UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

TRANSCRIPCIÓN AUTOMÁTICA DE REUNIONES: DESARROLLO DE UNA INTERFAZ HARDWARE Y SOFTWARE BASADA EN UN SMARTPHONE

Ismael Zaabi Sáez

Tutor: José Colás Pasamontes

JULIO 2015

TRANSCRIPCIÓN AUTOMÁTICA DE REUNIONES: DESARROLLO DE UNA INTERFAZ HARDWARE Y SOFTWARE BASADA EN UN SMARTPHONE

AUTOR: Ismael Zaabi Sáez

TUTOR: José Colás Pasamontes

HCTLab

Dpto. de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2015

Resumen

Debido a la actual necesidad de multitud de empresas y asociaciones de generar y guardar documentos que detallen lo acontecido en una reunión, diversas compañías de software están intentando conseguir un sistema eficaz de transcripción de reuniones con el cual a partir de las voces de cada asistente a la reunión se genere un acta de la misma. En este trabajo se presenta una nueva solución al problema actual de la transcripción de reuniones en el cual, a diferencia de otras tecnologías ya presentes, uno de los asistentes a la reunión registra al resto de participantes y crea una sala de reuniones virtual mediante una aplicación web. Cada uno de los usuarios registrados se conectan a través de sus Smartphones con S.O. Android a esta sala de reuniones virtual donde se recogen todas las voces enviadas por cada usuario mediante tecnología VoIP para que, posteriormente, la plataforma de procesamiento ubicada en la nube transcriba cada una de las señales de habla a texto de forma independiente, generándose así un acta de la reunión que le será enviada al creador de la sala virtual. Además de analizar y explicar el diseño y la arquitectura de este complejo sistema, también se diseñará y analizará la interfaz de usuario, por lo que también se estudiará exhaustivamente el uso de diferentes dispositivos que mejoren la calidad de las señales de voz y reduzcan el ruido (micrófonos, soportes, conectores de audio, etc.), así como diferentes aspectos que puedan mejorar la usabilidad y accesibilidad del producto.

Abstract

Due to the actual necessity from companies and associations to generate and preserve meeting documents, various software companies are trying to develop an efficient meeting transistor system in order to create a record from assistants voices. The aim of this work is to present a new solution to this actual problem, in which, differentiating from nowadays technologies, one of the meeting assistants records the rest of the assistants and creates a virtual room through a web application. Each of the registered users are connected via their Android smartphones. In this meeting room all of the user voices sent by VoIP technology are transcribed by the processing platform to text generating thus a meeting record that is send to the meeting room creator. In addition to analyze and explain both design and architecture of the complex system, user interface would be also design and analyze, so that difference dispositive such as microphones, sports, audio connectors… would be also studied in the field of improving audio signal quality and noise. Finally improving usability and accessibility would be an important task and different aspects would be studied in order to improve both of them.

Palabras clave

VoIP, SIP , API, transcripción, segmentación de voces, sistema, VAD, ASR, sala de reunión, audioconferencia, nodo de servicio, smartphone, servidor web, plataforma, protocolo HTTP, cancelador de ruido, clustering, auriculares, polling, módulo.

Keywords

VoIP, SIP, API, transcription, voice segmentation, system, VAD, ASR, meeting room, audioconference, service node, smartphone, web server, platform, HTTP protocol, noise canceler, clustering, headsets, polling, module.

Agradecimientos

El primer agradecimiento va dirigido a mi tutor, José Colás, el cual ha me ha prestado una gran ayuda durante todo el período de tiempo que me ha llevado realizar y escribir este trabajo que, otorgándome todo tipo de recursos sin los cuales hubiera sido muy difícil terminar este proyecto. También quisiera agradecer el apoyo a todo el equipo de Vocalia, que me han tratado como a uno más, y en especial agradecimiento a Alberto Calvo que me ha ayudado con las labores de implementación y pruebas de este trabajo.

A todos y cada uno de mis compañeros durante estos largos cuatro años, tanto a aquellos que nos dejaron como a los que continúan día a día junto a mí. Gracias a Adri, Dani, Iñaki, Javi, Alex, Jose, Gallego y Juan, pues sin su ayuda probablemente esto no habría acabado tan pronto. A los demás, también muchísimas gracias, pues habéis demostrado ser todos unos fantásticos compañeros.

Por supuesto, a toda mi familia, que siempre han estado ahí apoyándome, tanto en los buenos como en los malos momentos, y que se han sacrificado siempre en mi beneficio. GRACIAS.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

En la actualidad muchas instituciones públicas y corporaciones privadas necesitan poseer un registro de las conversaciones producidas en las reuniones que tienen habitualmente con el objetivo de ratificar lo sucedido en las mismas. Algunas de estas sociedades utilizan métodos de transcripción manuales para obtener lo que comúnmente se llama 'acta de la reunión', como por ejemplo sucede en determinados tribunales, donde el/la taquígrafo/a anota todo lo acontecido durante la sesión.

Sin embargo, el hecho de que no todas las empresas pueden permitirse un taquígrafo para sus reuniones y conferencias, y que la responsabilidad de redactar estas actas tan valiosas recaiga sobre una persona procede a que se innove sobre nuevos procedimientos de transcripción automáticos más económicos y temporalmente más breves que el método tradicional. Por ello, y aprovechando que actualmente los smartphones forman parte de nuestra vida cotidiana, en este trabajo se afronta el diseño y desarrollo de un nuevo sistema de transcripción automático de reuniones basado en smartphone Android, es decir, este sistema será accesible por todo aquel que posea un dispositivo Android (siempre y cuando este dispositivo soporte llamadas VoIP y las APIs SIP).

Al utilizar dispositivos móviles para captar las voces y atendiendo a que de forma habitual los usuarios disponen de ellos, el transcriptor de reuniones diseñado está pensado para que los usuarios a su vez puedan llevar a cabo audioconferencias, es decir, los participantes de la reunión no tienen por qué estar físicamente en la misma sala, algunos o incluso todos ellos pueden realizar la reunión desde entornos diferentes, aunque para este tipo de conferencias se recomienda estar en entornos poco ruidosos (en casas, bibliotecas y espacios cerrados en general).

No obstante, hoy en día la mayoría de los módulos de transcripción automática que existen en el mercado funcionan bajo ciertas condiciones, esto suele ser, bajo situaciones de bajo ruido y sobre todo con escaso solapamiento de voces, por lo que la implementación de un transcriptor en una reunión posee ciertas trabas ya que es en este escenario donde puede aparecer con más facilidad cualquier tipo de ruido o alboroto que impida la correcta transcripción.

Por tanto, este TFG no se limitará únicamente al diseño del sistema de transcripción de reuniones, sino que también se buscarán diferentes soluciones que solventen estos problemas de ruido y solapamiento de voces. Para ello se describirá en esta memoria el módulo de transcripción empleado en este sistema y se mostrarán cual han sido las diferentes mejoras añadidas al mismo que corrigen estos inconvenientes, aunque también se hará un análisis en el apartado de pruebas de los diferentes dispositivos que evitan la aparición de ruidos y solapamientos.

1.2 Objetivos

En consonancia con lo comentado en el apartado anterior, el objetivo principal de este proyecto es conseguir el acta de una reunión a partir de todas y cada una de las transcripciones generadas por la plataforma de procesamiento situada en la nube. Idealmente, la técnica de transcripción de las señales de voz implementada permite una

conversión de voz a texto perfecta, aunque en este trabajo se intentará mejorar para que funcione en condiciones de ruido.

Por consiguiente, el primer objetivo será realizar un análisis íntegro sobre las dos técnicas más populares de segmentación empleadas en la transcripción de reuniones: "la segmentación de hablantes multicanal" [1] y "la detección de cambios acústicos" [2], las cuales sirven de fundamento a la técnica de transcripción empleada en este sistema que también se definirá.

Sin embargo, el objetivo primordial de este trabajo será construir toda la arquitectura que permite el acceso de los usuarios a la plataforma de transcripción, esto es, diseñar una aplicación web que permite crear salas virtuales de reuniones y una aplicación en Android que conecte a los usuarios de una misma reunión (lo que se conoce como front-end del sistema), y desarrollar todas las aplicaciones web que gestionen las bases de datos y el sistema de información (back-end).

Como último objetivo, tras diseñar la arquitectura e implementarla junto a la plataforma de reconocimiento y transcripción ya existente en la nube, se estudiará el uso de determinados equipos hardware que mejoren la calidad de las grabaciones, así como determinadas actitudes que los usuarios de este sistema deberán seguir para obtener una correcta transcripción de la reunión.

1.3 Organización de la memoria

La presente memoria está estructurada de la siguiente manera:

- En el capítulo 1 se introduce una motivación previa donde se refleja el problema actual de las transcripciones de reuniones y donde se define a grandes rasgos el comportamiento del sistema diseñado, así como también se esclarece los objetivos de dicho trabajo.
- Posteriormente, en el capítulo 2, se detallará el estado del arte en cuanto a transcripción de reuniones se refiere y se explicarán las principales y actuales técnicas de segmentación. Finalmente, se comentará los beneficios y soluciones que aporta el sistema realizado y como se adapta al marco actual de las transcripciones de reuniones.
- El capítulo 3 hace referencia al diseño y arquitectura del sistema completo. Se detallará información sobre todos y cada uno de los elementos a desarrollar e implementar.
- En el apartado correspondiente al capítulo 4, se explicará el desarrollo seguido tanto para diseñar el front-end como para establecer el back-end. Por supuesto, también se detallará información sobre las herramientas utilizadas en este trabajo.
- Una vez desarrollado el sistema, en el capítulo 5 se analizarán los resultados obtenidos de la fase de pruebas, así como también se estudiará y comentará el uso y acoplamiento de otros dispositivos a este sistema con el fin de mejorar la calidad del mismo.
- Por último, el capítulo 6 tratará sobre las conclusiones y trabajo futuro de este proyecto.

2 Estado del arte

2.1 Concepto, tipos y ejemplos comerciales de transcriptores

Desde el punto de vista software, un transcriptor simplemente traduce el habla humana en texto, bien de forma automática o de forma manual.

