Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

IMPLEMENTACIÓN DE SERVICIO DE STREAMING ENTRE DISPOSITIVOS ANDROID

Alumno: **Francisco Manuel Pozo del Moral**

Tutor 1: Prof. D. José Manuel Pérez Lorenzo
Tutor 2: Prof. D. Juan Carlos Cuevas Martínez

Depto.: Ingeniería de Telecomunicación

Octubre, 2019

RESUMEN

El presente Trabajo Fin de Grado bajo el nombre ‘Implementación de servicio de *streaming* entre dispositivos Android’, persigue el objetivo principal de desarrollar una aplicación para dispositivos con sistema operativo Android que permita una comunicación de audio y video bidireccional entre usuarios, mediante varias codificaciones de audio y video.

Con esta aplicación usuarios de todo el mundo podrán mantener videoconferencias entre ellos en tiempo real. A su vez, la aplicación contará con un sistema de control de usuarios gracias a un servidor web, que se apoyará a su vez en una base de datos en la que almacenará la información de todos los usuarios.

ABSTRACT

The Final Degree Project with title ‘Implementation of a streaming service between Android devices’, has as main objective to develop an application for Android devices that allows a video and audio bidirectional communication between users. Furthermore, this application allows to choose some audio and video codecs.

Thanks that application users will be able to make videoconferences between them in real time. Moreover, this application provides of a user control system on a web server, that uses a database to store information about users.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

Hoy en día, el sistema operativo más extendido para teléfonos inteligentes (también conocidos como *smartphones*, su denominación en inglés) es Android[1]. En España, el 90% de los *smartphones* funcionan con Android. Su rápido auge desde que se lanzó la primera versión se debe a su gran comunidad de programadores, su amplia documentación y su licencia de código abierto. Dispone de una gran cantidad de aplicaciones en el mercado, del orden de dos millones , las cuales están disponibles a través de la plataforma *Google Play Store*. En comparación con la plataforma del sistema operativo iOS de Apple, en la plataforma Google Play Store podemos encontrar aproximadamente medio millón más de aplicaciones[2]. La filosofía de los *smartphones* es la de un terminal móvil conectado siempre a internet. Esto ha permitido que a través de las redes de datos móviles, los usuarios obtengan tasas de descarga cada vez más altas con latencias muy bajas.

En este proyecto se lleva a cabo el diseño, implementación y prueba de una aplicación para Android. La función principal de esta aplicación será la dar un servicio de *streaming* multimedia entre usuarios en tiempo real. Dicho servicio permitirá que los usuarios mantengan una llamada en la que podrán intercambiar audio y/o vídeo. Además en este proyecto también se incluye un servidor que se utilizará para controlar las sesiones de usuarios, y para dotar a dichos usuarios de los parámetros necesarios para iniciar, mantener y finalizar una llamada.

La comunicación entre usuarios se podrá realizar a través de una red local, en el caso de que los usuarios estuviesen en una misma LAN (en inglés *Local Area Network*), o bien se podrá realizar a través de una VPN (en inglés *Virtual Private Network*), en el caso de que estos usuarios estén en redes locales distintas. Además, mediante la utilización de una VPN, la comunicación entre usuarios sería segura, garantizado así autenticación, confidencialidad e integridad.

Para la comunicación del audio se utilizará el protocolo de la capa de transporte RTP (en inglés *Real Time Protocol*, Request for Comment 3550), que a su vez irá sobre el protocolo UDP (en inglés *User Datagram Protocol*, RFC 768). Para ello se dispone de la librería de Android a*ndroid.net.rtp, disponible en el paquete de Red..* Los codificadores que pueden ser usados son los siguientes:

- **Multi-tasa adaptativo** (en inglés *Adaptive Multi-Rate*, AMR, RFC 3267), que utiliza una tasa de bits desde los 4.75 hasta los 12.2 kbits/s. La frecuencia de muestreo utilizada es de 16 KHz. Este codificador está disponible en todas las versiones del sistema operativo de Android.

(1). https://www.xatakamovil.com/sistemas-operativos/asi-como-android-se-ha-comido-mercado-diez-anos

(2). https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

- **GSM** (en inglés *Global System for Mobile communications*) ***Full-rate*** códec, también conocido como GSM-FR, GSM 06.10, o simplemente FR. Fue el primer estándar de codificación de voz digital utilizado en el sistema de telefonía móvil GSM. La tasa de bit de este codificador es de 13 kbits/s. Este codificador está disponible en todas las versiones del sistema operativo de Android.
- **GSM Enhanced Full-rate**, también conocido como GSM-EFR, GSM 06.60, o simplemente EFR. Este estándar de codificación fue creado para mejorar la calidad de GSM FR. Presenta una tasa de bit de 12.2 kbits/s, a una frecuencia de muestreo de 8 KHz. Este codificador está disponible en todas las versiones del sistema operativo de Android.
- **G.711**, también conocido como Ley-A (usado en Europa y en el resto del mundo, excepto en Estados Unidos y Japón que se usa Ley-μ), que proporciona un flujo de datos de 64kbits/s, con una frecuencia de muestreo de 8 KHz. Este codificador está disponible desde la versión 4.1 del sistema operativo de Android en adelante.

Por otra parte, en la comunicación del vídeo los datos correspondientes a este servicio son enviados directamente sobre el protocolo de la capa de transporte TCP (en inglés *Transmission Control Protocol*, RFC 793). El codificador que puede ser usado es el siguiente:

- MJPEG (en inglés *Motion Joint Photographic Experts Group),* que consiste en un formato de compresión de video en el que cada trama de vídeo es codificada como una imagen JPEG. Este codificador enviará 15 tramas de vídeo por segundo con una resolución fija de 320x240.Mediante esta configuración se podrá asegurar que la transmisión de vídeo entre usuarios se produzca con fluidez. Este codificador está disponible en todas las versiones del sistema operativo de Android.

Para el intercambio de parámetros necesarios para establecer una comunicación entre usuarios, se utilizará el protocolo SIP (en inglés *Session Initiation Protocol*, RFC 2543), que se usará para invitar a usuarios a sesiones, y el protocolo SDP (en inglés *Session Description Protocol*, RFC 4566), que se usará para intercambiar parámetros tales como tipo de medio a intercambiar, protocolos de la capa de transporte a utilizar, codificadores disponibles de cada usuario, puertos de la capa de transporte a usar, etc.

Para desarrollar los servicios mencionados anteriormente, se van a utilizar las siguientes tecnologías:

- Sistema operativo Android para la aplicación móvil.
- HTTP (en inglés *HyperText Transfer Protocol*, RFC 7231*)* como protocolo de comunicación entre aplicación móvil y servidor.

- TLS (en inglés *Transport Layer Security,* 8446), como protocolo criptográfico entre aplicación móvil y servidor.
- Framework Spring para el desarrollo del servidor en Java.
- MySQL (en inglés *My Structured Query Language)* como gestor de base de datos en el servidor.
- Google Cloud como servicio de almacenamiento en la nube para el servidor web y la base de datos MySQL.

1.1 Alcance

El alcance del proyecto abarca el planteamiento, diseño e implementación de una aplicación móvil para sistemas operativos Android (desde la versión 4.0 hasta la versión actual 10.0) que permita mantener un servicio de *streaming* en tiempo real entre usuarios. Además, con la ayuda de un servidor web y una base de datos se pretende gestionar y controlar las sesiones de usuarios, y dotar a los usuarios de la información necesaria para iniciar una video-llamada en tiempo real.

El objetivo al diseñar esta aplicación es que llegue al mayor número de usuarios posibles. Distinguiendo entre uso personal y profesional, los usuarios podrán usar la aplicación entre ellos para uso personal. Para ello bastaría con que ambos usuarios compartieran una misma VPN, o se encontraran en la misma red local. El primer método dotaría a los usuarios de una mayor movilidad, pudiéndose encontrar cada uno en cualquier parte del mundo. En cuanto al ámbito profesional, sería más fácil que los usuarios se comunicaran compartiendo una misma red local, ya que medianas y grandes empresas disponen de redes locales de gran alcance.

1.2 .Estado del arte

En la actualidad existe una gran variedad de servicios de Internet que ofrecen realizar llamadas y vídeo llamadas, incluso éstas últimas han ido perfeccionándose para poder permitir a más de dos usuarios en una misma video-llamada. En la actualidad esta tecnología es de vital importancia en las comunicaciones enfocadas al sector empresarial y al personal, permitiendo mayor acercamiento en tiempo real entre personas a través de infraestructura Internet.

En este apartado se va a hablar de aplicaciones similares disponibles en el mercado para *smartphones* con sistema operativo Android que persiguen los objetivos del presente trabajo fin de grado.

1.2.1 Skype

Skype es una de las herramientas más utilizadas en reemplazo de las costosas comunicaciones telefónicas convencionales, ya que permite a través de una simple conexión a internet que diversos usuarios establezcan una conversación de voz y video en

tiempo real. Se trata de una aplicación de Voice Over IP (en inglés *Internet Protocol,* RFC 791) que nos ofrece la posibilidad de conectarnos por chat, voz o video-llamada, con individuos o empresas en cualquier parte del mundo. Puede ser utilizado en plataformas como Windows, Mac OS X, Android e iOS.

Figura 1.Skype. Fuente: Skype

1.2.2 Facetime

Facetime es una aplicación de mensajería propiedad de Apple que incluye vídeo, mensajes de texto, voz y más. Con esta herramienta los usuarios de Apple tienen la posibilidad de comunicarse a través de video-llamadas que puede operar tanto con redes WIFI (en inglés *Wireless Fidelity)* como con redes celulares 3G y 4G, ofreciendo así una flexibilidad que permite ser usada en cualquier escenario. Esta aplicación está solo disponible para sistemas operativos iOS y Mac OS X.

Figura 2.FaceTime.Fuente: FaceTime

1.2.3 Instagram

Instagram es una red social que permite a sus usuarios subir imágenes y vídeos con múltiples efectos fotográficos como filtros, marcos, colores retro, etc. para

posteriormente compartir estos contenidos en la misma plataforma. Una de las funcionalidades de esta aplicación es la video-llamada. Permite opciones como desactivar vídeo/audio, silenciar sonido propio o cambiar cámara frontal a cámara *selfie*. Además, permite realizar una llamada a varias personas a la vez. Esta aplicación está disponible para sistemas operativos Android e iOS.

Figura 3.Instagram. Fuente:Instagram

1.2.4 Google Meet

Google meet es una aplicación destinada a la videoconferencia en el ámbito profesional desarrollada por Google. Dicha aplicación permite videoconferencias grupales con un límite de participantes de hasta 30 personas simultáneamente. Está disponible en versión web y en sistemas operativos Android e iOS.

Figura 4. Google Meet. Fuente: Google Meet

1.3 Abreviaturas

1. AMR: Adaptive Multi-Rate
2. EFR: Enhanced Full Rate
3. ETSI:European Teleccomunications Standards Institute
4. FR: Full Rate
5. GSM: Global System for Mobile communications
6. GPS: Global Positioning System

7. HTTP: HyperText Transfer Protocol
8. IDE: Integrated Development Environment
9. IP: Internet Protocol
10. ITU-T:Telecomunication standardization sector of International Telecomunication Union
11. JDBC:Java Database Connectivity
12. JPEG: Joint Photographics Experst Group
13. JSON: JavaScript Object Notation
14. LAN: Local Area Network
15. MJEG: Motion Joint Photographic Experts Group
16. MOS: Mean Opinion Score
17. MySQL: My Structured Query Language
18. REST: Representational state transfer
19. RFC: Request for Comments
20. RTP: Real Time Protocol
21. SDP: Session Description Protocol
22. SIP: Session Initiation Protocol
23. TCP: Transmission Control Protocol
24. TLS: Transport Layer Security
25. UDP: User Datagram Protocol
26. UIT-T: Sector de normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones
27. VPN: Virtual Private Network

2 OBJETIVOS

En los objetivos del presente Trabajo Fin de Grado, cabe distinguir entre objetivos generales y objetivos específicos.

El objetivo general del presente TFG es el desarrollo de un servicio de *streaming* multimedia bidireccional entre dos extremos para dispositivos móviles Android. Para ello se realizará un estudio previo de las distintas opciones disponibles para usar en una aplicación nativa Android. Una vez implementado el servicio, se incluirá el estudio de distintas configuraciones de codificación de audio y vídeo, y el testeo bajo distintas condiciones de red.

Como objetivos específicos tenemos los siguientes:

- Realizar un control de usuarios a la hora del inicio de sesión.
- Facilitar a los clientes una lista de usuarios con los parámetros correspondientes para establecer una comunicación entre extremos.

3 MATERIALES Y MÉTODOS

En este apartado se informa sobre los requisitos de diseño que debe cumplir el sistema, las tecnologías empleadas para el desarrollo del sistema, y la estructura que forma el sistema.

3.1 Requisitos de diseño

Como requisitos de diseño para el desarrollo de la aplicación tenemos:

- La aplicación móvil deberá estar orientada a *smartphones* con sistema operativo Android.
- La aplicación deberá contemplar la posibilidad de usar distintas configuraciones de audio y video.
- La aplicación deberá funcionar en escenarios con usuarios que se encuentren en la misma red de área local, y en escenarios en los que los usuarios estén en distintas redes locales.

