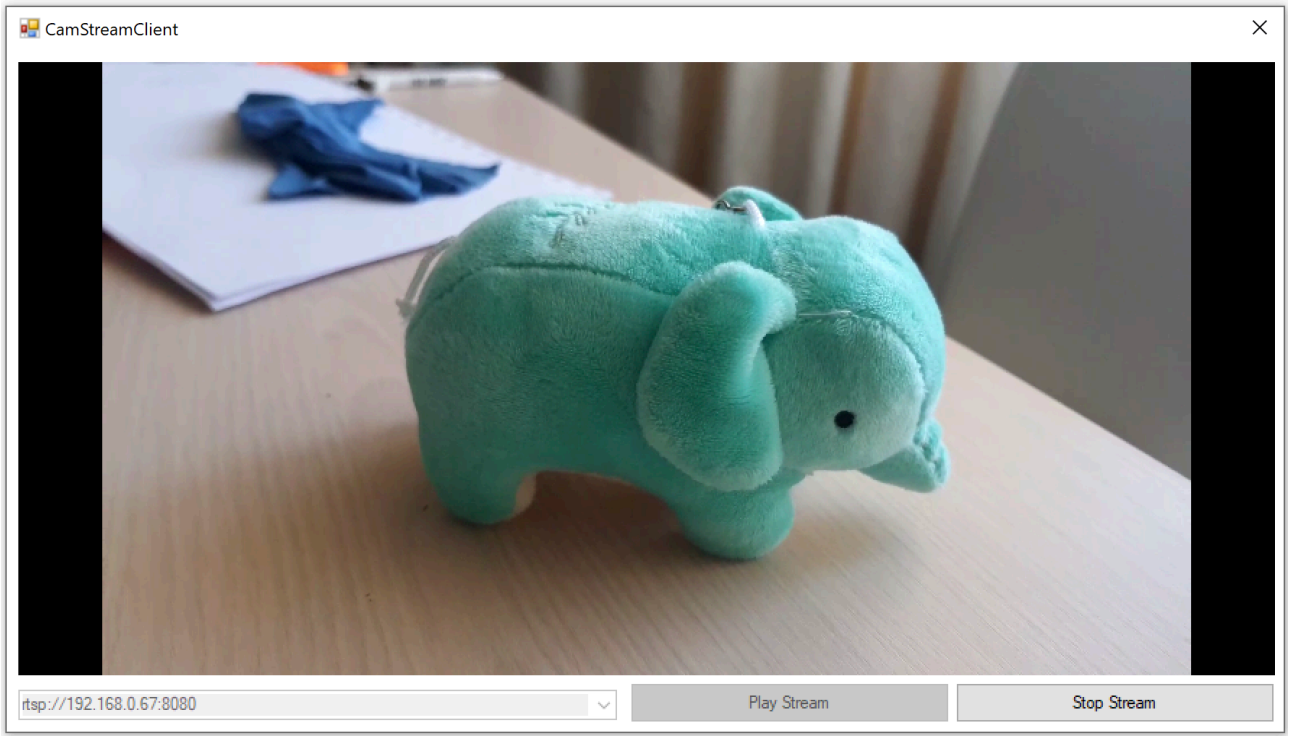

Рисунок 15 – Трансляция на десктопном приложении

3.4 Вывод

В данном разделе произведен выбор средств программной реализации, приведены структура комплекса приложений и интерфейс пользователя.

ЗАКЛЮЧЕНИЕ

В ходе данной работы были выполнены следующие задачи:

1) выбран протокола для потока трансляции;
2) выбраны средств для разработки;
3) разработано мобильное Android-приложения;
4) разработан рестример;
5) разработано десктопного приложения.

Таким образом достигнута цель – разработан программный комплекс для трансляции видеоизображения с камеры мобильного устройства на платформе Android на десктопное приложение компьютера под управлением Windows. Для передачи трансляции был использован протокол RTSP вместе с протоколом RTP. Для разработки были выбраны такие средства как: Java, Windows Forms, Node.js, CoffeeScript.

Список использованных источников

1. Седокова И. Ю. Разработка адаптивного дизайна видеостримингового сервиса. : – Томск, 2019.

2. Когда использовать протокол HLS // Boomstream.com [Электронный ресурс]. URL: https://boomstream.com/ru/news/using-hls.html (дата обращения: 08.12.2020).