En el caso de un transcriptor manual (o semi-automático) la transcripción es realizada (o supervisada) por un humano, lo cual ocasiona cierta dependencia, aunque a pesar de ello existen en el mercado muchas herramientas de transcripción manuales que facilitan esta tarea (conocidos como asistentes de transcripción o dictado digital). Los asistentes de transcripción más conocidos son, entre muchos otros, Transana [4] que también permite al usuario editar y transcribir el audio correspondiente a un video, y el software creado por el Instituto Internacional de Ciencias y Computación de Berkeley (ICSI) y denominado Transcriber [5]. Este último quizás sea el más conocido y utilizado para transcribir puesto que permite transcripción multicanal, es decir, permite al usuario transcribir múltiples audios simultáneamente, obteniendo así un fichero en formato XML. Cabe destacar que en el sistema de transcripción diseñado en este trabajo se empleará este último programa tras finalizar la transcripción automática, ya que este sistema es susceptible de tener errores de transcripción, con lo que tras finalizar este proceso, un humano deberá, a partir de este programa externo al sistema, corregir y retocar el acta de la reunión que ha sido transcrita automáticamente.

Por otro lado, actualmente existen transcriptores de uso comercial completamente automáticos, como puede ser la API "Speech-to-Text" de VoxSigma [6] y cuyo comportamiento es similar al realizado en este trabajo (permite obtener actas en XML con un porcentaje de error aceptable, está disponible hasta en 7 idiomas y es capaz de realizar transcripciones automáticas de teleconferencias). Otro transcriptor automático disponible en el mercado y ampliamente utilizado por su capacidad de acoplamiento con multitud de programas (como por ejemplo Microsoft Word y Pages) y por su facilidad de implementación en otros sistemas privados a través de su API es el Dragon Speech Recognition Software de Nuance [7]. No obstante, este último producto no ofrece la posibilidad de realizar transcripciones automáticas multicanal y simultáneas.

2.2 Reconocedor de voz automático (ASR)

En este apartado se explica la tecnología empleada en los actuales transcriptores automáticos conocida como tecnología ASR (Automatic Speech Recognition), la cual se encarga de reconocer la voz humana (con mejor o peor precisión) de forma automática obteniendo generalmente un reconocimiento con marcas de tiempo correspondientes al instante en el que cada palabra es pronunciada [3]. Sin embargo, existen transcriptores automáticos que en vez de estimar el instante exacto en el que cada palabra es articulada, inicialmente segmentan el audio en fragmentos más pequeños donde no se produce ningún silencio para posteriormente calcular el intervalo de tiempo (inicio y final) que comprende a cada uno de estos segmentos. Habitualmente, la forma de establecer las marcas de tiempo anteriormente comentadas se justifica según el método de reconocimiento automático empleado, que pueden ser el reconocimiento de palabras parciales y localización de palabras clave (correspondiente al primer caso donde se marcan los instantes de tiempo de

cada palabra pronunciada), y el método LVCSR (Large vocabulary Continuous Speech Recognition) correspondiente al segundo.

2.2.1 Método de localización de palabras clave y reconocimiento de palabras parciales.

Este método es ciertamente en esencia la unión de dos similares métodos de reconocimiento de voz mencionados en el título de este subapartado. La localización de palabras clave se encarga, como bien dice su nombre, de localizar una serie de palabras preestablecidas e indexadas en una pequeña lista que posteriormente transcribe. Como cabe esperar, este método es computacionalmente rápido, pero evidentemente no efectúa una transcripción completa, aunque su asociación con el segundo método de reconocimiento de palabras parciales permite una transcripción definitiva. Básicamente este método en vez de reconocer cada palabra una a una, lo que ejecuta es un reconocimiento de palabras parciales como pueden ser las sílabas y los fonemas, es decir, examina fonema a fonema y sílaba a sílaba para posteriormente transcribir. Aunque no lo parezca, la asociación de ambos métodos sigue siendo computacionalmente veloz [3].

2.2.2 Método LVCSR

Este caso es muy similar al método de reconocimiento de palabras parciales, siendo además el método más ampliamente utilizado de los ASR. En este proceso, a diferencia de realizar un proceso de reconocimiento silábico y fonético, el reconocimiento de voz se realiza mediante modelos estadísticos basados en las unidades fonéticas (fonemas) y en el modelo del lenguaje correspondiente al audio a transcribir [3]. La incorporación del modelo del lenguaje propicia que las transcripciones de este método sean más precisas y contengan un número menor de fallos, aunque el cálculo del error utilizando este método depende del vocabulario empleado en el audio, es decir, si la voz incorpora un vocabulario diferente al introducido en el modelo de lenguaje (palabras foráneas propias de otros idiomas) la tasa de fallos aumenta drásticamente. Por lo tanto, se hace inevitable añadir al modelo utilizado vocabulario (palabras completas) de otros idiomas, lo que a su vez supone un esfuerzo computacional mayor a la hora del reconocimiento, aunque por fortuna con la potencia de procesamiento de los equipos disponibles hoy en día esto no supone un desgaste computacional muy grande, permitiendo añadir vocabularios de 50.000 hasta 300.000 palabras, reduciendo al mínimo el número de palabras no contempladas por el modelo de lenguaje utilizado.

Tras definir los dos métodos más comunes de reconocimiento de voz automático, se llega a la conclusión de que el método de reconocimiento LVCSR es superior siempre y cuando la voz a reconocer no contenga demasiadas palabras OVV (del inglés out-of-vocabulary, fuera del vocabulario), lo cual hace de este método el preferido para transcribir audio originado en salas de reuniones [3]. En el caso en el que se produzca un sobreuso de palabras OVV, es difícil de pronosticar cuál de los dos métodos se usará.

2.3 Segmentación

Según la tésis doctoral sobre segmentación, diarización y transcripción del habla de Marijn Huijbregts [3], la segmentación es un módulo perteneciente a los sistemas de reconocimiento que utilizan el método LVCSR. Este nuevo módulo, como se explicó en anteriores apartados, se encarga de trocear el audio de entrada en fragmentos más pequeños para facilitar el reconocimiento y posterior transcripción de los mismos, aunque bien es cierto que existen variaciones en cuanto a la segmentación se refiere.

2.3.1 Clases de segmentación

De forma habitual, la técnica empleada para segmentar un audio concreto consiste en localizar las zonas de voz y diferenciarlas de las zonas de silencio y de ruido, consiguiendo así un filtrado de las zonas de audio que no interese reconocer y, por tanto, optimizando el reconocedor y mejorando la tasa de errores del mismo. El uso de esta técnica permite además incorporar todo tipo de información adicional a cada fragmento (metadatos) como pueden ser las anteriores marcas de tiempo mencionadas que, para el caso que nos ocupa, nos será de gran utilidad. No obstante, esta forma de segmentar el audio en función del silencio (denominada segmentación basada en el silencio) no es la única. Existen otras dos técnicas de segmentación: segmentación basada en modelos y segmentación basada en métrica.

Los sistemas LVCSR que utilizan la técnica de segmentación basada en modelos efectúan inicialmente el entrenamiento de un modelo mixto gaussiano (GMM) por cada tipo de segmento, es decir, dependiendo del tipo de audio fragmentado (si es audio correspondiente a voz, a silencio, a música, o a otro tipo de audio preestablecido) realizará el entrenamiento de uno u otro GMM. Estos GMM se utilizan como funciones de densidad de probabilidad (PDF) en un modelo oculto de Markov (HMM), el cual se utiliza para realizar una decodificación de Viterbi que obtiene la segmentación del audio [3]. El inconveniente de este método es que los GMM tienen que ser previamente entrenados, lo que propicia a una alta tasa de errores a posteriori si el audio a reconocer tiene características muy diferentes a los datos previamente entrenados. Por esta razón, no es una de las técnicas de segmentación más empleadas, sobre todo si se trata en el campo de la transcripción de reuniones, ya que es en este escenario donde pueden aparecer multitud de tipos de ruidos que dificultan las tareas de reconocimiento.

Por otra parte, la segmentación basada en métrica emplea una ventana desplazable o variable que determina el tipo de audio en intervalos. Para ello la ventana se sitúa en un intervalo concreto del audio y resuelve si el punto medio de este intervalo debe ser o no ser marcado como un extremo de un segmento, es decir, comprueba si el intervalo seleccionado posee las mismas características que los intervalos adyacentes, y si no es así, marca ese instante de tiempo como inicio (o fin) del segmento. Para contrastar si dos intervalos pertenecen a un mismo segmento se utilizan una distancia métrica que verifica lo parecido que son ambos intervalos. Habitualmente, la distancia métrica se calcula a partir del Criterio de Información Bayesiano (BIC), el cual es muy frecuente en sistemas de reconocimiento que detectan el cambio de hablante, esto es, en sistemas con multitud de audios mezclados como pueden ser los sistemas de transcripción de reuniones propuestos por J.M. Pardo, X. Anguera y C. Wooters [1], y por S.S. Chen y P.S. Gopalakrishnan [2] que más tarde se definirán.

2.3.2 VAD (Voice Activity Detection)

Cabe resaltar en un único apartado esta técnica puesto que es una de las más empleadas en el caso del reconocimiento de voz enviada por VoIP, y por tanto, será un método más empleado en este sistema. Esta técnica normalmente viene acompañada por un proceso previo de segmentación del cual, como bien se sabe, se obtienen fragmentos del audio a reconocer, aunque en determinadas ocasiones es el propio VAD el que se comporta como un segmentador basado en silencios, diferenciando los fragmentos de audio de los de silencio, tal y como sucede en el sistema implementado.