3.2 Tecnologías empleadas

3.2.1 Android

Android es un sistema operativo ideado en su origen para teléfonos móviles del mismo modo que iOS, Windows Phone o BlackBerry. A diferencia de estos, Android está basado en el núcleo del sistema operativo libre Linux, el cual es gratuito y multiplataforma.

El S.O. Android proporciona interfaces necesarias para desarrollar aplicaciones que puedan acceder a funciones del teléfono (como el GPS, en inglés *Global Positioning System,* la agenda de contactos…), de manera sencilla, a través del lenguaje de programación Java.

Android fue desarrollado por Android Inc, estudio prácticamente desconocido hasta que Google lo adquirió en 2005. Ese mismo año comienzan a trabajar en la creación de una máquina virtual Java optimizada para dispositivos móviles.

En noviembre de 2007 se lanza una primera versión del Android SDK. Al año siguiente apareció el primer teléfono con Android. En el año 2010 llegaría la consolidación de Android, como uno de los sistemas operativos móviles más usados.

3.2.2 Android Studio

Android Studio es el IDE (en inglés *Integrated Development Environment*, o *Entorno de Desarrollo Integrado* en español) desarrollado por Google utilizado en este trabajo para la creación de la aplicación Android. Es un entorno optimizado para el desarrollo de Android y ofrece una gran variedad de herramientas y opciones tales como:

- Renderización en tiempo real de la aplicación.
- Construcción basada en Gradle, herramienta de automatización de construcción de código.

- Arreglos rápidos para errores de código.
- Plantillas para crear diseños comunes de Android.
- Soporte para programar aplicaciones para Android Wear, versión de Android para dispositivos wearables como relojes inteligentes.
- Emulación en dispositivo virtual o físico de las aplicaciones desarrolladas.

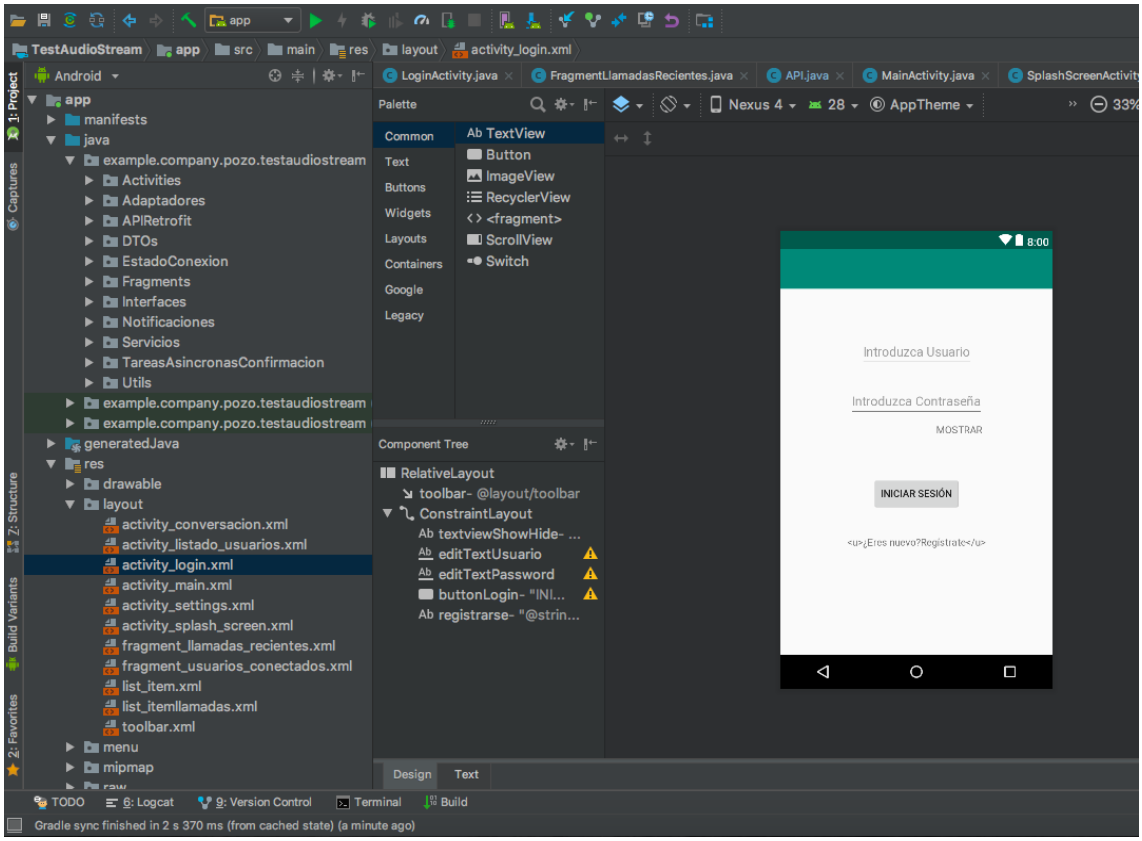


Figura 5. Entorno de desarrollo Android Studio

3.2.3 MYSQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por OracleCorporation y está considerada como el sistema de base de datos OpenSource más popular del mundo

En nuestro proyecto hemos optado por utilizar MySQL como opción de almacenamiento de todos los datos de la aplicación: esto significa que utilizaremos dicho gestor de base de bases de datos con el fin de que guarde toda la información relacionada con usuarios, tales como credenciales y datos personales, y con parámetros para establecer una comunicación, tales como dirección IP local de cada usuario, puerto de escucha de mensajes SIP o la fecha de inicio de sesión de cada usuario.

El hecho de que hayamos elegido esta tecnología se debe a diversos factores:

- La disponibilidad de una interfaz de conexión a base de datos MySQL compatible con Java, denominada JDBC (en inglés *Java Database Connectivity*), y optimizada en el *framework* Spring en la clase JdbcTemplate.

- La alta popularidad de la que goza en Internet y el mundo de la computación en general.
- La capacidad de gestionarla y diseñarla fácilmente mediante las herramientas de asistencia MAMP.

3.2.4 Intercambio de datos- JSON

JSON (en inglés *JavaScript Object Notation)* es un tipo de formato para la representación de datos siguiendo una sintaxis específica, cuya función fundamental consiste en que diferentes aplicaciones puedan intercambiar información independientemente de la plataforma o lenguaje en el que han sido desarrolladas. En nuestro proyecto hemos optado por JSON como intercambio de datos entre el cliente y servidor web. Esto se debe a los siguientes factores:

- Su análisis sintáctico, y su generación son relativamente sencillas.
- Spring Framework utiliza en su última versión JSON como formato estándar de intercambio de datos en desarrollo de servicios REST (en inglés *representational state transfer*).

3.2.5 Java

Java es un lenguaje de programación de propósito general y de alto nivel. Es concurrente (un mismo programa puede generar varios procesos que se ejecutan simultáneamente) y orientado a objetos (el programa está compuesto por entidades que interactúan entre ellas por medio de mensajes que modifican su estado).

3.2.6 Eclipse

Eclipse es un IDE (en inglés *Integrated Development Environment*). Es un software que proporciona diversas herramientas para facilitar la tarea de los desarrolladores de aplicaciones software en cualquier lenguaje de programación. Entre las distintas herramientas, Eclipse incluye un compilador, un intérprete, un editor de código fuente y un depurador. Está desarrollado por completo en lenguaje Java, por lo que es necesario tener un JRE instalado previamente en el sistema.

En nuestro proyecto Eclipse va a ser utilizado para desarrollar el servidor web mediante Spring Framework, del que hablaremos a continuación.

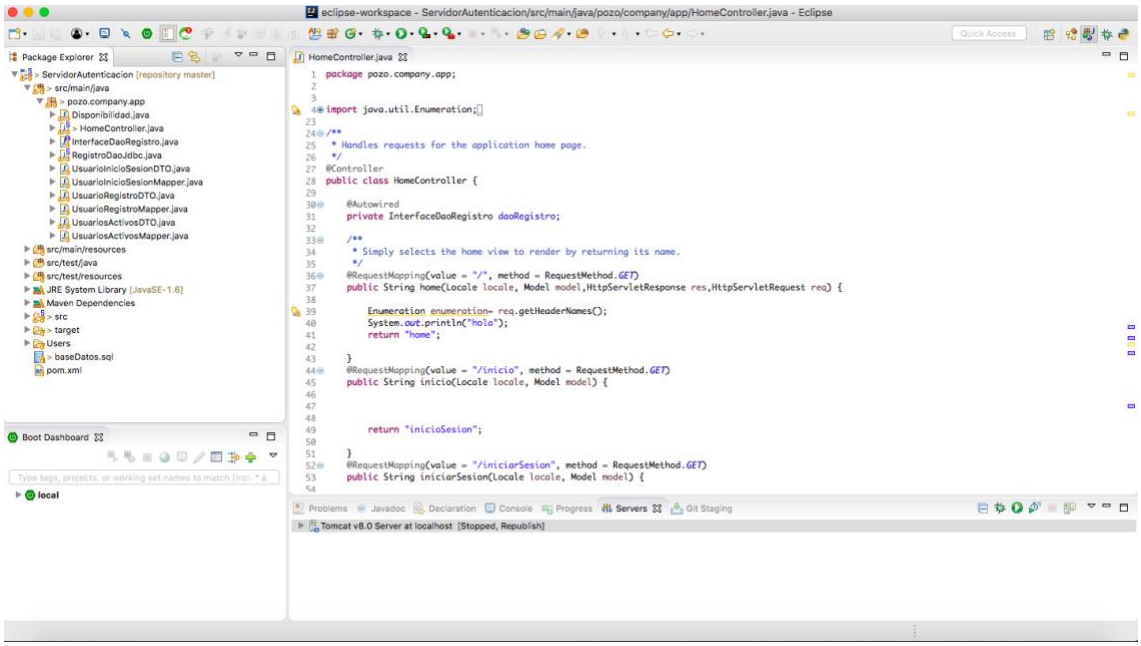


Figura 6. Entorno de desarrollo Eclipse

3.2.7 Spring Framework y servicios Rest.

Spring es uno de los *frameworks* más utilizados en la actualidad para Java empresarial. Su finalidad es la de estandarizar, agilizar, manejar y resolver los problemas que puedan ir surgiendo en el trayecto de la programación. Permite desarrollar aplicaciones de manera más rápida, eficaz y corta, ahorrando líneas de código.

Los servicios REST son utilizados entre sistemas que usan HTTP para recoger datos o producir operaciones sobre dichos datos en formatos como XML o JSON. Las características más importantes de los servicios REST son las siguientes:

- Cada método del protocolo HTTP es utilizado para un servicio concreto: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- Los objetos en REST siempre se manipular a partir de la URI.
- En cada petición HTTP se incluye la información necesaria para ejecutarse, de manera que ni cliente ni servidor necesitan guardar ningún estado previo a la petición.

3.2.8 Google Cloud Platform

Google Cloud Platform es una plataforma de Google que ofrece servicios tales como computación en la nube, acceso a Big Data, soluciones de almacenamiento y recursos de Inteligencia Artifical y *Machine Learning*. Presenta las siguientes ventajas:

- No se necesita una infraestructura propia, debido a que se puede acceder a todos los servicios a través de la plataforma de Google.

- Ofrece un modelo de pago por uso, en el que se pagará por los servicios utilizados.
- Ofrece características de escalabilidad.

3.3 Arquitectura del sistema

El sistema está formado por dos elementos, un servidor y un cliente. El servidor se podrá ejecuta en una máquina remota, y el cliente estará instalado en un *Smartphone* Android con conexión a Internet. Dentro de la arquitectura del sistema, encontramos dos partes. La primera parte es la siguiente:

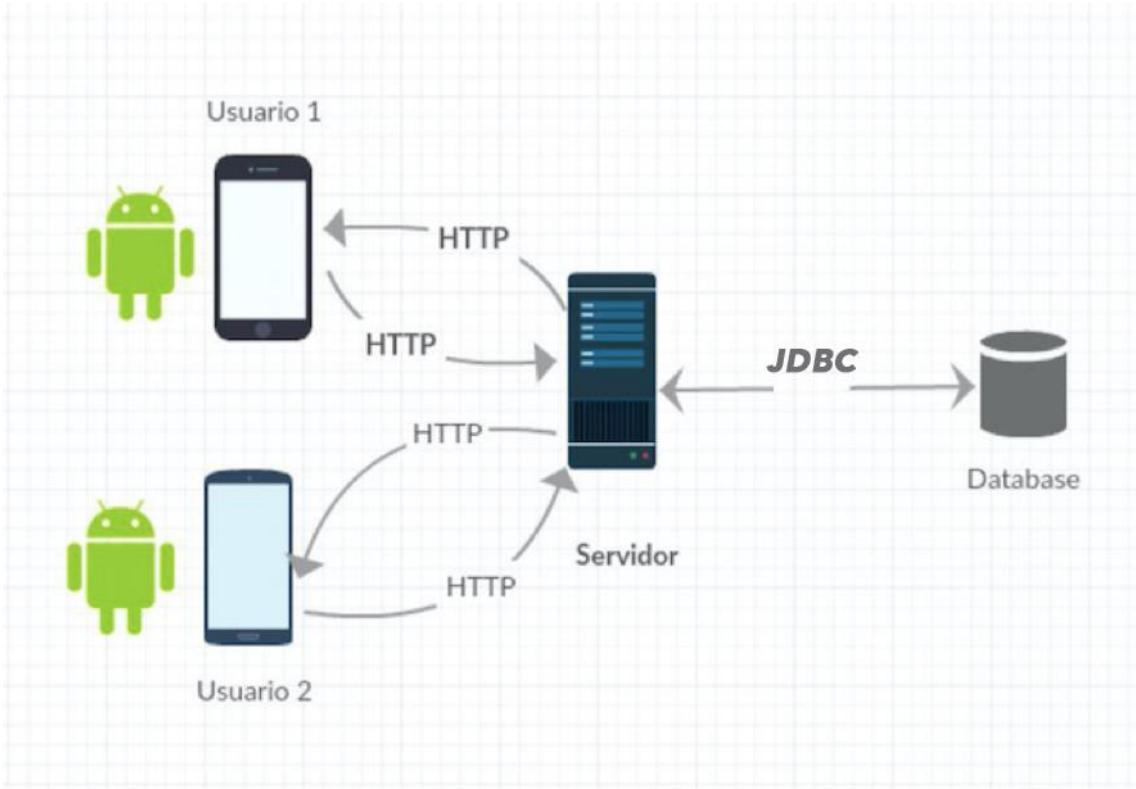


Figura 7. Arquitectura del sistema entre cliente-servidor

En la Figura 7 se puede observar la primera parte de la arquitectura del sistema. A la izquierda se encuentra el cliente, instalado en un Smartphone con sistema operativo Android. A la derecha se encuentra el servidor, que mediante el protocolo HTTP atenderá peticiones por parte de los clientes. Dicho servidor mantiene conexión con la base de datos MySQL, mediante la cual podrá realizar consultas y obtener la información correspondiente para contestar a las peticiones de los usuarios. Tanto el servidor como la base de datos MySQL se encuentran almacenados en la plataforma Google Cloud, de esta manera los clientes pueden acceder desde cualquier red con conexión a Internet a los recursos que ofrece el servidor.