3. Протоколы RTSP, RTP, UDP и TCP в системах видеонаблюдения // tdostup.ru [Электронный ресурс]. URL: http://tdostup.ru/index.php/stati/24-protocols/ (дата обращения: 08.12.2020).

4. Протокол резервирования ресурсов RSVP // opennet.ru [Электронный ресурс]. URL: https://www.opennet.ru/docs/RUS/inet_book/4/44/rsv_4496.html (дата обращения: 08.12.2020).

5. Mobile Operating System Market Share Worldwide // StatCounter [Электронный ресурс]. URL: http://gs.statcounter.com/os-market-share/mobile/worldwide# (дата обращения: 10.11.2020).

6. Формат файлов MP4 Video // online-convert.com [Электронный ресурс]. URL: https://www.online-convert.com/ru/file-format/mp4 (дата обращения: 30.11.2020).

7. Как выбрать язык программирования для создания Андроид — приложения // habr.com [Электронный ресурс]. URL: https://habr.com/ru/post/477578/ (дата обращения: 10.11.2020).

Приложение 1

```
class RestreamerService
  constructor: ->
    @ffmpeg_control = new FfmpegControl()
    @app = express()
    @app.use(bodyParser.urlencoded({ extended: true }))

    @app.post '/streams', @addStream
    @app.delete '/streams/:name', @deleteStream
    @app.get '/streams', @allStreams

    @app.listen(config.restreamerPort)
    logger.info ("Started Restreamer service on port #{config.restreamerPort}")

  addStream: (req, res) =>
    stream_name = req.body.name
    camera_url = req.body.url

    console.log "RestreamerService addStream stream_name: #{stream_name};
camera_url: #{camera_url}"

    unless stream_name?
      res.json({'err': "Null stream_name"})
      return
    unless camera_url?
      res.json({'err': "Null camera_url"})
      return

    db.Stream.create({
      'name': stream_name,
      'url': camera_url
    }).then (stream) =>
      @ffmpeg_control.launch_stream(stream)
      res.json({})
    .catch (err) ->
      res.json({'err': err})

  deleteStreamSub: (stream_name, on_result) =>
    unless stream_name?
      on_result({'err': "Null stream_name"})

    db.Stream.findOne({'where': {'name': stream_name}}).then (stream) =>
      @ffmpeg_control.stop_stream(stream)
      stream.destroy()
      on_result({})
    .catch (err) ->
      on_result({'err': err})

  deleteStream: (req, res) =>
    stream_name = req.params.name
    console.log "RestreamerService deleteStream stream_name: #{stream_name}"
    @deleteStreamSub(stream_name, (result) => res.json(result))

  allStreams: (req, res) =>
    db.Stream.findAll().then (streams) ->
```

```
      res.json(streams)
    .catch (err) ->
      res.json({'err': err})
```

Листинг 1. REST API

```
interface RestreamerApi {
    @POST("streams")
    @FormUrlEncoded
    suspend fun postVideo(@Field("name") name: String,
                          @Field("url") url: String): Response<BaseResponse>

    @DELETE("streams/{name}/")
    suspend fun deleteVideoByName(@Path("name") name: String):
Response<BaseResponse>

    @GET("streams")
    suspend fun getVideo(): Response<List<Stream>>
}
```

Листинг 2. Restreamer Api

```
private void startStream() {
    if (!_streamStartFlag) {
        initCamera();
        initFileWriter();
        initAudioCodec();
        initVideoCodec();
        if (_audioCodec != null && _videoCodec != null) {
            _audioCodec.start();
            _videoCodec.start();
        }
        initRtspServer();
        if (_rtspServer != null)
            _rtspServer.start();

        showNotification();
        updateCameraSurfaces();
        _camera.printCameraCharacteristics();
        try {
            _camera.startCapture(handler);
        } catch (CameraAccessException e) {
            Log.e(TAG, " Не удалось получить доступ к камере : " +
e.getMessage());
        }

        _streamStartFlag = true;
    }
}
```

Листинг 3. startStream

```
private void stopStream() {
    if (_streamStartFlag) {
        stopAll();
        _streamStartFlag = false;
    }
}
```

Листинг 4. stopStream