Conforme al estudio realizado por A. Sangwan, Chiranth M.C., H.S. Jamadagni, R. Sah, R.V. Prasad y V. Gaurav [8] sobre las técnicas VAD para la transmisión por internet del habla humana en tiempo real, los algoritmos VAD (detección de actividad de la voz) empleados en sistemas VoIP tienen como objetivo reducir el ancho de banda, pero siempre manteniendo la calidad de audio, consiguiendo así una transmisión de datos eficiente y optimizada. Para ello, el VAD determina en las tramas RTP (segmentos correspondientes a la voz enviada) cuales poseen señales de voz, un proceso que lleva a cabo midiendo la energía de las señales de voz encapsuladas en cada trama RTP. Si la energía medida del audio correspondiente a esa trama supera un cierto umbral (calculado a partir de la formulación incluida en este paper [8]), la trama se clasifica como "activa", y si no alcanza ese límite se le clasifica como "inactiva". Este umbral es recalculado cada vez que el VAD detecta una trama inactiva mediante un PNU (actualización periódica de ruido). Sin embargo, existen otros algoritmos VAD que en vez de determinar mediante el cálculo de la energía que tramas son correspondientes a silencio y cuales a señal de voz, se basan en otras características de la señal como el número de cruces por cero (ZCD). Este algoritmo sabe que una señal de voz de 10 ms de duración habitualmente tiene de 5 a 15 cruces por cero mientras que la señales correspondientes al ruido tienen un número aleatorio y despreciable de cruces, lo cual le permite formular un nuevo umbral que posibilita la identificación del tipo de trama. Así mismo, es frecuente emplear este algoritmo ZCD tras haber ejecutado el algoritmo AED (detector adaptativo de la energía) correspondiente al cálculo de la energía de un audio anteriormente definido debido a que es común encontrarse con que el AED haya definido como inválida una trama de voz con una energía muy baja. Por tanto, la sucesión de estos dos algoritmos proporciona que aquellas tramas donde la voz del hablante posea una energía baja puedan ser catalogadas como válidas. No obstante, en ocasiones el algoritmo ZCD malinterpreta las tramas con determinados tipos de ruidos como activas debido a que en algunos momentos estas tramas superan el nuevo umbral establecido. Este defecto se puede corregir con el último algoritmo VAD denominado detector de fricativas débiles (WFD), que se ocupa de identificar mediante las consonantes fricativas ('s', 'x' y 'f' en el castellano) las tramas que son de audio, independientemente de la SNR del mismo. Este algoritmo obtiene por sí solo buenos resultados en el reconocimiento de la voz, aunque si se quiere obtener un gran rendimiento al reconocer, se suele asociar con el algoritmo AED.

Como se mencionó antes, la técnica VAD también se puede emplear como segmentador, puesto que es capaz de diferenciar en un audio que partes son correspondientes a voz humana y cuáles no, permitiendo así trocear audios en fragmentos más pequeños, pero siempre y cuando este audio se identifique con un único hablante, es decir, podrá segmentar cuando la voz a reconocer provenga de un mismo usuario. Por tanto, como en este trabajo se empleará un canal por cada usuario (evitando así aparentemente el acoplamiento de diferentes voces), el algoritmo VAD será elegido como segmentador para este sistema.

2.3.3 Otras técnicas de segmentación

A parte de las 3 técnicas de segmentación explicadas en anteriores apartados, existen otro tipo de métodos que permiten fragmentar el audio. Por lo general, estas técnicas son derivadas de las 3 principales y se usan en conjunción con otra clase de métodos como puede ser la diarización (o clusterización) de hablantes. En este trabajo hablaremos de dos técnicas empleadas en la segmentación de audio: "la detección de cambios acústicos" y "la segmentación de hablantes multicanal". La primera técnica está pensada para incorporarse como segmentador de cualquier sistema ASR, mientras que la segunda tiene un uso más limitado pues se emplea en sistemas multicanal correspondientes a reuniones con múltiples micrófonos y distantes entre sí (MDM).

2.3.3.1 Detección de cambios acústicos

Esta técnica, investigada y desarrollada por S.S. Cheng y P.S. Gopalakrishnan [2], permite identificar en un audio los cambios de hablante, de entorno y de canal, es decir, detecta donde se produce un punto de transición (o como denominan en su trabajo, un cambio acústico) y guarda constancia de ello marcando el instante de tiempo en el que se produce. El algoritmo de segmentación diseñado sigue un modelo basado en métrica que utiliza el criterio BIC para calcular las distancias métricas con las que posteriormente define con gran certeza los distintos cambios acústicos producidos y segmenta el audio en función de los mismos.

Además, en este algoritmo el criterio BIC también es aplicado para realizar un clustering de los diferentes tipos de fragmentos de audio que han sido previamente segmentados. Inicialmente, se preestablecen los diferentes tipos de cluster a los que un segmento de audio puede ser asociado (en el caso de las reuniones, existe como mínimo un cluster por cada usuario, además de haber clusters por cada modelo de ruido u otros tipos de audio como la música). Después, cada cluster es modelado según un modelo mixto gaussiano (GMM) correspondiente a cada tipo de audio. Así, utilizando el criterio BIC sobre los clusters y los fragmentos, el algoritmo es capaz de agrupar los segmentos de audio con una buena precisión (entorno al 19% de errores de agrupamiento).

2.3.3.2 Segmentación de hablantes multicanal

En el trabajo realizado por J.M. Pardo, X. Anguera y C. Wooters [1], se expone un caso práctico de la segmentación de audio. En él, proponen un sistema basado en una reunión con múltiples hablantes que utilizan diversos micrófonos (sistema MDM). Sin embargo, el sistema no conoce nada acerca del número de participantes en la reunión ni de su localización dentro de la sala. Tampoco conoce el entorno acústico que les rodea, así como tampoco sabe a quién corresponde cada micrófono (si es que solo le corresponde a un único hablante) ni las características técnicas de estos dispositivos. Por tanto, aunque los participantes empleen canales separados, sus audios (con voz, ruido, música, etc.) se verán obligados a mezclarse en un único audio al que se le ejecutará un algoritmo de segmentado y de clustering para posteriormente ser reconocido mediante un ASR.

La técnica de segmentación empleada se corresponde con una variante de la segmentación basada en modelos, es decir, emplea modelos estadísticos para determinar los fragmentos del audio a reconocer. No obstante, en este escenario se incluye un procedimiento de clustering para agrupar los segmentos con sus semejantes, o lo que es lo mismo, para identificar los diferentes hablantes y ruidos aparecidos. Hasta este proceso de clustering no se conoce el número exacto de participantes, por lo que inicialmente esta técnica requerirá una suposición del número de participantes para modelar los clusters. Para esto último, se utiliza un HMM cuyos estados son modelos gaussianos mixtos.

Cabe resaltar que en la segmentación, para determinar la longitud de un fragmento de audio, se emplea el criterio BIC. Lo mismo sucede en el clustering, donde mediante este criterio se establece a que cluster pertenece un segmento determinado.

En la figura inferior, se ilustra un esquema del sistema investigado por los autores en el que existe un participante por cada micrófono, aunque como bien se explicó antes, el sistema no conoce a priori el número de usuarios. El audio obtenido en cada dispositivo es trasladado a un mezclador que unifica en un único audio todas las voces, donde posteriormente es grabado. Este audio se introduce por un módulo que calcula los coeficientes cepstrales en la frecuencia de Mel (MFCC) que no hace más que extraer las características de las componentes de esta señal de audio. Después, se buscan los puntos de transición (o cambios acústicos) siguiendo un modelo similar al detector de cambios acústicos anteriormente explicado (que por supuesto, hace uso del BIC) y se obtienen los diferentes fragmentos de audio aún sin catalogar. El siguiente paso es la clusterización (o diarización) de dichos segmentos, los cuales son agrupados y posteriormente identificados en función del hablante al que correspondan. Una vez obtenido los pequeños fragmentos de las voces de cada usuario, estos se introducen por un módulo ASR que transcribe cada fragmento a texto, obteniendo un acta de la reunión.

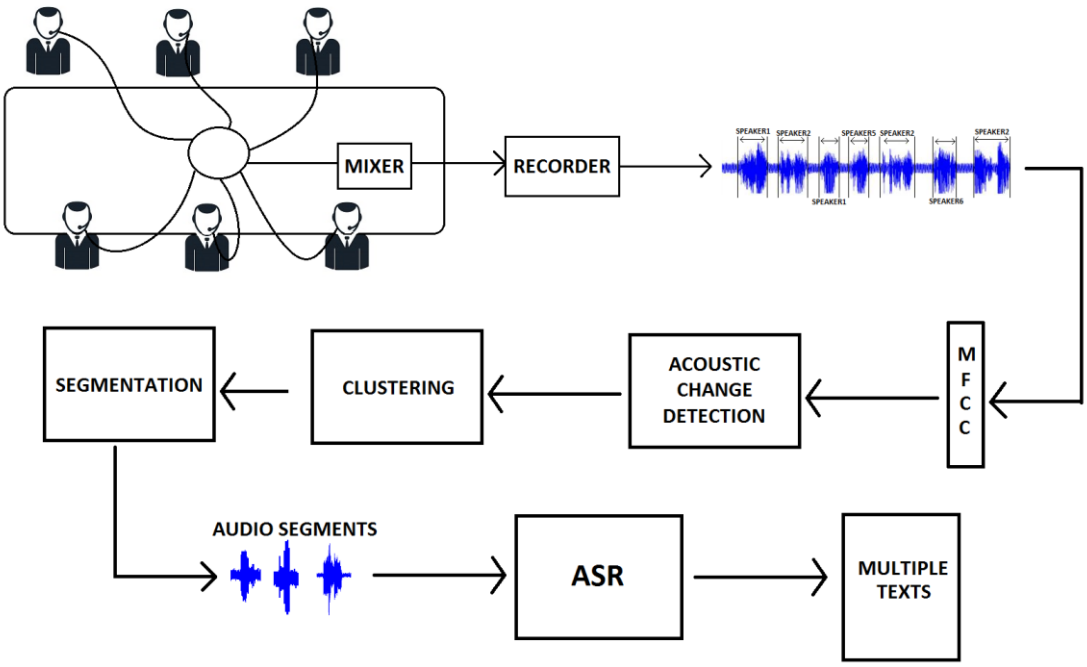

Figura 2-1: Esquema del sistema MDM

Las pruebas realizadas con este sistema se realizaron en 10 reuniones diferentes de las cuales se extrajeron 10 minutos de audio de cada reunión. Estos audios, con todas las voces humanas y ruidos mezclados, se pasaron por el sistema del esquema y se obtuvieron unas tasas de error comprensiblemente aceptables (entorno al 30 % para diferentes tipos de errores).

2.4 Sistema implementado

Una vez definido todos los conceptos relativos a una transcripción, en este apartado se analizará el sistema implementado en función de todos estos conceptos anteriores y dentro del marco de los transcriptores actuales.