Esta situación es un escenario típico cliente-servidor, el cliente realiza peticiones al servidor y éste enviará las respuestas pertinentes. El servidor tiene los siguientes servicios disponibles:

- Control de inicio de sesión de usuarios.
- Lista de usuarios con su disponibilidad.
- Control de cierre de sesión de usuarios.
- Almacenamiento de parámetros de usuarios tales como dirección IP local, puerto de escucha de mensajes SIP, puerto de escucha de mensajes ACK, disponibilidad, fecha de inicio de sesión.

El cliente pide al servidor iniciar sesión. Para ello, enviará sus credenciales (usuario y *password)*, su disponibilidad a la hora de poder tener comunicaciones con otros usuarios (por defecto la disponibilidad será “disponible”), su dirección IP de la red local, su dirección IP de la interfaz VPN (en el caso de que no disponga interfaz VPN, se enviará un valor nulo) el puerto en el que va a recibir mensajes SIP y el puerto en el que recibirá mensajes ACK (en inglés *acknowledgement*), utilizados para llevar un control de la comunicación entre usuarios.

En las peticiones posteriores a las del inicio de sesión, el cliente enviará en el mensaje HTTP dos cabeceras de tipo *cookie* indicando su nombre de usuario y su contraseña, para que el servidor reconozca a cada usuario en todo momento. El cliente podrá realizar peticiones como pedir una lista de los usuarios que han iniciado sesión. Esta lista podrá pedirla en cualquier momento, de manera que un cliente podrá tener actualizada la lista de usuarios disponibles, así como sus parámetros actualizados. En este caso, el servidor contestará con el nombre de cada usuario, su disponibilidad, su dirección IP de red local, su dirección IP de interfaz VPN y sus puertos de escucha de mensajes SIP y de mensajes ACK. Una vez recibida esta lista de usuarios, el cliente podrá comenzar un intento de llamada a otro usuario, ya que conocerá todos los parámetros necesarios para comenzar dicha llamada.

Otras peticiones que el cliente podrá realizar son las de cerrar sesión, y modificar su disponibilidad.

La segunda parte de la arquitectura del sistema es la siguiente:

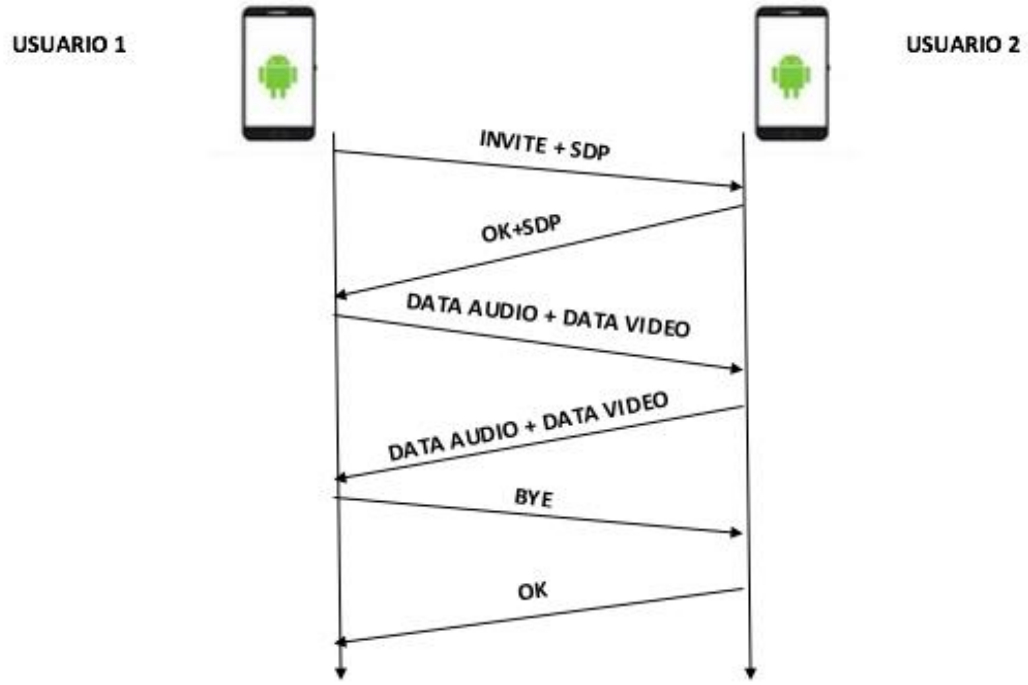


Figura 8. Arquitectura del sistema entre usuarios

Este escenario se produce cuando un usuario (que previamente haya iniciado sesión) intenta llamar a otro usuario para mantener una conversación de audio y/o vídeo. El primer paso es llamar a dicho usuario. Para ello se envía un mensaje INVITE del protocolo SIP junto a un mensaje SDP que contiene parámetros necesarios como tipo de medio a compartir, puerto para cada tipo de medio, codificadores disponibles, duración de la llamada, etc.

En el caso de que el segundo usuario acepte la llamada, éste envía su respectivo mensaje SIP junto al mensaje SDP. Una vez recibido éste mensaje, se produce la comunicación de audio y de vídeo a través del protocolo RTP. La comunicación terminaría en el caso en que un usuario quiera terminar la llamada, o en el caso en el que un usuario pierda la conexión a Internet.

3.4 Base de datos

La base de datos MySQL utilizada por el servidor está formada por dos tablas:

- RegistroUsuarios: almacena datos personales de los usuarios, tales como Nombre, Apellidos, Nombre de Usuario, Teléfono, Email y Clave.
- SesiónUsuarios: almacena datos relativos a la sesión del usuario, tales como Nombre de Usuario, Dirección IP local, Puerto de escucha de

mensajes SIP, Puerto de escucha de mensajes ACK, Disponibilidad y Fecha de inicio de sesión.

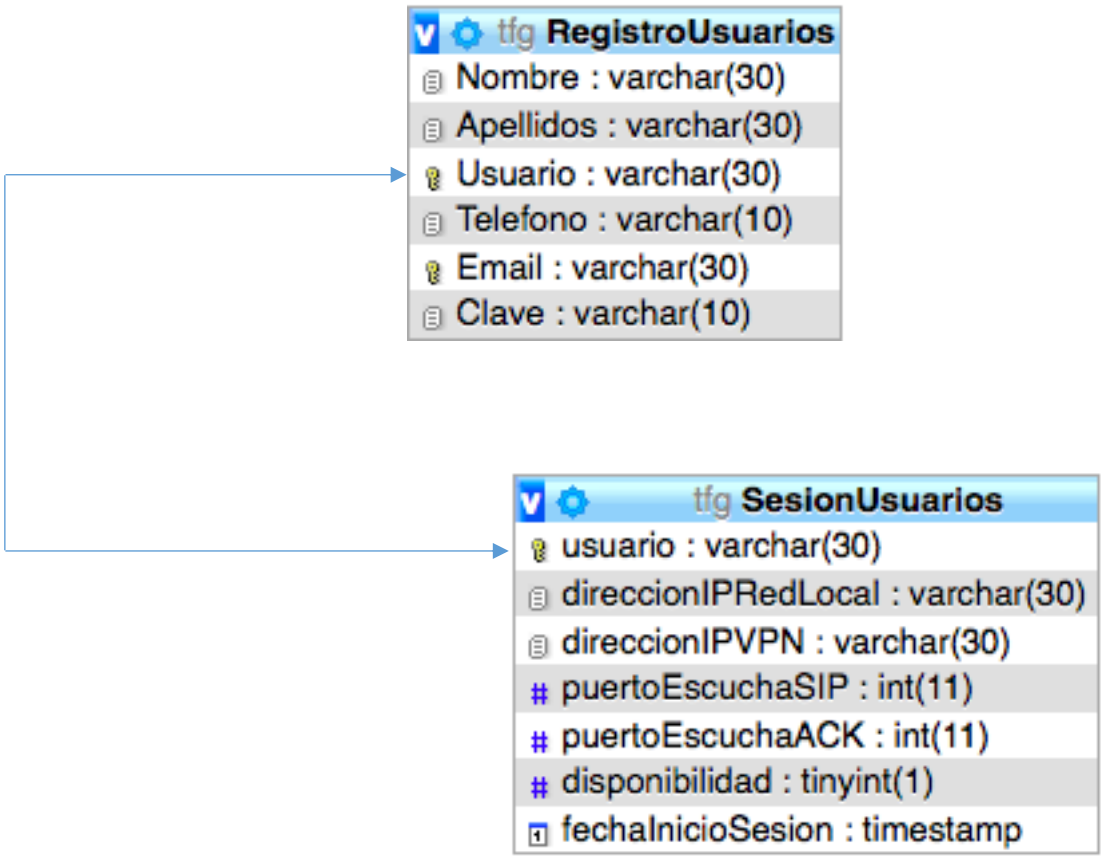


Figura 9. Esquema de tablas y atributos de la Base de Datos

A continuación, se incluyen algunas consultas realizadas sobre las tablas *RegistroUsuarios* y *SesionUsuarios.*

```
mysql> select * from RegistroUsuarios;
+---------+-----------+---------+-----------+-------------------+--------+
| Nombre  | Apellidos | Usuario | Telefono  | Email             | Clave  |
+---------+-----------+---------+-----------+-------------------+--------+
| daniel  | leon leon | daniel  | 123456    | daniel@gmail.com  | 123456 |
| fran    | pozo      | fran    | 123456    | fran@gmail.com    | 1234   |
| luis    | pozo      | luis    | 12345     | fran@gmail.com    | 1234   |
| ivan    | lopez     | ivan    | 12345     | ivan@gmail.com    | 123456 |
| juanfra | laserna   | juanfra | 12345     | juanfra@gmail.com | 123456 |
+---------+-----------+---------+-----------+-------------------+--------+
5 rows in set (0.00 sec)
```

Figura 10. Consulta sobre tabla RegistroUsuarios

```
mysql> select * from SesionUsuarios;
+---------+--------------------+---------------+----------------+-----------------+---------------+---------------------+
| usuario | direccionIPRedLocal | direccionIPVPN | puertoEscuchaSIP | puertoEscuchaACK | disponibilidad | fechaInicioSesion   |
+---------+--------------------+---------------+----------------+-----------------+---------------+---------------------+
| ivan    | 192.168.1.129      | a             |          38298 |           42285 |             0 | 2019-10-16 08:27:37 |
| juanfra | 192.168.1.132      | a             |          40609 |           38466 |             0 | 2019-10-16 08:27:38 |
+---------+--------------------+---------------+----------------+-----------------+---------------+---------------------+
2 rows in set (0.00 sec)
```

Figura 11. Consulta sobre tabla SesionUsuarios

3.5 Desarrollo de la aplicación móvil

En este apartado se verán las clases que componen la aplicación y los aspectos más relevantes en la implementación Java. A grandes rasgos, la aplicación cuenta con siete Actividades, dos Fragmentos, cuatro tareas asíncronas, dos hebras y un Servicio y un *Broascast Receiver.*

3.5.1 Descripción de interacciones de clases Java para servicios de la aplicación

A continuación, se va a describir las interacciones entre las clases Java para los servicios más importantes que ofrece la aplicación.