Este sistema es ciertamente similar al anterior explicado, aunque posee ciertos cambios. En primer lugar, cada usuario posee un único dispositivo que se encargará de conectarse de forma independiente (esto es, sin que a priori se mezclen las voces de diferentes usuarios en un mismo audio, enviando así un flujo de audio con la voz de un solo locutor) a una sala virtual previamente creada a través de una aplicación web donde el número de usuarios ya ha sido establecido, por lo cual, el posterior proceso de clustering e identificación de locutores no será necesario puesto que sus audios ya están separados. Esto significa pues que el servidor de audio-conferencia que recibe todos los audios (tantos como participantes conectados) deberá grabarlos, lo cual puede suponer algún tipo de malgasto de recurso, aunque el mero hecho de no incorporar un módulo de identificación supone un ahorro computacional mayor que en el sistema del apartado anterior. Así mismo, al tener a los locutores separados, la técnica de segmentación es más simple debido a que ahora en vez de incorporar un módulo que detecte todos y cada uno de los cambios de acústica producidos, el módulo utilizado no es más que un segmentador VAD que segmenta el audio de un locutor con las zonas de voz, lo cual también permite, como cuando se definió el VAD, que se guarden metadatos entorno a estos segmentos como puede ser la ID del usuario, su dirección SIP y las marcas de tiempo de cada segmento, posteriormente utilizadas para establecer el acta de la reunión. Tras obtener los segmentos de voz de todos los usuarios, los pequeños fragmentos son enviados al módulo ASR que transcribe el audio a texto, al igual que en el resto de sistemas. Finalmente, a partir de las marcas de tiempo de cada fragmento se lleva a cabo un ordenamiento de las transcripciones, consiguiendo así un archivo de texto con el acta de la reunión.

La imagen mostrada aquí debajo ejemplifica el proceso de segmentación y reconocimiento, tal y como se ha explicado en el párrafo previo.

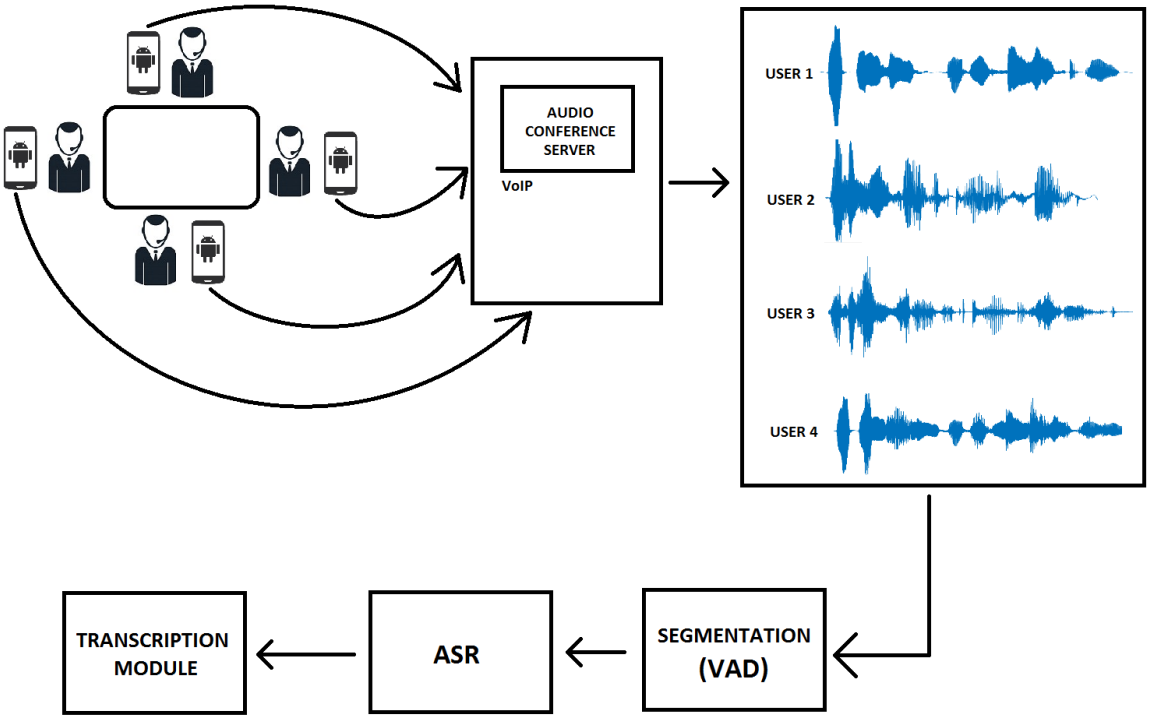

Figura 2-2: Esquema de transcripción en el sistema propuesto

Como se explicará más adelante, es posible que este sistema no sea robusto frente a los ruidos puesto que no presenta ningún cancelador de ruido, por lo cual es probable que próximamente se tenga que integrar algún módulo que suprima el ruido. Esta decisión de integrar un nuevo módulo se determinará cuando el sistema esté construido, es decir, en el apartado de pruebas se valorará su incorporación.

3 Diseño

3.1 Arquetipo del sistema

Tras haber predefinido anteriormente el funcionamiento teórico de los módulos de reconocimiento de voz y transcripción, en este apartado se definirá en líneas generales el modelo del sistema implementado, incluyendo todos sus componentes. El sistema presenta tres componentes principales: los dispositivos Android por donde cada usuario, a través de una aplicación móvil, enviará su respectiva voz para ser grabada y procesada; la plataforma en la nube de procesamiento que contiene, entre otras cosas, los módulos de reconocimiento y transcripción de los audios recibidos; y un servidor web intermedio y separado de la anterior plataforma que gestiona las salas virtuales de reunión y que dirige las comunicaciones entre los otros dos elementos del sistema.

Cabe recalcar que la plataforma en la nube utilizada en este sistema para realizar llamadas VoIP y para llevar a cabo los procesos de transcripción pertenece a la empresa Vocalia Technologies bajo el dominio de dict2me, aunque como se verá, el sistema diseñado también está pensado para funcionar junto a otras plataformas que implementen los mismos (o similares) módulos de reconocimiento y tecnología VoIP.

3.2 Arquitectura del sistema

En este apartado se describirán todos los detalles específicos entorno a la arquitectura diseñada para este sistema, de la cual, como se comentó previamente, existen tres partes diferenciadas: el cliente Android, el servidor web y la plataforma en la nube de Vocalia.

3.2.1 Cliente Android

El cliente Android es quizás el elemento más importante del sistema pues es este dispositivo el único que estará constantemente en contacto con el cliente durante la reunión. Básicamente, en un principio se encarga de conectar con el servidor web para llevar a cabo el proceso de identificación del usuario y de la sala de reunión que le haya sido previamente asignada. Tras obtener una identificación satisfactoria, el dispositivo trata de iniciar una sesión SIP directamente con el servidor SIP situado dentro de la plataforma de Vocalia pasa así poder registrarse en la misma y poder hacer uso de los módulos que esta plataforma incorpora (en este caso, solo se hará uso de los módulos de audio-conferencia y de reconocimiento y transcripción de la voz). La imagen inferior muestra un esquema de esta primera comunicación.

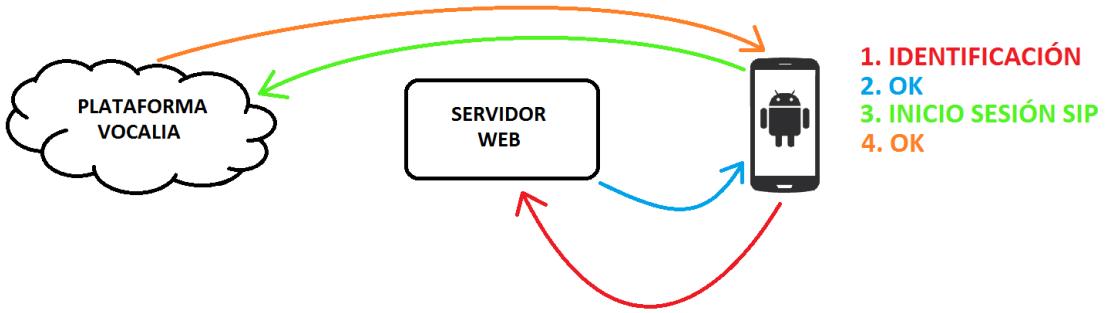

Figura 3-1: Identificación de usuario y registro SIP

Una vez que el dispositivo ha realizado el registro SIP, el cliente ya puede iniciar una llamada de VoIP que se ocupa de transmitir el flujo de audio generado por el mismo y de recibir el audio de los demás usuarios que se encuentren conectados, es decir, la llamada siempre es bidireccional y simultánea con el resto de usuarios, como si se tratase de una audio-conferencia. Pero antes de llamar, la plataforma de Vocalia necesita reservar los recursos necesarios para iniciar la transcripción, por ello previamente el cliente se comunica con el servidor web, el cual a su vez se encarga de comunicarse con la plataforma de Vocalia, la cual retorna una dirección SIP correspondiente con el módulo de audio-conferencia contenido dentro de un nodo de servicio (que ejecuta una máquina virtual) al que el usuario debe llamar, tal y como se muestra en la siguiente figura. En resumen, la plataforma de Vocalia, a partir de su servidor web y de la petición HTTP creada desde el cliente Android, asocia a los usuarios de una misma sala un mismo nodo de servicio que contiene un módulo de VoIP (en este caso, un servidor de audio-conferencia), el cual tiene una dirección SIP única a la que un usuario SIP previamente registrado en la plataforma puede llamar, lo que supone, a efectos prácticos, que esta dirección SIP esté asociada a un único nodo de servicio que contiene diversos módulos (y entre ellos este módulo VoIP).

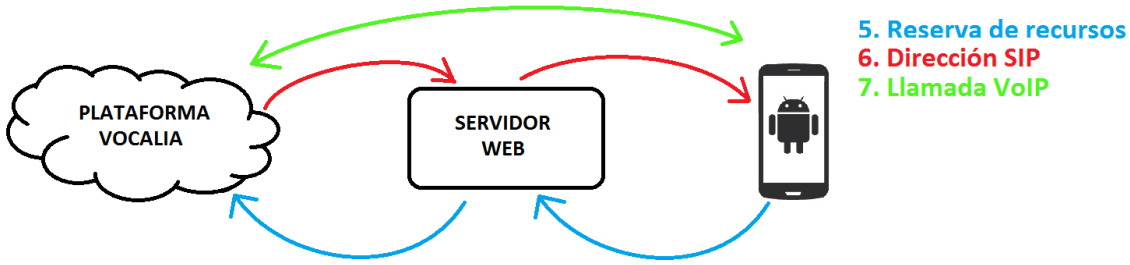

Figura 3-2: Reserva de recursos y establecimiento de llamada

Después de que el cliente reciba la dirección SIP e inicie la llamada, mientras ésta continúe activa, el cliente preguntará al servidor web sobre el estado de los demás usuarios mediante la técnica de polling HTTP, es decir, durante el transcurso de la llamada el cliente mandará cada cierto período de tiempo una petición HTTP preguntando si los demás usuarios se encuentran activos. Esto tendrá utilidad para el posterior diseño y desarrollo de la aplicación Android y su interfaz de usuario, pero

también servirá para saber cuándo concluye una reunión, pues ésta finalizará cuando el usuario nombrado como administrador cuelgue su llamada.