3.5.1.1 Servicio Audio

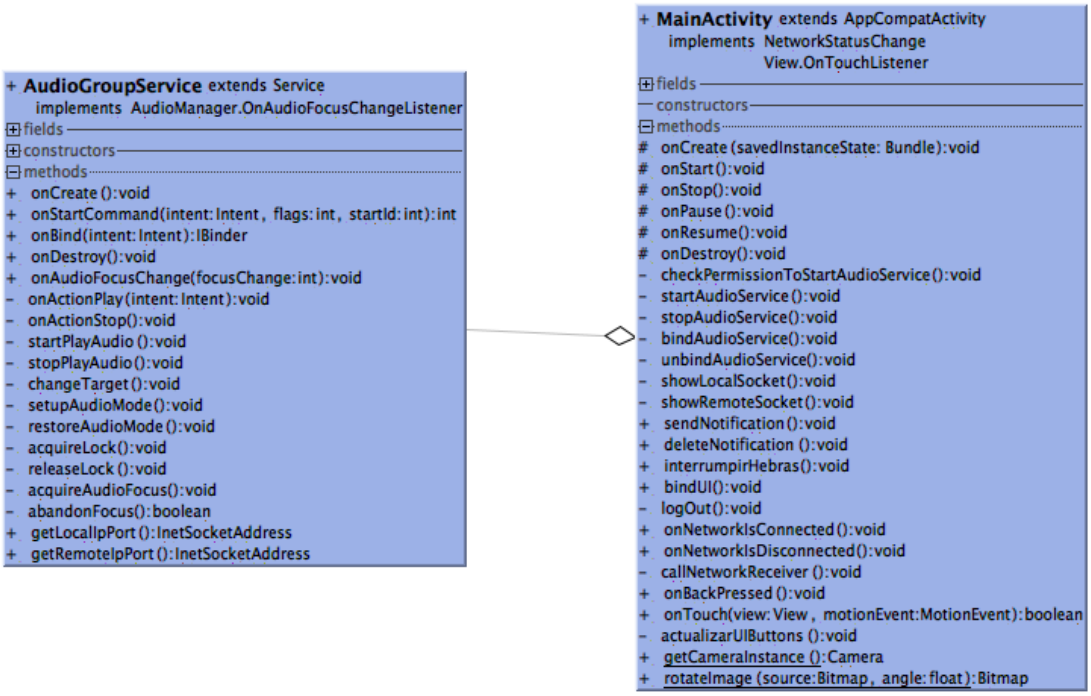


Figura 12. Relación entre clases para servicio de Audio

3.5.1.2 Servicio de generación de vídeo

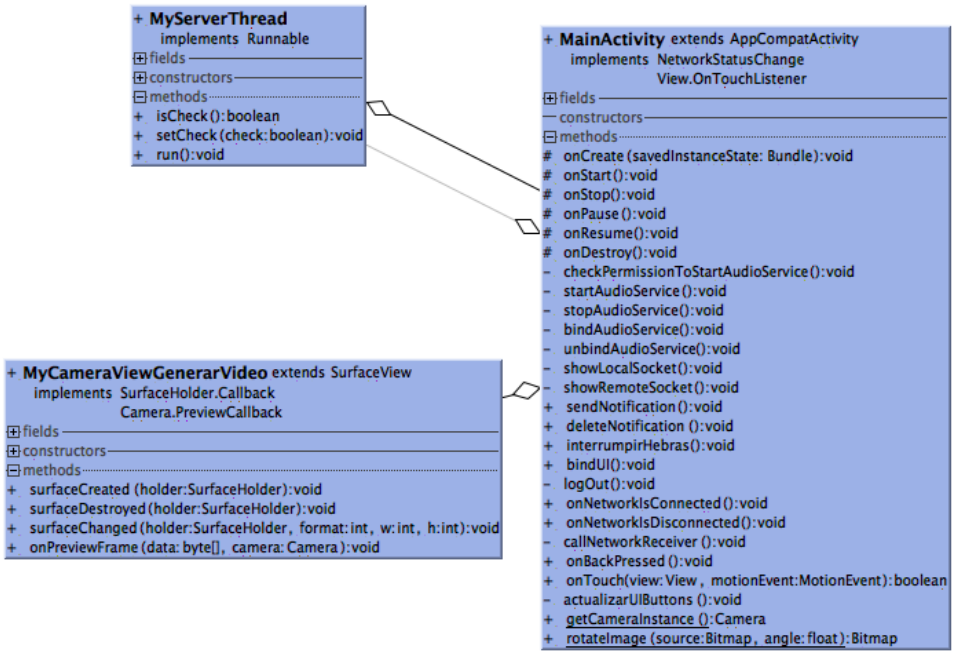


Figura 13. Relación entre clases para servicio generación de vídeo

3.5.1.3 Servicio de recepción de vídeo

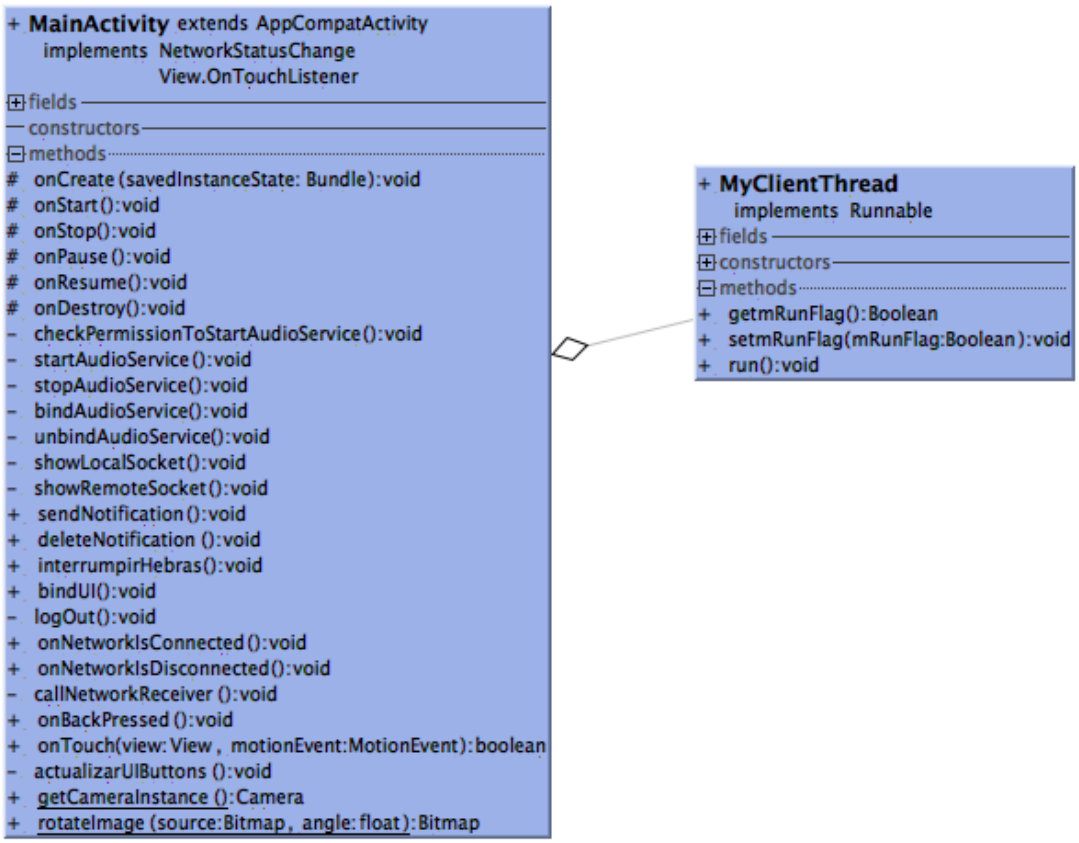


Figura 14. Relación entre clases para servicio recepción de vídeo

3.5.1.4 Servicio de estado de conexión

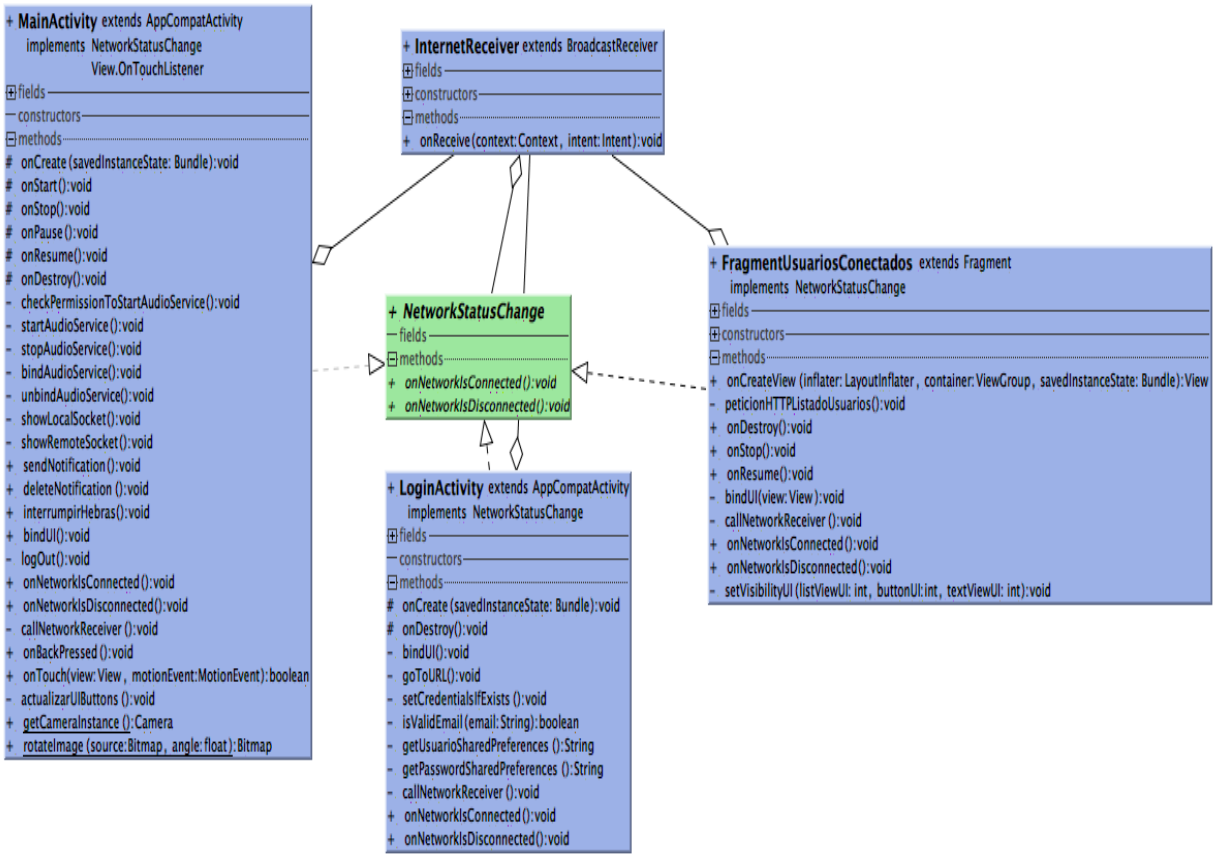


Figura 15. Relación entre clases para servicio estado de conexión

3.5.2 Descripción de clases Java de la aplicación

A continuación se va a describir brevemente el funcionamiento de cada clase Java de la aplicación móvil.

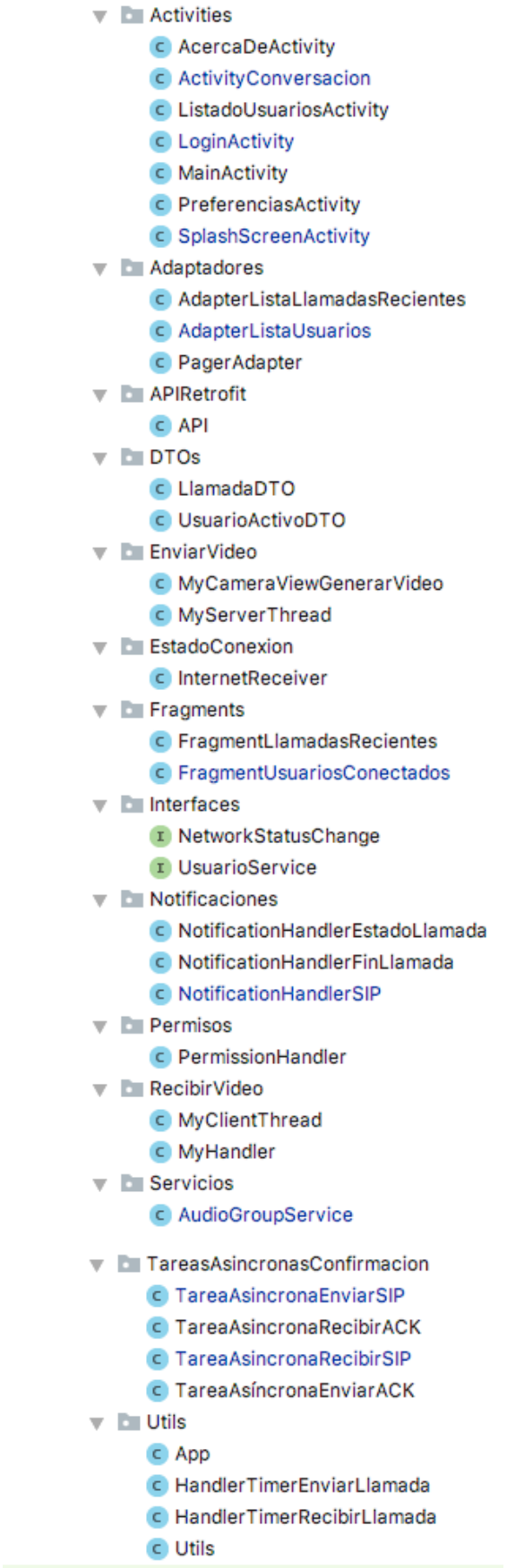


Figura 16. Estructura de clases

3.5.2.1 AcercaDeActivity.java

Esta clase será la encargada de mostar por la interfaz de usuario los datos correspondientes a la aplicación.