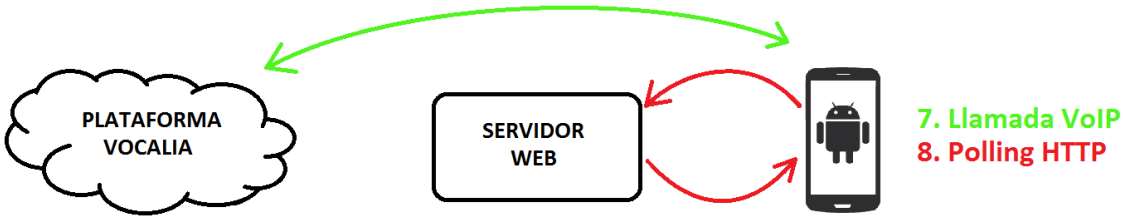

Figura 3-3: Polling HTTP

Si el usuario decide abandonar la sala y desconectar, el dispositivo primeramente notificará al servidor web su abandono, el cual a su vez lo remitirá a la plataforma. Inmediatamente después, el cliente se desconecta del servidor SIP, finalizando así la llamada que tenía en curso, tras lo cual se inicia el proceso de transcripción por parte de la plataforma.

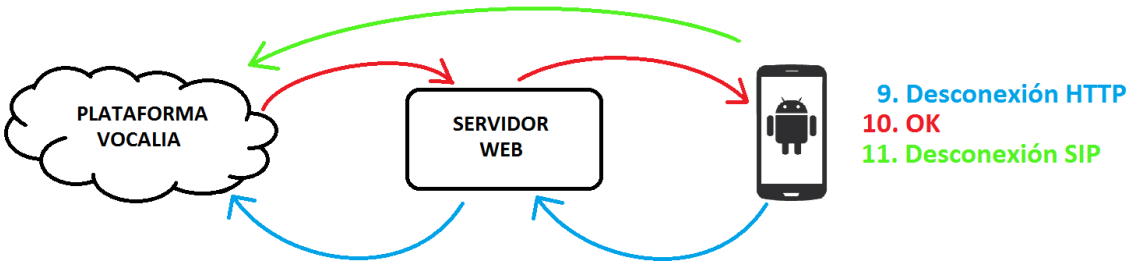

Figura 3-4: Desconexión de un cliente del sistema

Por último, si todos los usuarios abandonan la sala, el cliente envía una petición HTTP de borrado de sala, es decir, se remite al servidor web que los datos de esa sala virtual van a ser eliminados, y simultáneamente se informa a la plataforma que la reunión ha finalizado, iniciando así el proceso de transcripción de la reunión.

Como hecho significativo, se recalca que el funcionamiento habitual del cliente es el detallado previamente, y sólo se comportará así bajo una buena conexión de red, en otras palabras, si el dispositivo no se encuentra conectado a una red Wi-Fi con buena cobertura, es bastante probable que el sistema falle. Sin embargo, esto no implica en principio muchos inconvenientes pues este sistema está pensado para funcionar en salas de reunión o en las viviendas de los usuarios donde la cobertura Wi-Fi suele ser excelente.

3.2.2 Servidor web

Como se ha comentado antes, la funcionalidad del servidor web es la de gestionar las salas virtuales y los usuarios de las reuniones, así como también actuar como nexo de la

mayoría de las comunicaciones entre los clientes y la plataforma. La motivación de separar este componente de la plataforma es simple, con esta división se consigue que el módulo de reconocimiento de voz sea independiente de las reuniones y de sus usuarios, permitiendo así que la parte de cliente más la del servidor web pueda ser incorporada a otras plataformas similares a la de Vocalia y, por tanto, otorgando más escalabilidad al sistema. Además, el hecho de que estén separadas permite incorporar al servidor web su propia seguridad, sin inferir de ninguna forma en la seguridad de la plataforma.

Para poder gestionar las salas virtuales, este elemento contiene una base de datos que almacenará toda la información relativa a los usuarios (identificadores, nombres y correos) y la referente a las salas virtuales de reunión (identificadores de sala, nombres, fechas de inicio de las reuniones y duración estimada). A partir de esta base de datos, se crea una aplicación web que permite crear una sala de reunión virtual y asociarla a diferentes usuarios que pueden o no haber participado en otra reunión. Tras haber creado la sala satisfactoriamente, los usuarios reciben dos códigos a través de los correos insertados en el formulario: un código identificador único de usuario y otro de sala. Obviamente, los usuarios que hayan sido registrados en la misma sala recibirán el mismo código identificador de sala virtual. Esto se empleará para el paso inicial de identificación a través del cliente Android antes mencionado.

Posteriormente, el servidor web diseñado proporciona la gestión de la sala virtual creada y la comunicación entre la aplicación Android y la plataforma mediante varios scripts PHP y HTML, los cuales tienen varias tareas como verificar los identificadores enviados desde los dispositivos, notificar la reserva de recursos a la plataforma, entregar al cliente la dirección SIP con la que llamar a su nodo de servicio, cotejar el estado (conectado o no conectado) de los demás usuarios, ejecutar el proceso de desconexión de los usuarios y eliminar de la base de datos las salas de reunión finalizadas. En el apartado de desarrollo se profundizará sobre la implementación de estas funcionalidades mencionadas.

Para terminar, cabe destacar que una vez que se obtiene el acta de la reunión generado desde la plataforma, esta se manda al servidor web que a su vez lo reenvía al correo del usuario administrador.

3.2.3 Plataforma en la nube

Como se ha venido comentando, este elemento dentro del sistema creado se encarga de tres tareas fundamentales: registrar y desconectar a los usuarios en el servidor SIP para que puedan enviar audio, asignarles el mismo nodo de servicio a los usuarios de una misma sala, y llevar a cabo las tareas de reconocimiento y transcripción de los audios que recibe.

El esquema de la plataforma de Vocalia se basa prácticamente en 3 componentes: el primero es un servidor media proxy que contiene al servidor SIP en el que se registran los usuarios; el segundo es un servidor web que controla la lógica de la plataforma durante la comunicación entre otros dispositivos externos mediante HTTP/HTTPS, permitiendo así que cuando se reciba una petición de reserva de recursos por parte de un cliente, este servidor ejecute sus scripts PHP que asignan al cliente un nodo de servicio; y por último, los diferentes nodos de servicios que ejecutan cada uno una máquina virtual en la que están contenidos los diferentes módulos que ofrece la plataforma.

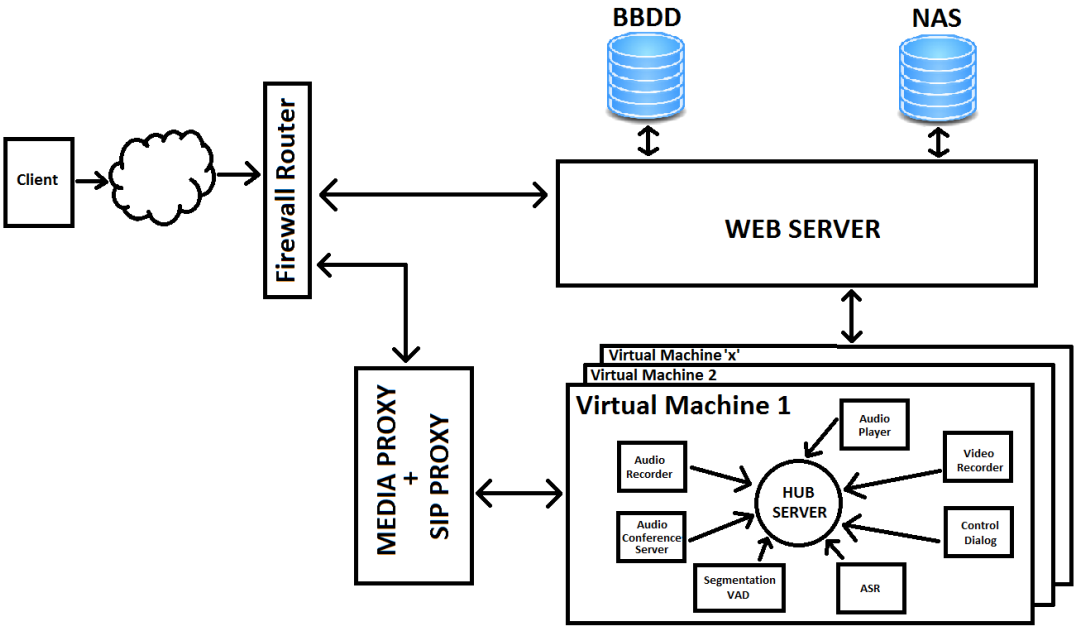

Figura 3-5: Esquema de la plataforma en la nube de Vocalia

Como se ha comentado antes, el registro SIP se ejecuta en un servidor SIP que la plataforma de Vocalia posee dentro de un media proxy, donde los diferentes usuarios que desean utilizar algún módulo (y en defecto, un nodo de servicio) que precise de la tecnología VoIP deben ser registrados. Tras ello, estos usuarios (si tienen permisos y si ya se encuentran asignados a un nodo de servicio o máquina virtual) pueden ejecutar diferentes módulos de la plataforma como son el grabador de voz, el reproductor de vídeo, el ASR, el segmentador VAD, el servidor de audio-conferencia y muchos más módulos.