3.5.2.2 SplashScreenActivity.java

Esta clase será la encargada de comprobar si en las *SharedPreferences* existen credenciales almacenadas. En el caso de existir, se producirá intento de inicio de sesión automático con dichas credenciales. Si el inicio de sesión no es satisfactorio o no existen credenciales, ésta actividad nos dirige a la actividad *ListadoUsuariosActivity*. Para realizar el inicio de sesión, se utilizará la clase API.java y la interfaz UsuarioService.java. Mientras se produce la comprobación de las credenciales en las *SharedPreferences* y el inicio de sesión, se mostrará por la interfaz de usuario un fondo de pantalla que durará 2 segundos.

3.5.2.3 LoginActivity.java

Esta clase será la encargada de solicitar al usuario mediante un formulario las credenciales para iniciar sesión. Una vez recogidas las credenciales, realiza el inicio de sesión. Si éste es satisfactorio, esta actividad nos dirige a la actividad *ListadoUsuarios.* Para realizar el inicio de sesión, se utilizará la clase API.java y la interfaz UsuarioService.java.

3.5.2.4 ListadoUsuariosActivity.java

Esta clase estará compuesta a su vez por dos fragments: *FragmentLlamadasRecientes* y *FragmentUsuariosConectados.* Dicha clase utilizará por defecto *FragmentLlamadasRecientes*, y será el usuario el que cambie de un fragmento a otro mediante la interfaz de usuario.

3.5.2.5 SettingsActivity.java

Esta clase es utilizada para modificar las preferencias de la aplicación, las cuales pueden ser: usar red local o VPN, y realizar llamadas de audio o llamadas de audio y vídeo. Dichas modificaciones son guardadas en las *SharedPreferences.*

3.5.2.6 ActivityConversacion.java

Esta clase es utilizada cuando tenemos una llamada entrante. En este caso, se preguntará al usuario si desea aceptar la llamada. En el caso de aceptarla, la actividad nos dirige a la actividad *MainActivity*. Si por el contrario el usuario rechaza la llamada, la actividad nos dirige a la actividad ListadoUsuariosActivity.

3.5.2.7 MainActivity.java

Esta clase se corresponde con la actividad principal de la aplicación. En ella se produce tanto la recepción como el envío de audio y vídeo del usuario con el que estamos manteniendo la llamada. Si el usuario sale de la aplicación, la actividad sigue funcionando

en segundo plano, por lo que el usuario sigue recibiendo datos relativos al audio y al vídeo. Además, cuando se crea esta actividad, se llama al servicio *AudioGroupService* y a las hebras *ThreadEnviarVídeo* y *ThreadRecibirVideo* , encargados de capturar el audio y el vídeo respectivamente a nuestro *Smartphone*, y enviarlo al usuario con el que nos estamos comunicando, así como recibir los datos correspondientes de dicho usuario. En el caso de que se quiera finalizar la llamada, desde esta actividad se pararía el servicio y las hebras. Para ello el usuario dispone de un botón en la interfaz. En el caso de que el usuario perdiese la conexión a Internet, también se detendría la comunicación.

3.5.2.8 FragmentLlamadasRecientes.java

Esta clase se encargará de mostrar en una lista las llamadas que ha realizado un usuario. En cada llamada aparecerá el usuario con el que se mantuvo la comunicación, la fecha en la que se mantuvo, y la duración de la llamada.

3.5.2.9 FragmentUsuariosConectados.java

Esta clase se utilizará tras haber realizado inicio de sesión. En este caso, se realiza una petición HTTP al servidor, en la que se pide la lista de usuarios conectados, junto con los parámetros necesarios para establecer una comunicación. Una vez recibida la respuesta por parte del servidor, se mostrará en una lista en la interfaz el nombre de cada usuario junto con su disponibilidad. Tras mostrar esta lista, si se desea llamar a aun usuario que esté disponible, se ejecutará la tarea asíncrona *TareaAsincronaEnviarSIP.*.

3.5.2.10 LlamadaDTO.java

Esta clase es utilizada para crear un objeto con los atributos necesarios para describir una llamada, que será utilizada en el *fragment FragmentLlamadasRecientes.*

3.5.2.11 UsuarioActivoDTO.java

Esta clase es utilizada para crear un objeto con los atributos necesarios para describir un usuario que está activo. Dicha clase se utilizará en el *fragment* FragmentUsuariosConectados.

3.5.2.12 API.java

Esta clase es la encargada de realizar la petición HTTP al servidor web. Para ello se utiliza la librería Retrofit disponible para Android.

3.5.2.13 InternetReceiver.java

Esta clave es utilizara para comprobar en todo momento el estado de la conexión a Internet. Esta clase es utiliza por el *fragment FragmentUsuariosConectados,* ya que, si estamos visualizando la lista de usuarios conectados, y se pierde la conexión a Internet, se mostrará un mensaje en la interfaz informando al usuario de la pérdida de conexión a Internet.

3.5.2.14 NetworkStatusChange.java

Esta interfaz contiene define dos métodos que serán utilizados en el momento en que *InternetReceiver* detecte un cambio de conectado a desconectado en la conexión y viceversa. Con estos métodos se mostrará por la interfaz de usuario un mensaje informando a dicho usuario.

3.5.2.15 UsuarioService.java

Esta interfaz es utilizada para definir los recursos que puede solicitar el usuario al servidor, los cuales son: *login,listarUsuarios,cerrarSesion* y *actualizarDisponibilidad.*

3.5.2.16 NotificationHandlerEstadoLLamada.java

Esta clase es la encargada de crear una notificación en el momento en el que se produce el inicio de la llamada. Dicha notificación es útil cuando el usuario sale de la aplicación y desear acceder a la aplicación de forma rápida.

3.5.2.17 NotificationHandlerFinLlamada.java

Esta clase es utilizada para crear una notificación en el momento en el que se produzca el fin de una llamada.

3.5.2.18 NotificacionHandlerSIP.java

Esta clase se utiliza cuando recibimos una llamada entrante, y el usuario mantiene la aplicación en segundo plano. De esta forma, mediante la notificación el usuario sabe que está recibiendo una llamada entrante.

3.5.2.19 AudioGroupService.java

Esta clase es la encargada de montar el servicio mediante el cual captura captura audio y lo envía. Para ello utiliza la librería *android.net.rtp* disponible en el paquete de red de Android.

3.5.2.20 TareaAsincronaEnviarSIP.java

Esta clase es utilizada en el momento en el que un usuario quiere mantener una comunicación con otro usuario. Para ello, enviará a través de un *Socket* el mensaje SIP necesario para invitar a otro usuario a una sesión. Además en el cuerpo de dicho mensaje irá otro mensaje del protocolo SDP, necesario para intercambiar parámetros relacionados con la comunicación.

3.5.2.21 TareaAsincronaRecibirSIP.java

Esta clase es utilizada para recibir mensajes SIP de un usuario, bien porque hemos iniciado un intento de llamada anteriormente, o bien porque un usuario quiere contactar con nosotros.

3.5.2.22 TareaAsincronaEnviarACK.java

Esta clase es utilizada para enviar mensajes ACK entre usuarios que están manteniendo una comunicación. Dichos mensajes son utilizados para que la aplicación sepa en todo momento si el otro extremo de la comunicación ha perdido la conexión. Por lo tanto, la aplicación mandará mensajes ACK cada 2 segundos.

3.5.2.23 TareaAsincronaRecibirACK.java

Esta clase es utilizada para recibir mensajes ACK utilizados para saber si un usuario está activo durante una comunicación. Si la tarea asíncrona deja de recibir mensajes ACK, querrá decir que el otro extremo de la comunicación habrá perdido la conexión, por lo tanto se dará por finalizada la llamada, informando al usuario.

3.5.2.24 MyCameraViewGenerarVideo.java

Esta clase será la encargada de obtener los *frames* de vídeo a través de la cámara. Para ello utilizará las interfaces *SurfaceHolder.Callback y Camera.PreviewCallback,* disponibles en el paquete *View* y en el paquete *Hardware* de Android respectivamente. Esta clase será utilizada en la actividad *MainActivity.*

3.5.2.25 MyServerThread.java

Esta clase será utilizada para enviar las tramas de vídeo al otro extremo a través de una hebra.

3.5.2.26 MyClientThread.java

Esta clase es utilizada para recibir las tramas de vídeo del otro extremo a través de una hebra.

3.5.2.27 MyHandler.java

Esta clase hace de nexo de unión entre *MyClientThread* y *MainActivity,* de forma que cuando se recibe una trama de vídeo a través de *MyClientThread,* la actividad *MainActivity* puede acceder a ella mediante esta clase.

3.5.2.28 PermissionHandler.java

Esta clase es utilizada para pedir los permisos necesarios de grabación de audio y acceso a cámara al usuario cuando se ejecuta la aplicación por primera vez.

3.5.2.29 AdapterListaLlamadasRecientes.java

Esta clase se utiliza para visualizar por la interfaz de usuario los datos correspondientes a las últimas llamas realizadas. Dicha clase se utiliza en el *fragment FragmenLlamadasRecientes.*

3.5.2.30 AdapterListaUsuarios.java

Esta clase es la encargada de mostar por la interfaz de usuario los datos correpondientes a los usuarios conectados. Dicha clase se utilizada en el *fragment FragmentListaUsuarios.*

3.5.2.31 PagerAdapter.java

Esta clase se utiliza para llamar a los fragments FragmentListaUsuarios y FragmentLlamadasRecientes cuando nos encontramos en la actividad ListadoUsuariosActivity.

3.5.2.32 App.java

Esta clase es utilizada para definir el tiempo en el que se debe mostrar por la interfaz de usuario el fondo de pantalla mientras la actividad *SplashScreenActivity* está siendo ejecutada.

3.5.2.33 HandlerTimerEnviarLlamada.java

Esta clase es utilizada para definir el tiempo en el que un usuario espera respuesta tras llamar a otro usuario. Si tras este tiempo no se obtiene respuesta del otro usuario, la llamada será fallida.

3.5.2.34 HandlerTimerRecibirLlamada.java

Esta clase es utilizada para definir el tiempo del que dispone un usuario para contestar una llamada entrante. Si tras este tiempo el usuario no contesta, la llamada se dará por fallida.

3.5.2.35 Utils.java

En esta están implementados métodos que son utilizados por varias clases a su vez, tales como guardar parámetros en las *SharedPreferences*, obtener datos de las *SharedPreferences,* obtener la dirección IP de una interfaz local, etc.

4 RESULTADOS Y DISCUSIÓN

4.1 Discusión sobre codificadores de audio

En este apartado se va a discutir sobre la calidad de los posibles codificadores de audio que se pueden usar en la aplicación, descritos en el apartado de Introducción. Para comparar dichos codificadores, se utilizará la Puntuación de opinión Media (en inglés MOS, *Mean Opinion Score).* Este estándar mide la calidad del audio teniendo en cuenta la compresión, transmisión, y descompresión. La puntuación se asigna mediante los estándares P.800 y P.300 del Sector de normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones (UIT-T). Los valores MOS que puede obtener un codificador son los siguientes:

Clasificación MOS	Calidad Subjetiva
5	Excelente
4	Buena
3	Suficiente
2	Pobre
1	Mala

Tabla 1. Clasificación MOS

De acuerdo con la clasificación descrita anteriormente, pasaremos a comparar la clasificación para los codificadores disponibles en nuestra aplicación, de acuerdo con los valores otorgados por la UIT-T, que son los siguientes:

Codificador	Clasificación MOS
AMR	3.65
GSM Full-Rate	3.7
GSM Enhanced Full-Rate	4.16
G.711	4.2

Tabla 2. Clasificación MOS para distintos codificadores

Como podemos observar, el codificador AMR presenta un valor de 3.65 y el codificador GSM *Full-Rate* presenta un valor de 3.7. Ambos codificadores tienen una calidad subjetiva de suficiente, por lo que podrían ser utilizados en una llamada sobre IP.

El codificador GSM *Enhanced Full-Rate* presenta un valor de 4.16, muy similar al codificador G.711, que presenta un valor de 4.2. Estos dos últimos codificadores tienen una calidad subjetiva de buena, por lo que se aconsejaría utilizar alguno de estos dos frente a los codificadores AMR y GSM FR, siempre y cuando las condiciones de la red sean las óptimas.