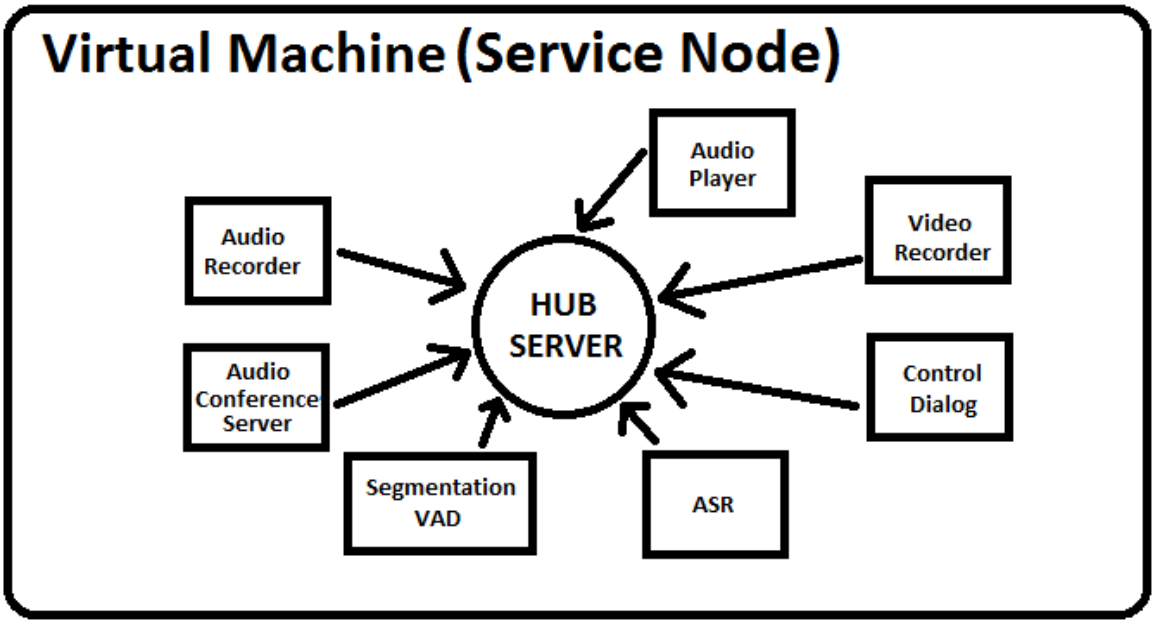

Figura 3-6: Esquema de un nodo de servicio

Tal y como se explicó previamente, para asignar a los usuarios de una misma sala virtual el mismo nodo de servicio, la plataforma, desde su propio servidor web y a partir de los identificadores de sala recibidos desde las peticiones de reserva de recursos creadas por el cliente, ejecuta unos scripts PHP de inicio de sesión de usuario que otorgan a los participantes una dirección SIP activa correspondiente a una sala de audio-conferencia (que funciona con tecnología VoIP) contenida en el nodo de servicio asignado a los usuarios de esa misma sala. Una vez que el usuario ha llamado y la llamada ha sido establecida, el usuario permanece en el servidor de audio-conferencia junto al resto de usuarios conectados, donde se graban simultáneamente los audios de cada usuario.

Cuando un usuario abandona la sala, se generan automáticamente un archivo de audio en formato ".wav" correspondiente a la grabación de su voz anterior, el cual se guarda en un directorio del nodo de servicio común al resto de participantes. Sin embargo, hasta que la sala no se cierre por completo (esto es, que todos y cada uno de los usuarios abandonen la sala, o como se verá en la implementación, que el administrador abandone la reunión), el proceso de segmentación, reconocimiento y transcripción de las grabaciones no se inicia. Cuando se termina la reunión, el servidor web de la plataforma de vocalía recibe por parte del cliente Android la petición HTTP de eliminación de sala. Tras esto, el servidor ejecuta un script PHP que ordena (en un único nodo de servicio determinado) a un módulo gestor del resto de módulos clientes (llamado servidor HUB) ejecutar otro módulo de control de diálogo que contiene un script en JavaScript, el cual contiene las directrices posteriores que debe realizar el HUB. Estas reglas son las relativas al inicio y final de ejecución de un módulo determinado. Así, cuando el HUB reciba la orden de ejecutar el módulo de diálogo desde el servidor web, éste generará un script que el HUB interpreta y que mandará ejecutar los módulos VAD, ASR y generador de arhivos XML (en este mismo orden), para obtener de esta forma un fichero XML con el acta de la reunión que será remitido al servidor web del sistema.

Posteriormente, tal y como se comentó en el estado del arte, este fichero antes de ser enviado al usuario, deberá pasar por una aplicación web independiente a este trabajo (el Transcriber) en la que un humano deberá revisar la transcripción y revisar que no haya habido ningún fallo en la misma, corrigiéndolo si es necesario.

4 Desarrollo

4.1 Herramientas empleadas

En primer lugar, se concretarán y definirán las cuatro principales herramientas empleadas que permiten el desarrollo del proyecto que son: Android Studio [9], WinSCP, MySQL y AudioConference Server.

La primera y más importante es el entorno libre de desarrollo y programación Android Studio, creado en 2013 por Google y sin coste alguno para el desarrollador. Este entorno en sus inicios contenía una gran cantidad de errores y su uso se podía tornar desquiciante debido a su gran lentitud. Sin embargo, actualmente se han corregido muchos de estos errores que le han permitido colocarse como el entorno de programación de Android más empleado, desbancando al otro entorno de programación en Android de Eclipse llamado ADT (Eclipse Android Developer Tools).

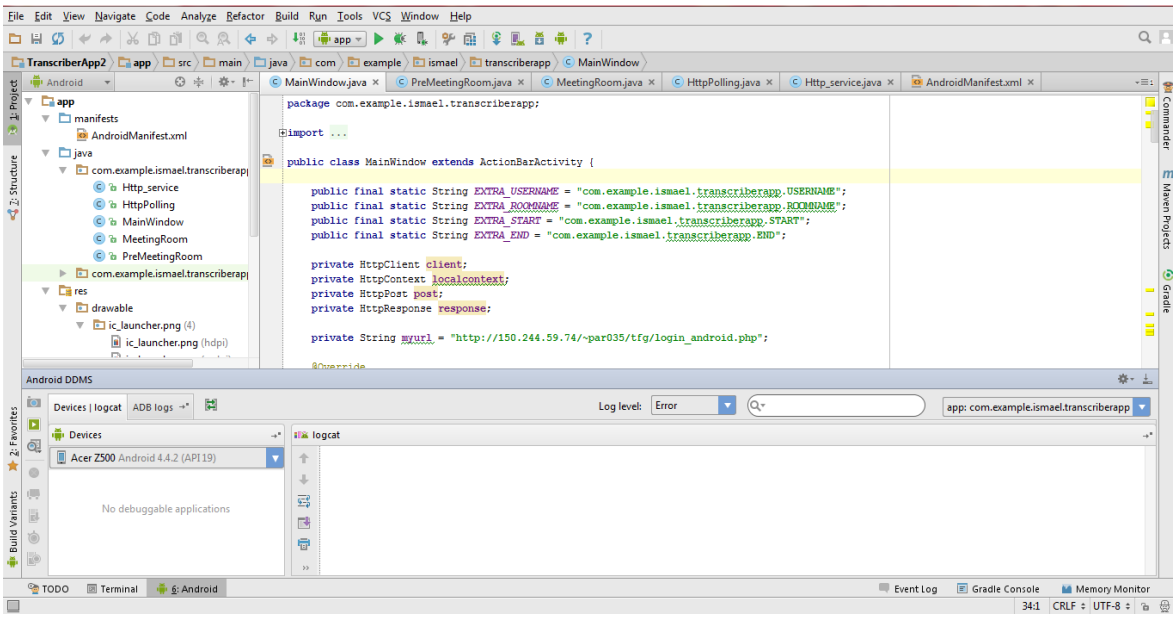

Figura 4-1: Entorno de Android Studio

Es perfectamente compatible con los tres sistemas operativos más comunes del mercado (Windows, Linux y Mac OS), y permite desarrollar todo tipo de aplicaciones que se ejecuten exclusivamente en dispositivos con sistema operativo Android (Smartphones, Tablets, Smartwatches, Google Glass, etc.). El entorno posee un potente editor de código inteligente que ayuda al usuario de forma impecable, proporcionando incluso unos tips (o consejos) que ayudan al desarrollador a ser más productivo a la hora de escribir el código. Además, para poder probar la aplicación diseñada, Android Studio posee su propio emulador con distintos y predefinidos perfiles móviles, es decir, desde el ordenador se puede emular una aplicación con el mismo perfil de móvil en el que el usuario instalará y ejecutará la aplicación.

Figura 4-2: Emulador incorporado en Android Studio

No obstante, éste método de simulación y prueba de la aplicación es considerablemente lento, por tanto, en el desarrollo de la aplicación Android se ha empleado el propio dispositivo móvil como emulador, instalando desde el ordenador la aplicación en el teléfono y posteriormente probando su funcionamiento.

Cabe destacar que las aplicaciones Android requieren de un mínimo de dos lenguajes de programación: Java y XML, pues es en estos dos lenguajes con los que se programará la parte funcional de la aplicación y la parte de diseño del layout respectivamente. Es por esto que el editor de código permite escribir en ambos lenguajes, siempre dando la opción de autocompletar el código y por tanto facilitando las labores de programación.

La parte correspondiente al código XML no solo es relativa al diseño del Layout, es decir, al diseño estético de la aplicación, sino que también se emplea para ajustar las configuraciones básicas de la aplicación y, sobre todo, para otorgar a la aplicación los permisos necesarios para que pueda utilizar ciertos componentes del móvil como puede ser la conexión a internet, las llamadas VoIP, la configuración del audio del teléfono y conexión Bluetooth entre muchos otros. Esto tiene sentido pues estos componentes (o recursos) se emplean en otras aplicaciones y el sistema operativo debe mantener cierto control sobre los mismos.