El siguiente paso consiste en comparar los codificadores descritos anteriormente de forma experimental. Para ello se ha realizado la encuesta disponible en el Anexo 6.3 a 10 personas. Los resultados obtenidos son los siguientes:

	G.711	GSM	GSM EFR	AMR
Persona nº 1	3	3	5	5
Persona nº 2	3	3	5	5
Persona nº 3	2	4	4	5
Persona nº 4	2	3	5	5
Persona nº 5	4	3	3	5
Persona nº 6	4	3	4	5
Persona nº 7	4	2	4	3
Persona nº 8	5	4	5	2
Persona nº 9	4	4	3	5
Persona nº 10	4	2	3	4

Tabla 3. Resultados experimentales para codificadores según encuesta MOS

En relación a los resultados obtenidos, la media de los resultados para el codificador G.711 es 3.5, la media de los resultados para el codificador GSM es 3.1, la media de los resultados para el codificador GSM EFR es 4.1, y la media de los resultados para el codificador AMR es 4.4. Si comparamos los resultados obtenidos de forma experimental con los resultados proporcionados por la UIT-T, vemos que hay una pequeña diferencia, especialmente en el codificador G.711 y en el codificador AMR. Dicha diferencia puede deberse a la reducida cantidad de personas utilizadas para el experimento, pues dicho experimento realizado con un mayor número de personas nos arrojaría unos resultados más precisos. Además, los resultados obtenidos pueden estar determinados por factores ajenos tales como presencia de eco en sitos cerrados, etc.

4.2 Discusión sobre distintas condiciones de red

En cuanto al testeo de la aplicación bajo distintas condiciones de la red, se ha podido comprobar que la aplicación funciona con bastante fluidez sobre cualquier tipo de red, en gran medida al uso de una VPN, la cual facilita que la comunicación entre usuarios sea posible cuando estos se encuentren en distintas redes. En cuanto a los datos correspondientes al intercambio de audio, al utilizarse el protocolo UDP, no se inyecta un

alto tráfico de datos sobre la red, por lo que el servicio funciona correctamente. En cuanto a los datos correspondientes al intercambio de vídeo, al utilizar una tasa de 15 *frames* por segundo, la calidad del vídeo es SD (en inglés *Standard Definition),* y aunque se utiliza el protocolo de transporte TCP (dicho protocolo utiliza más cabeceras y genera más tráfico que el protocolo UDP debido a los mecanismos que implementa para el control de errores y de flujo), el tráfico de datos no es elevado, por lo que el servicio de intercambio de vídeo funciona correctamente sobre cualquier red.

5 CONCLUSIONES

Este Trabajo Fin de Grado ha servido para desarrollar una aplicación para sistemas operativos Android que permita que usuarios puedan realizar video-llamadas entre ellos. En la actualidad, todo el mundo se encuentra conectado a la red desde cualquier sitio, y por ello, es necesario ofrecer este tipo de servicios a usuarios.

5.1 Objetivos logrados

- Implementación de un cliente Android para el intercambio de *streaming* multimedia entre usuarios en tiempo real.
- Implementación de un servidor en la nube que autentique a usuarios y proporcione información necesaria a dichos usuarios para el correcto establecimiento de una video-llamada.

5.2 Vías de mejora

- Incorporación de un servidor de conferencias web que haga prescindible el uso de una VPN en el caso de que los usuarios se encuentren en distintas redes.
- Incorporación de utilidades tales propias de Android tales como intercambio entre cámara frontal y cámara trasera, silenciar micrófono o bloquear cámara.
- Incorporación de opción de llamada/vídeo llamada entre más de dos usuarios.

6 ANEXOS

6.1 Anexo I: Servicios de la aplicación

A continuación, se van a definir los diferentes servicios que ofrece el servidor web. Estos servicios devolverán respuestas en formato JSON. Las peticiones se podrán hacer con el método GET y POST del protocolo HTTP, y en el cuerpo de la petición podrá ir información en formato JSON.

- Servicio: /
  - Descripción: devuelve un documento HTML (en inglés *HyperText Markup Language)* con un formulario en el que un usuario podrá registrarse.
  - Tipo de servicio: GET
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/

Tabla 4. Url /

- Servicio: registro
  - Descripción: el usuario envía sus datos personales para registrarse.
  - Tipo de servicio:POST
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/registro

Tabla 5. Url registro

  - Ejemplo de petición realizada al servicio *registro:*

{“nombre”:”Daniel”,”email”:”daniel@gmail.com”, ”usuario”:”Daniel”,”password”:123456,”apellidos”:”Martínez León”,”teléfono”:600000000}

Tabla 6. JSON petición registro

  - Campos del JSON petición:

Campos	Tipo	Descripción
nombre	string	Nombre del usuario creado
email	string	Email del usuario
usuario	string	Nombre de usuario
password	string	Contraseña del usuario
apellidos	string	Apellidos del usuario
teléfono	int	Teléfono del usuario

Tabla 7. Campos del JSON petición. Servicio registro

- Respuesta: si la petición es satisfactoria, se envía el siguiente mensaje:

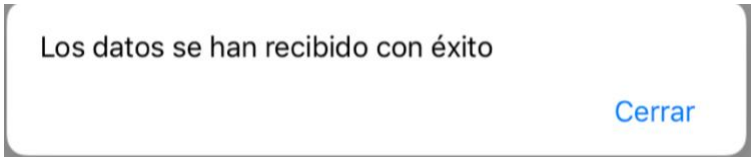
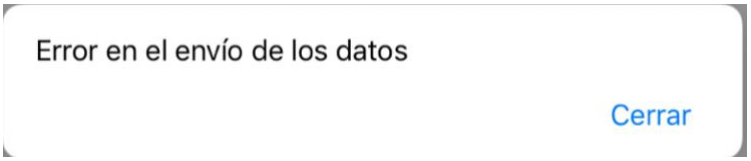


Figura 17. Respuesta satisfactoria del servicio registro

Si la petición es incorrecta, se envía el siguiente mensaje:



- Servicio: login
  - Descripción: el usuario intenta inicio de sesión.
  - Tipo de servicio: POST
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/login

Tabla 8. Url login

- Ejemplo de petición realizada al servicio *login:*

{“direccionIPRedLocal”:”192.168.1.141”,”direccionIPVPN”:”10.8.0.42”,
”usuario”:”Daniel,”puertoACK”:42335,”disponibilidad”:true,
”puertoSIP”:45461,”clave”:”123456”}

Tabla 9. JSON petición login

- Campos del JSON petición:

Campos	Tipo	Descripción
direccionIPRedLocal	string	Dirección IP local del usuario
direccionIPVPN	string	Dirección IP de la interfaz VPN del usuario
usuario	string	Nombre de usuario
puertoACK	int	Puerto de escucha de mensajes ACK
puertoSIP	int	Puerto de escucha de mensajes SIP
disponibilidad	boolean	Disponibilidad del usuario
clave	string	Contraseña del usuario

Tabla 10. Campos del JSON petición. Servicio login

- Respuesta: Si el inicio de sesión es satisfactorio, en el cuerpo de la respuesta irá el valor *true.* Si el inicio de sesión es incorrecto, en el cuerpo de la respuesta irá el valor *false.* Además, en la respuesta el servidor dotará al usuario de dos *cookies* en las que se guardará el valor del nombre de usuario y el valor de la contraseña. De esta manera, en posteriores peticiones HTTP, el cliente enviará en las cabeceras de la petición las dos *cookies,* para que de esta manera el servidor pueda identificar al cliente en todo momento.

- Servicio: listarUsuarios
  - Descripción: una vez iniciada sesión, el usuario pide una lista de usuarios conectados actualmente.
  - Tipo de servicio: POST
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/app/listarUsuarios

Tabla 11. Url listarUsuarios

  - Petición realizada al servicio: en esta petición no se envía información en el cuerpo. Cabe destacar que en las cabeceras de la petición se incluyen dos cabeceras de tipo *cookie,* una para indicar el nombre de usuario, y otra para indicar la contraseña del usuario. De esta manera el servidor podrá comprobar si este usuario ha iniciado sesión anteriormente.

- Ejemplo de respuesta obtenida del servicio:

[{“usuario”:”alba”,”direccionIPRedLocal”:”192.168.1.65”,”direccionIPVPN”:”10.8.0.14”,”puertoEscuchaSIP”:7687,”puertoEscuchaACK”:20145,”disponibilidad”:false},
{“usuario”:”maria”,”direccionIPRedLocal”:”192.168.1.102”,”direccionIPVPN”:”10.8.0.16”,”puertoEscuchaSIP”:7126,”puertoEscuchaACK”:14123,”disponibilidad”:true}]

Tabla 12. JSON respuesta listarUsuarios

- Campos del JSON respuesta:

Campos	Tipo	Descripción
dirección	string	Dirección IP local del usuario
disponibilidad	boolean	Disponibilidad del usuario
usuario	string	Nombre de usuario
puertoACK	int	Puerto de escucha de mensajes ACK
puertoSIP	int	Puerto de escucha de mensajes SIP
clave	string	Contraseña del usuario

Tabla 13. Campos del JSON respuesta. Servicio listarUsuarios

- Servicio: actualizarDisponibilidad
  - Descripción: el cliente utiliza este servicio en el caso de que ha establecido una comunicación con otro cliente, o en el caso de que ha dejado de comunicarse con otro cliente.
  - Tipo de servicio:POST
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/actualizarDisponibilidad

Tabla 14. Url actualizarDisponibilidad

  - Petición realizada al servicio: en esta petición se envía la clave “disponibilidad” con el valor que se quiere modificar en formato JSON.

{“disponibilidad”:true}

Tabla 15. JSON petición actualizarDisponibilidad

  - Campos del JSON petición:

Campos	Tipo	Descripción
disponibilidad	boolean	Disponibilidad del usuario

Tabla 16. Campos del JSON respuesta. Servicio actualizarDisponibilidad

- Respuesta: el servidor responderá con el mensaje 200 OK del protocolo HTTP, y en el cuerpo incluirá el valor *true* en formato JSON.

- Servicio: cerrarSesion
  - Descripción: el usuario cierra sesión.
  - Tipo de servicio: POST
  - Url para invocar el servicio:

https://tfgpozo-dot-smartpv.appspot.com/app/cerrarSesion

Tabla 17. Url cerrarSesion

- Petición realizada al servicio: en esta petición no se envía información en el cuerpo. El servidor reconoce al usuario mediante las dos cabeceras de tipo *cookie,* en las que se incluyen las credenciales del usuario.
- Respuesta: el servidor responderá con el mensaje 200 OK del protocolo HTTP, y en el cuerpo incluirá el valor *true* en formato JSON.

6.2 Anexo II: Manual de usuario de la aplicación

Una vez instalada la aplicación, para poder utilizarla el usuario deberá abrir la aplicación pulsando sobre el icono de la misma.

Figura 18. Icono StreamingUJA

6.2.1 AcercaDeActivity

En esta actividad se muestra por la interfaz del usuario los datos relativos a la aplicación móvil.

Figura 19. AcercaDeActivity

6.2.2 SplashScreenActivity

En esta actividad se comprueba si existen credenciales en las *SharedPreferences.* En el caso de que existan, se produce intento de inicio de sesión con dichas credenciales. Si el inicio de sesión no es satisfactorio o no existen credenciales, ésta actividad nos dirige a la actividad *Login.* Si el inicio de sesión es satisfactorio, ésta actividad nos dirige a la actividad *ListadoUsuariosActivity*.

Figura 20. SplashScreenActivity.

6.2.3 LoginActivity

En esta actividad se presenta un formulario para que el usuario introduzca su nombre de usuario y contraseña. Una vez introducidos ambos datos, se produce la autenticación contra el servidor. Si la autenticación es correcta, las credenciales (nombre de usuario y contraseña) serán guardadas en las *SharedPreferences* y la actividad nos dirigirá a la actividad *ListadoUsuariosActivity.* Si la autenticación no es correcta, la actividad nos volverá a pedir las credenciales. Además esta actividad dispone de un enlace a una página web en la que un nuevo usuario podrá registrarse. En el caso de que no dispongamos de conexión a internet o el servidor no esté disponible, la aplicación nos mostrará un mensaje informándonos.

Figura 21. LoginActivity. Introduzca credenciales

Figura 22. LoginActivity. Sin conexión a Internet

Figura 23. LoginActivity. Credenciales incorrectas

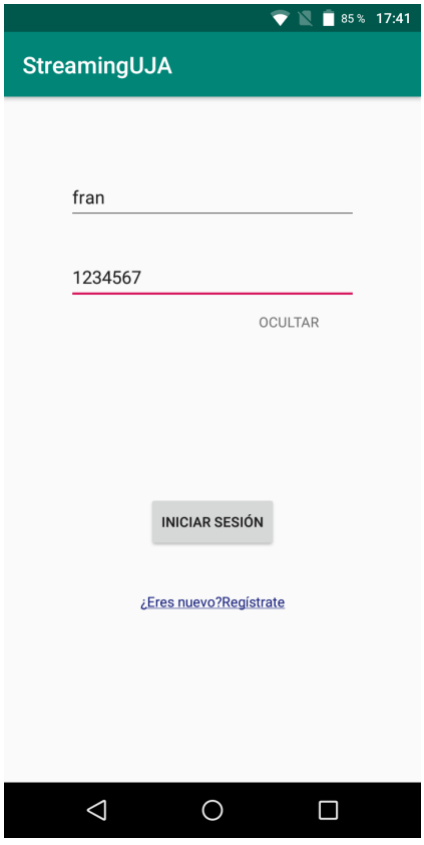


Figura 24. LoginActivity. Contraseña visible

Figura 25. LoginActivity. Servidor inactivo

6.2.4 ListadoUsuariosActivity

Esta actividad contiene dos fragmentos que serán explicados a continuación. Además contiene un menú desplegable para que el usuario puede cerrar sesión, pueda acceder a las preferencias y pueda comprobar los datos de la aplicación.

6.2.5 FragmentUsuariosConectados

Esta pestaña pertenece a la actividad *ListadoUsuariosActivity,* y en ella podemos encontrar los usuarios que están conectados, junto con la disponibilidad de cada usuario en ese momento. En esta pestaña encontramos el botón "*Actualizar",* mediante el cual se obtiene la lista de usuarios conectados actualizada. Al pulsar sobre un usuario, la aplicación dará paso al intento de llamada a dicho usuario, siempre y cuando éste esté disponible. Si no está disponible, la aplicación nos avisaría con un mensaje. Si el intento de llamada no es satisfactorio, la aplicación nos avisaría con un mensaje, devolviéndonos a la pestaña *ListadoUsuarios.*

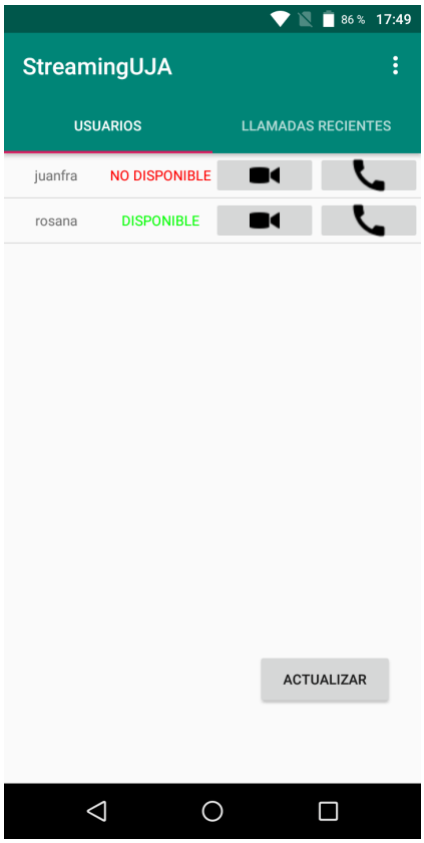


Figura 26. ListadoUsuariosActivity. Lista de usuarios

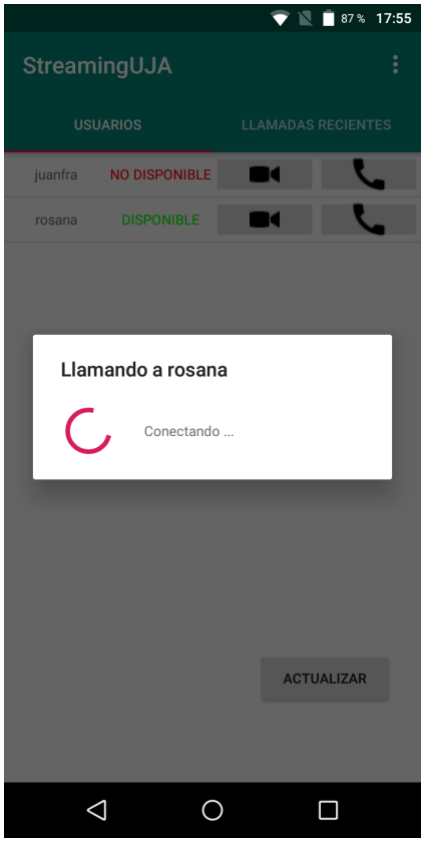


Figura 27. ListadoUsuariosActivity. Llamando...

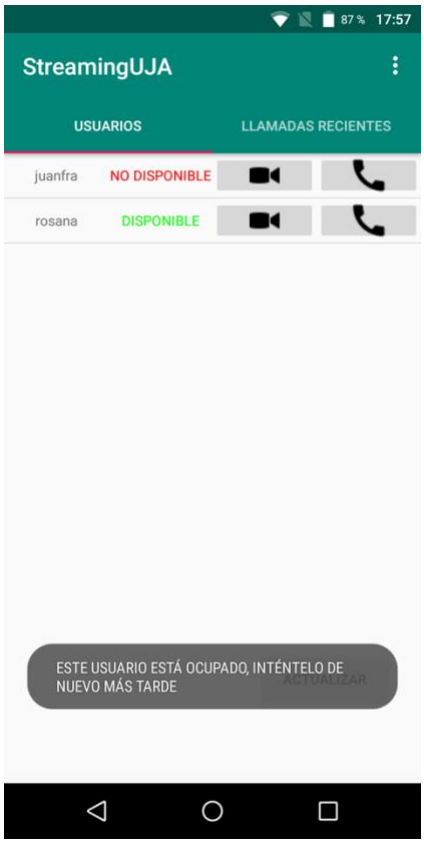


Figura 28. ListadoUsuariosActivity. Usuario ocupado

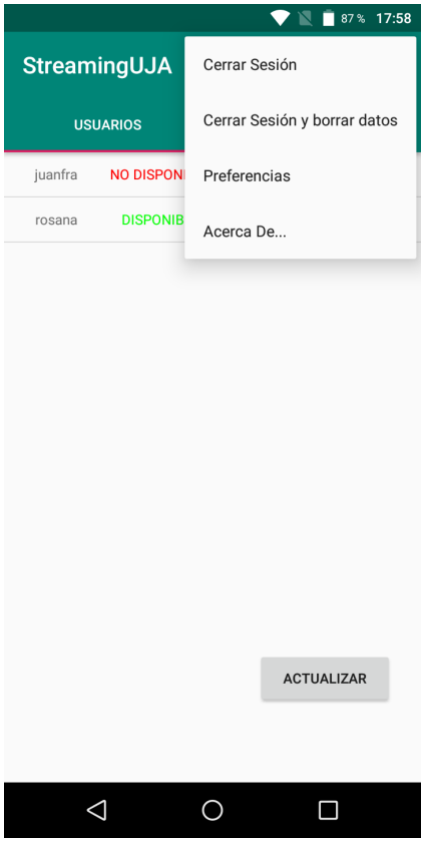


Figura 29.ListadoUsuariosActivity. Menú desplegable

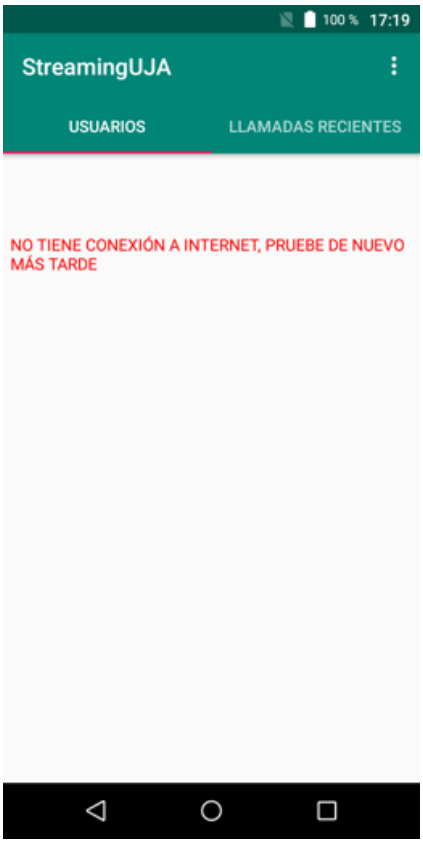


Figura 30. ListadoUsuariosActivity. Sin conexión a Internet.

6.2.6 FragmentLlamadasRecientes

Esta pestaña pertenece a la actividad ListadoUsuariosActivity, y en ella podemos encontrar las llamadas recientes, en las cuales aparece el usuario con el que hemos mantenido dicha llamada, la fecha en la que se produjo, y la duración.

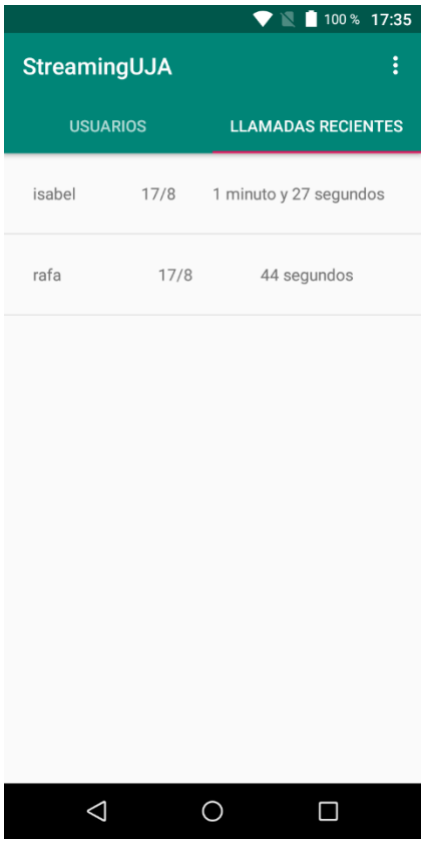


Figura 31. LlamadasRecientes. Llamadas recientes

6.2.7 ActivityConversacion

Esta actividad se produce cuando recibimos una llamada entrante. En la actividad se le pregunta al usuario si quiere aceptar dicha llamada. Si la llamada es aceptada, se producirá el intercambio de información multimedia, y se dirigirá a la actividad *MainActivity*. Si la llamada se rechaza, se dirigiría a la actividad *ListadoUsuariosActivity*.

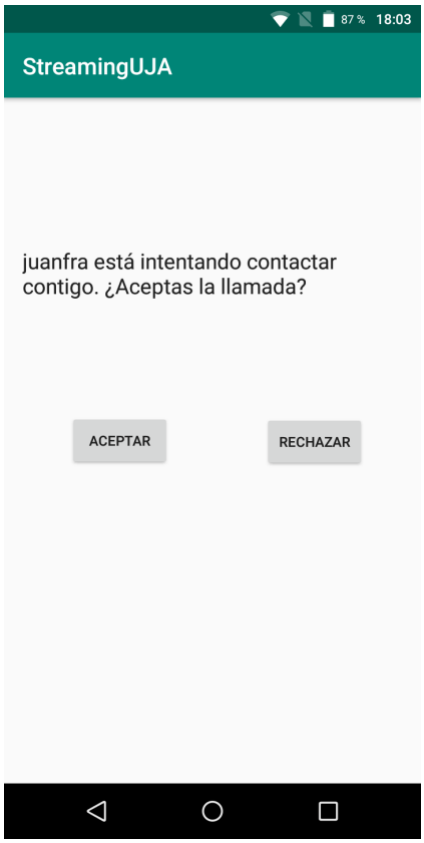


Figura 32. ActivityConversación. Llamada entrante

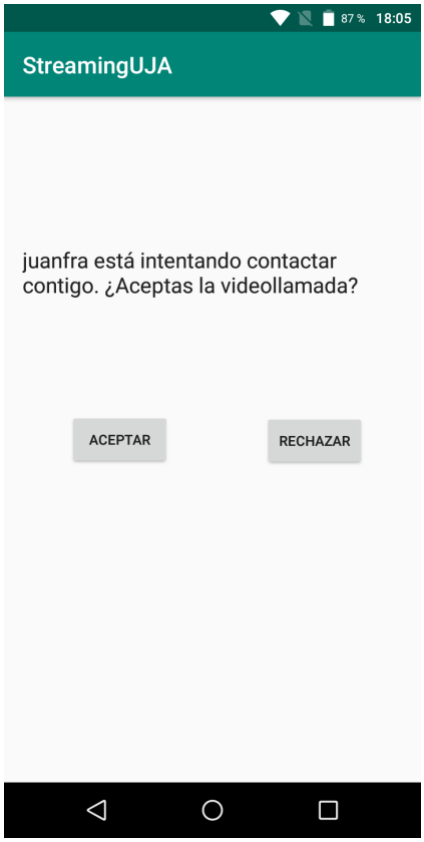


Figura 33. ActivityConversación. Videollamada entrante

6.2.8 PreferenciasActivity

En esta actividad el usuario podrá establecer la siguiente preferencia:

- Comunicación con usuarios a través de red local o a través de VPN.