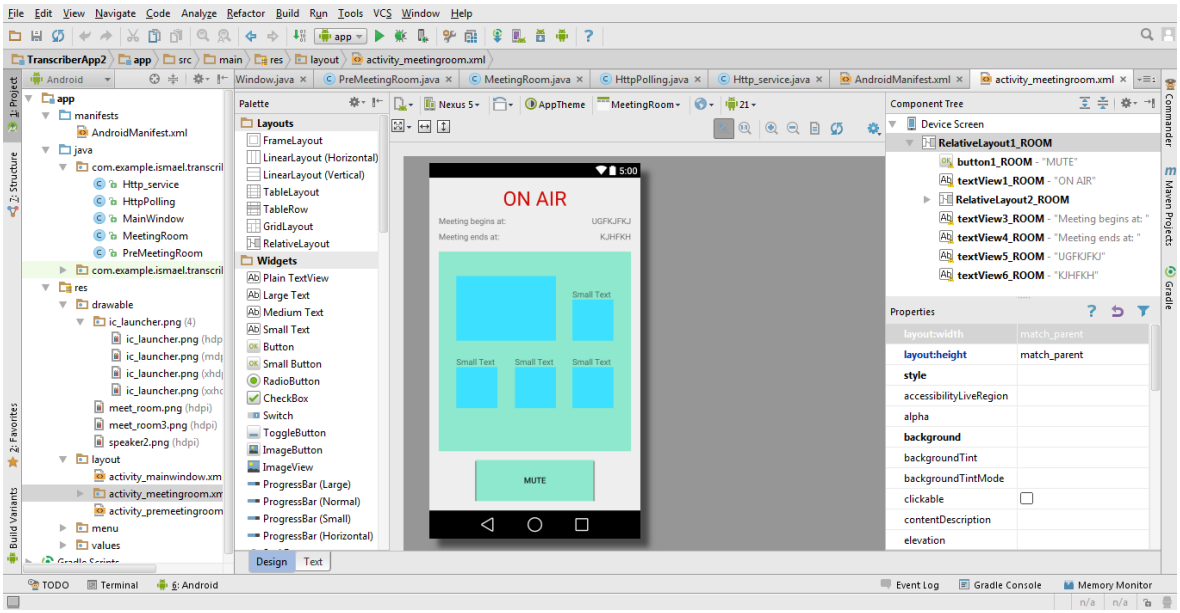

Figura 4-3: Entorno de diseño gráfico de Android Studio

La principal ventaja de Android Studio es que puede emplear conjuntamente el layout desarrollado con XMLs junto a los scripts Java que incorporan la parte funcional de la aplicación, pudiéndose así editar el layout creado previamente desde cualquier fichero Java sin necesidad de integrar ningún código XML dentro del fichero Java.

Por otro lado, la segunda herramienta empleada es WinSCP, una aplicación de software libre para Windows que reproduce un cliente SCP utilizando SSH, con lo cual permite la transferencia segura de archivos entre dispositivos remotos. En este proyecto, este cliente posibilita la transferencia (y editado) de los scripts PHP entre el ordenador local de trabajo y el servidor web, consintiendo así el desarrollo de, por una parte, la aplicación web, y por otra, de las comunicaciones entre los diferentes componentes del sistema (clientes android, servidor web y plataforma de Vocalia).

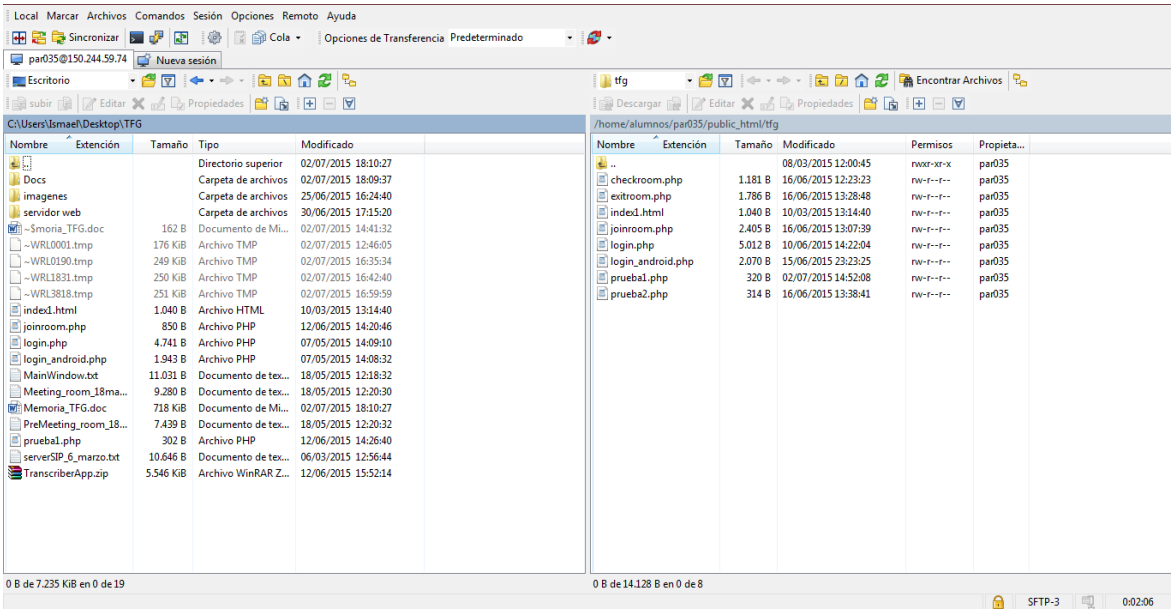

Figura 4-4: Entorno de WinSCP

La tercera herramienta empleada en este trabajo es MySQL, un sistema de gestión de bases de datos con propósitos de desarrollo de aplicaciones web. El servidor web de este trabajo dispone de este gestor que permite el diseño de la base de datos del transcriptor, la cual como pronto se verá, dispone de tres sencillas tablas.

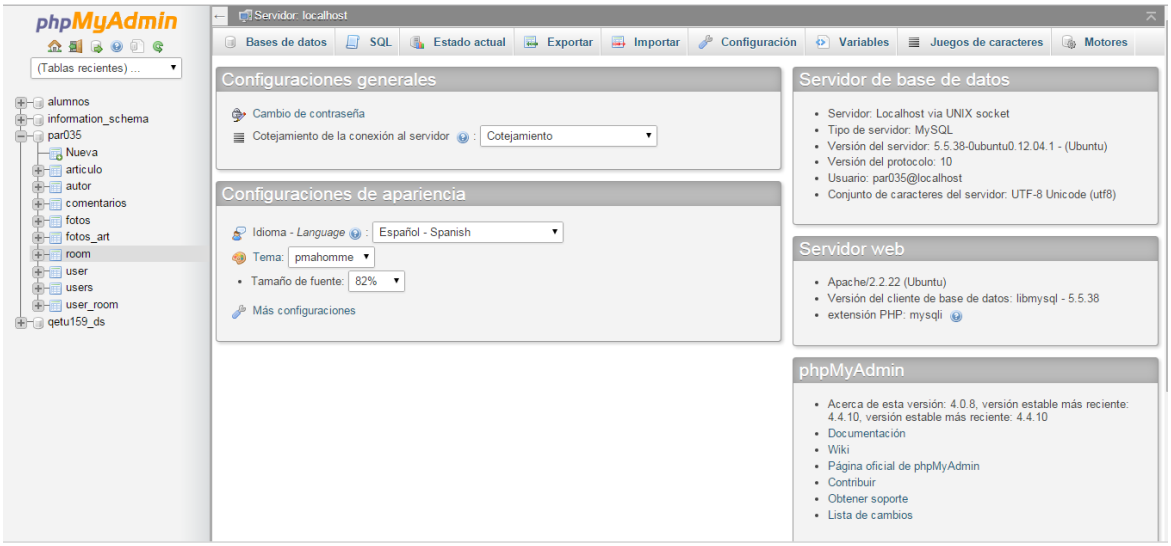

Figura 4-5: Entorno de MySQL

Por último se encuentra el AudioConference Server, del cual se sabe que es uno de los módulos dentro del nodo de servicio que permite visualizar de forma paralela el audio y la información de cada usuario conectado a la conferencia. Además, este módulo permite grabar y aislar los audios de los usuarios que lo utilizan para su posterior segmentación y transcripción. Esta aplicación permite hasta un máximo de cuatro usuarios simultáneamente conectados, es decir, solo se podrían realizar reuniones con menos de cinco participantes.

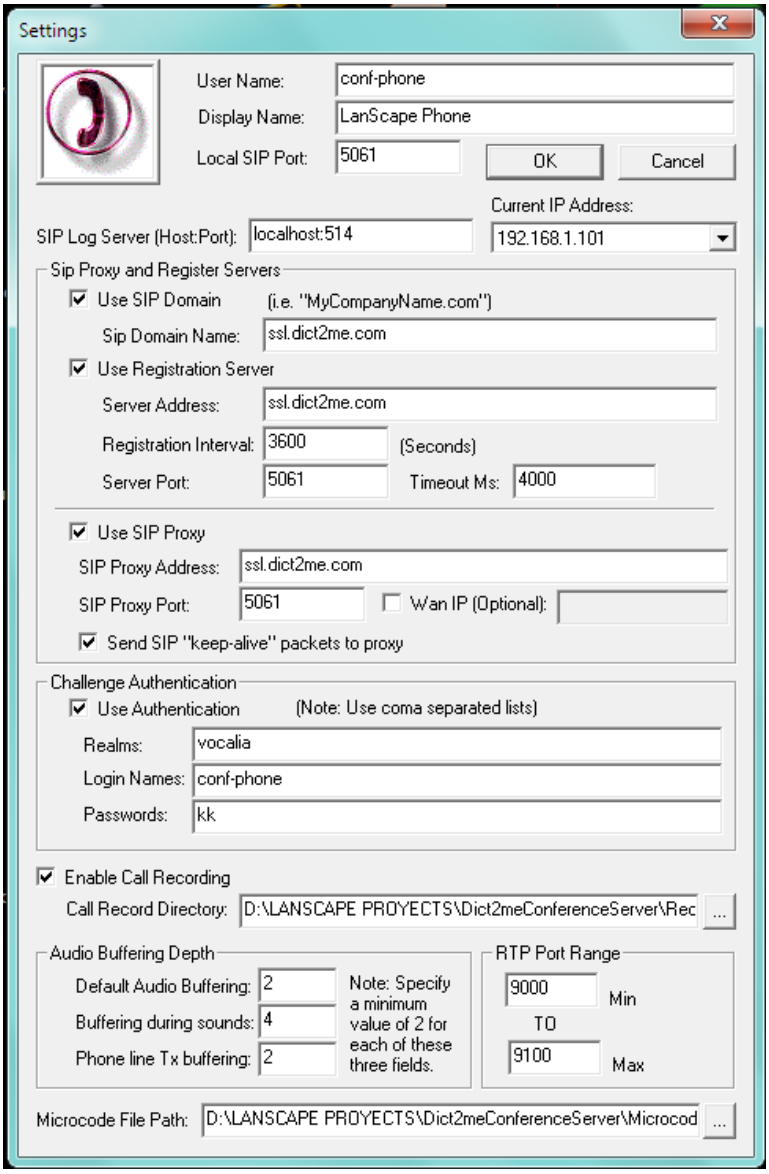

Figura 4-6: Asistente de configuración del AudioConference Server

4.2 Implementación

En este apartado se verá detalladamente la implementación software de todos y cada uno de los componentes, así como de sus funcionalidades.