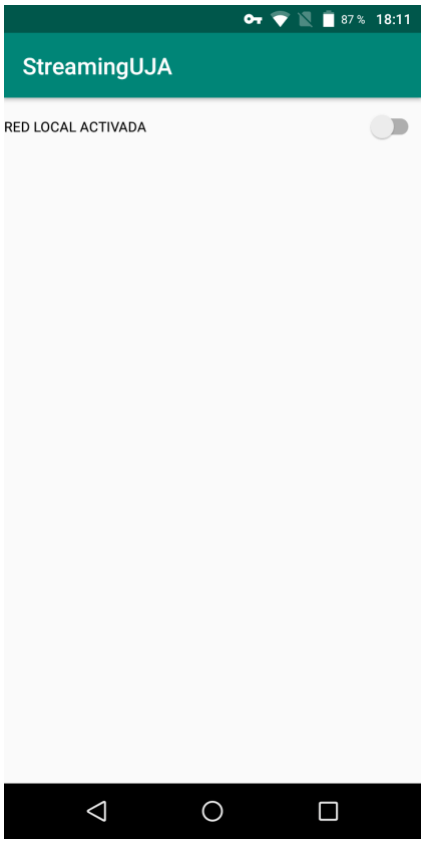


Figura 34. PreferenciasActivity. Red Local activada

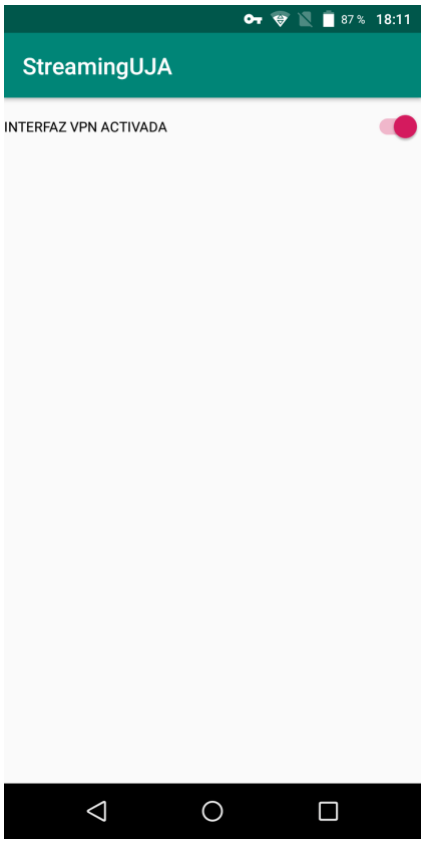


Figura 35. PreferenciasActivity. Interfaz VPN activada

6.2.9 MainActivity

En esta actividad se está produciendo la llamada entre usuarios, y en ella se reciben datos multimedia relativos al audio y video enviados por el otro extremo. Los datos recibidos relativos al audio serán reproducidos a través del altavoz, y los datos recibidos relativos al vídeo serán mostrados por la interfaz del usuario. Además, en la esquina superior derecha aparecerá un pequeño rectángulo en el que se podrá ver los *frames* que están siendo capturados por la cámara.

Figura 36. MainActivity. Videollamada. Botón STOP oculto.

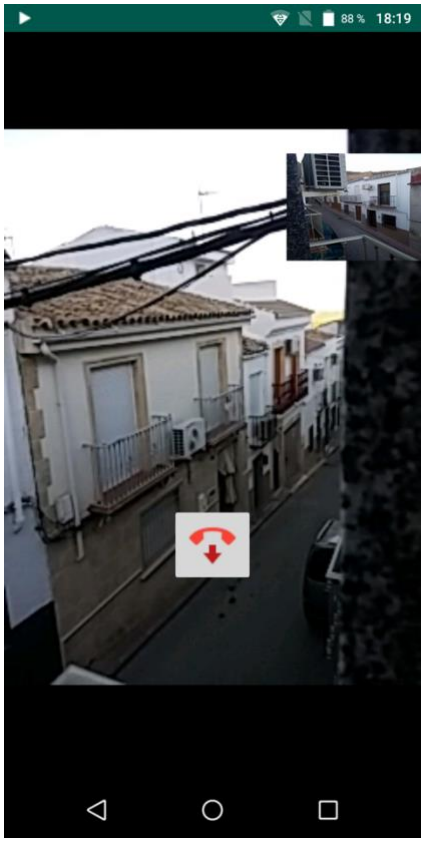


Figura 37. MainActivity. Videollamada. Botón STOP visible

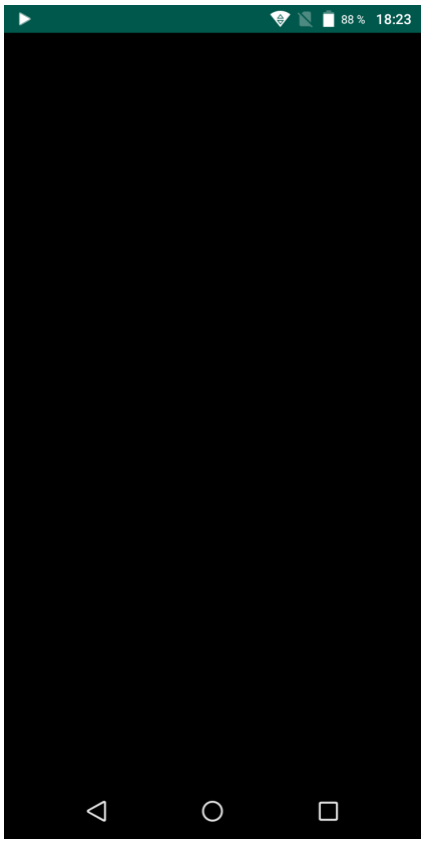


Figura 38. MainActivity. Llamada. Botón STOP oculto

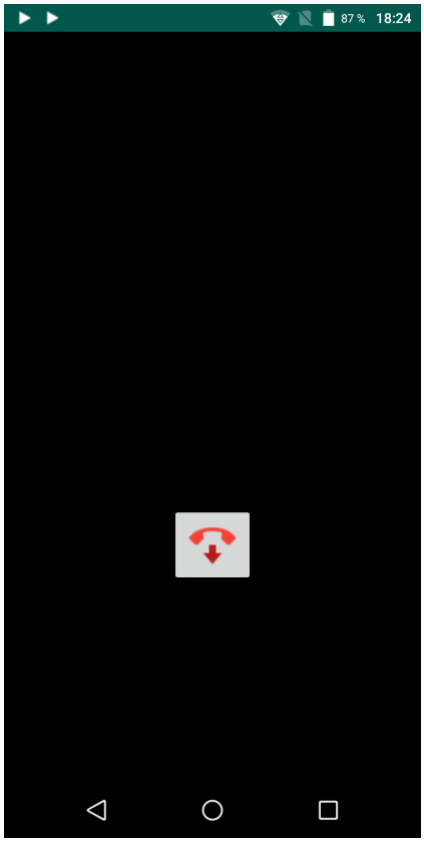


Figura 39. MainActivity. Llamada. Botón STOP visible

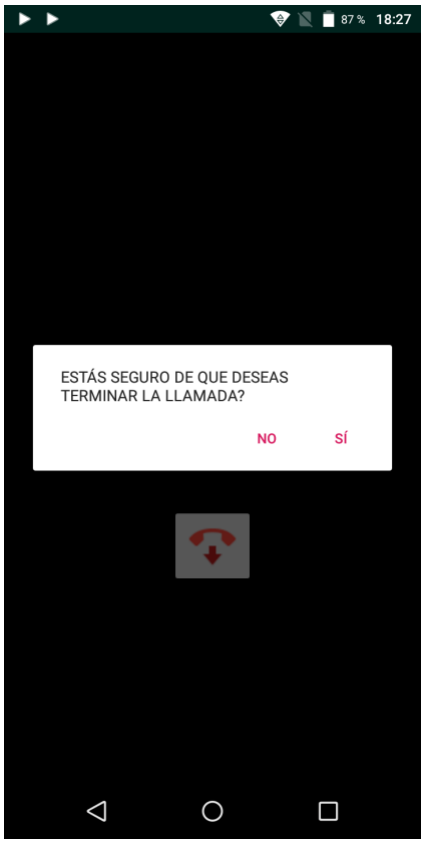


Figura 40. MainActivity. Llamada. ¿Estás seguro de...?

6.3 Anexo III:Cuestionario MOS

En este apartado se incluye el cuestionario que ha sido utilizado para valorar los codificadores de audio de acuerdo a la escala MOS de forma experimental. Para responder al cuestionario se ha producido una llamada telefónica con cada codificador de audio entre dos usuarios de duración 3 minutos. Cada usuario ha contestado a las siguientes preguntas:

1. ¿Cómo valora la comunicación que se ha producido usando el codificador G.711?
2. ¿Cómo valora la comunicación que se ha producido usando el codificador GSM-FR?
3. ¿Cómo valora la comunicación que se ha producido usando el codificador GSM-EFR
4. ¿Cómo valora la comunicación que se ha producido usando el codificador AMR?

Los valores posibles, de mayor a menor, son los siguientes:

5. Conversación perfecta con calidad excelente

4. Calidad buena aunque se no es perfecta
3. Ligeramente molesta
2. Muy molesta, difícil mantener la conversación
1. Imposible mantener la conversación

6.4 Anexo IV: Soporte Multi lenguaje de la aplicación

En este apartado se va a detallar los pasos realizados para que la aplicación móvil funcione en los idiomas Español e Inglés. La aplicación móvil funcionará con un idioma u otro teniendo en cuenta el idioma actual de los ajustes del *smartphone*. Para el soporte multi lenguaje, se ha configurado en la aplicación móvil el archivo *strings.xml*, disponible en la carpeta *res.* Al haber configurado la aplicación móvil para dos idiomas, se ha desarrollado dos archivos *strings.xml* correspondientes a cada idioma, habiendo un tercer archivo *strings.xml,* el cual se utilizará por defecto en el caso de que en el *smartphone* esté configurado otro idioma distinto a español e inglés.

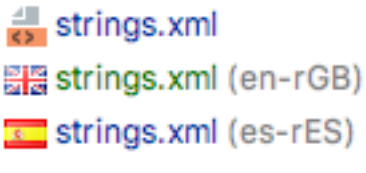


Figura 41. Archivos strings.xml de la aplicación móvil

7 REFERENCIAS BIBLIOGRÁFICAS

[1] Android Inc. Descripción general de notificaciones. [consulta: 10 abril 2019]. Disponible en: https://developer.android.com/guide/topics/ui/notifiers/notifications?hl=es-419

[2] Android Inc. SharedPreferences. [consulta: 15 marzo 2019]. Disponible en: https://developer.android.com/reference/android/content/SharedPreferences

[3] Android Inc. Intents y filtros de intents [consulta: 20 marzo 2019]. Disponible en: https://developer.android.com/guide/components/intents-filters.html?hl=es

[4] Universidad Politécnica de Valencia. Multimedia en Android. 11 enero 2013 [consulta: 3 febrero 2019]. Disponible en: https://www.youtube.com/watch?v=L98ra5MInEI

[5] James Revelo. Tutorial para crear un servicio en Android. 2 Julio 2015 [consulta: 5 marzo 2019]. Disponible en: http://www.hermosaprogramacion.com/2015/07/tutorial-para-crear-un-servicio-en-android/

[6]. Gonzalo Eduardo Pérez Correa. PreferenceActivity en Android. 29 septiembre 2014 [consulta: 10 junio 2019]. Disponible en: https://www.youtube.com/watch?v=Uxxs4scQ3Vk

[7]. Android Inc. Actividades. [consulta: 1 febrero 2019]. Disponible en: https://developer.android.com/guide/components/activities?hl=es-419

[8]. Android Inc. Android.net.rtp. [consulta: 16 febrero 2019]. Disponible en: https://developer.android.com/reference/android/net/rtp/package-summary

[9]. Academia Android. Thread, Handler y AsyncTask, ¿cuál elegir?. 20 septiembre 2014 [consulta: 5 mayo 2019]. Disponible en: https://academiaandroid.com/thread-handler-asynctask-cual-elegir/

[10]. Ricardo Moya. Gson (JSON) en Java, con ejemplos. 2 mayo 2017 [consulta 20 mayo 2019]. Disponible en: https://jarroba.com/gson-json-java-ejemplos/

[11]. Stan Malcolm. Add cookies to retrofit 2 request. 17 julio 2016 [consulta 15 mayo 2019]. Disponible en: https://stackoverflow.com/questions/38418809/add-cookies-to-retrofit-2-request

[12]. Coding in flow. Detect Wifi State Changes with BroadcastReceiver- Android Studio Tutorial. 6 marzo 2018 [consulta 1 julio 2019]. Disponible en: https://www.youtube.com/watch?v=esWDhcxH7tA

[13]. Square. Retrofit. A type-safe http client for Android and Java. 2013 [consulta 10 mayo]. Disponible en: https://square.github.io/retrofit/

[14]. José Manuel Pérez Lorenzo. Documentación proporcionada como apoyo a la asignatura Ingeniería de Servicios de Telecomunicación de la EPSL-UJA, Tema 1, “Aplicaciones MVC”. Curso 2017/2018.

[15]. José Manuel Pérez Lorenzo. Documentación proporcionada como apoyo a la asignatura Ingeniería de Servicios de Telecomunicación de la EPSL-UJA, Tema 2, “Framework Spring MVC”. Curso 2017/2018.

[16]. José Manuel Pérez Lorenzo. Documentación proporcionada como apoyo a la asignatura Ingeniería de Servicios de Telecomunicación de la EPSL-UJA, Tema 3, “Arquitectura orientadas a Servicios: REST y SOAP”. Curso 2017/2018.

[17]. European Telecommunications Standars Institute (ETSI). Digital cellular telecommunications system (Phase 2+); Full rate speech; Transcoding (GSM 06.10 version 8.1.1. Release 1999). 30 noviembre 2000 [consulta 18 febrero 2019].

[18]. European Teleccomunications Standars Institute (ETSI). Digital cellular telecommunications system (Phase 2+); Enhanced Full Rate (EFR) speech transcoding; (GSM 06.60 version 8.1.1 Release 1999). 15 noviembre 2000 [consulta 18 febrero 2019].

[19]. Teleccomunication standardization sector of International Teleccomunication Union (ITU-T). General aspects of digital transmission systems. Terminal Equipments. Pulse code modulation (PCM) of voice frequencies. ITU-T Recommendation G.711. 28 junio 1990 [consulta 18 febrero 2019].

[20]. Teleccomunication standardization sector of International Teleccomunication Union (ITU-T). Series P: Terminals and subjective and objective assessment methods. Methods for objective and subjective assessment of speech quality.