4.2.1 Bases de datos

Para llevar a cabo el desarrollo de las bases de datos se he empleado la herramienta MySQL. Con ello, se ha creado una base de datos que contiene únicamente tres tablas, una para los usuarios denominada “users”, otra para las salas virtuales de reunión llamada “room”, y una última tabla llamada “user_room” que relaciona ambas tablas con el fin de poseer un registro sobre los usuarios pertenecientes a una determinada sala.

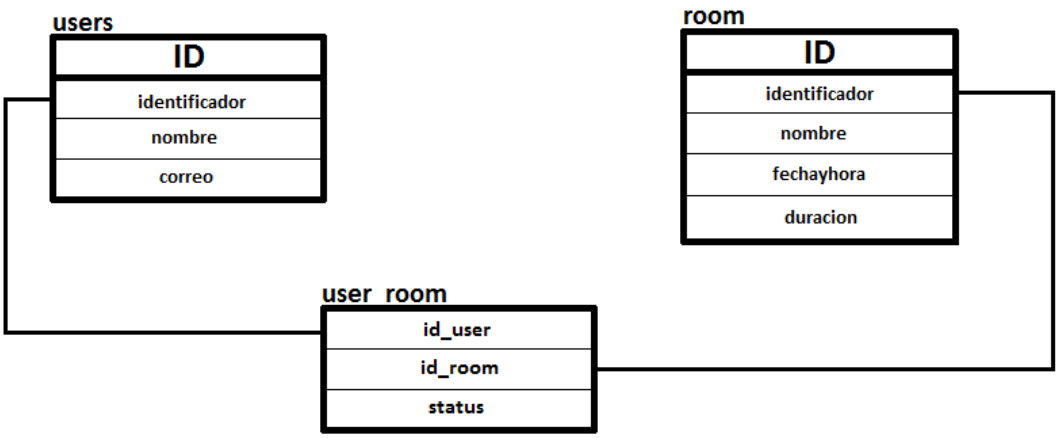

Figura 4-7: Diagrama de Bases de Datos

En la tabla de usuarios se encuentran datos relativos al usuario en cuestión, como son su nombre, su correo y su identificación. Todos ellos son campos obligatorios a rellenar por el usuario a través de la aplicación web de la que próximamente se hablará. Concretamente, el identificador es un código único obtenido al aplicar un hash criptográfico (md5) sobre el correo de cada usuario, el cual sirve para realizar la identificación del usuario cuando éste se conecte a la aplicación Android. Cabe recalcar que por el momento el uso fin de este identificador es únicamente el anterior mencionado, aunque en posteriores actualizaciones de la aplicación el identificador también se usara para construir la dirección SIP del cliente y así evitar confusiones de usuarios por parte del servidor SIP de Vocalia.

La tabla "room" es similar a la tabla de usuarios, posee un identificador de sala y un nombre únicos, pero en esta ocasión requiere de un campo llamado "fechayhora" que contiene una cadena con la fecha y la hora a la que se producirá la reunión, y un campo "duración" que contiene la duración estimada de la misma. Ambos campos se utilizan para calcular la hora de finalización de la reunión, con la que posteriormente la aplicación Android será capaz de dar por finalizada la reunión, y con la que la aplicación web podrá eliminar todo lo relativo a esta reunión acabada (la sala virtual correspondiente y sus usuarios asignados).

La última tabla es "user_room" y contiene los dos campos correspondientes a los identificadores de sala y de usuario, permitiendo la asociación de varias salas a un mismo usuario. Además posee un tercer campo denominado "status" que indica para el usuario de una determinada sala su estado de conexión a la misma (conectado y no conectado). A continuación se muestran tres imágenes correspondientes a un ejemplo de lo que pueden contener estas tablas. En ellas se observa la creación de una sala virtual (misala1) y de los cuatro usuarios que la conforman, los cuales se han registrado a partir de la aplicación web.

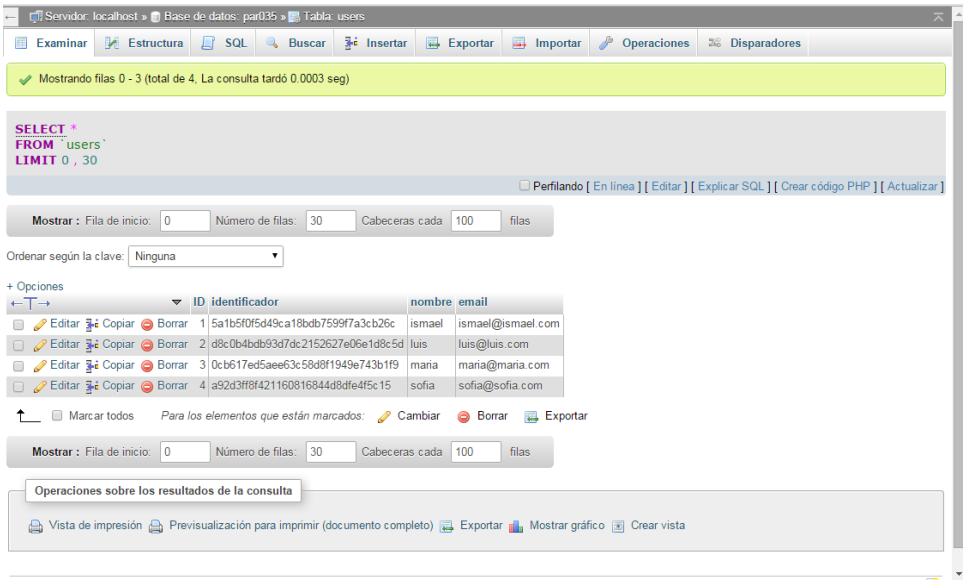

Figura 4-8: Contenido de la tabla “users”

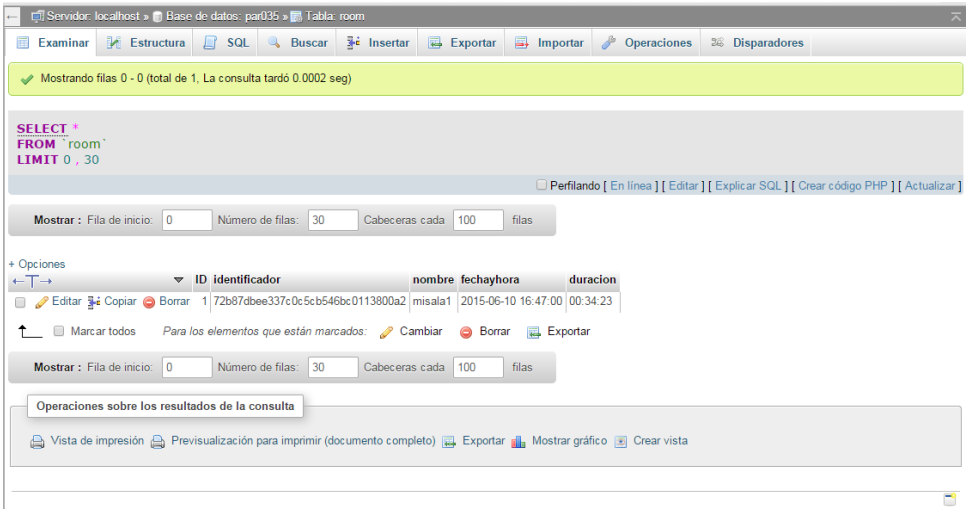

Figura 4-9: Contenido de la tabla “room”

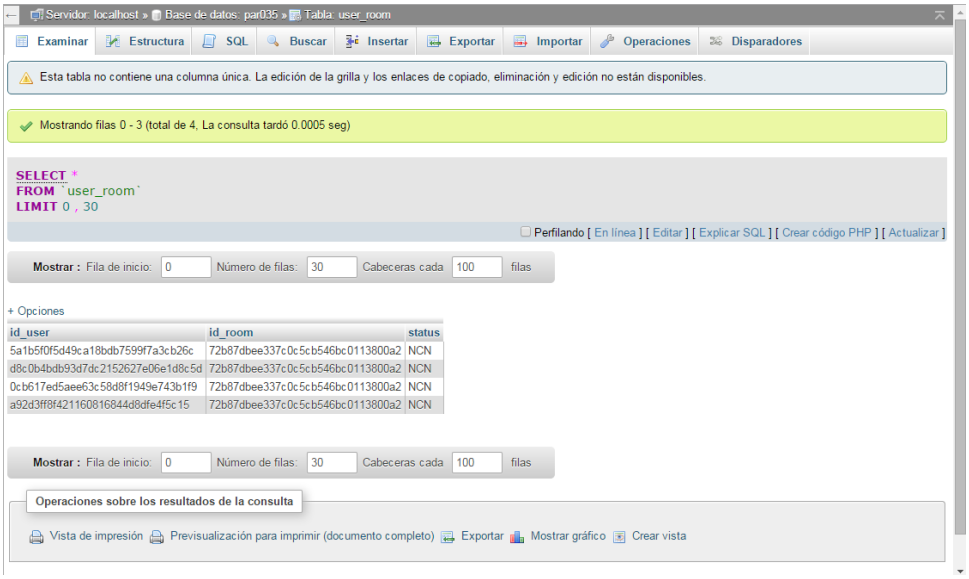

Figura 4-10: Contenido de la tabla "user_room"

Es importante remarcar que la inserción de los usuarios en la tabla "user_room" viene determinado según el formulario de la aplicación web, de manera que el primer usuario introducido por esta aplicación será el primer usuario introducido en la tabla, lo que permite (sin necesidad de añadir ningún campo más) identificar al administrador de una misma sala.

4.2.2 Aplicación web

En esta sección se detallará todo lo que concierne entorno a la aplicación web, la cual ha sido construida y diseñada en un servidor independiente de la plataforma de Vocalia y mediante la aplicación WinSCP, y que servirá para crear la sala virtual de reuniones con la que los usuarios registrados en la misma podrán llevar a cabo una audio-conferencia con sus dispositivos móviles.

En primer lugar cabe destacar que la aplicación web diseñada dispone únicamente de una ventana y sólo es funcional, en otras palabras, su interfaz de usuario es lo más sencilla posible, tal y como se muestra en la siguiente imagen.

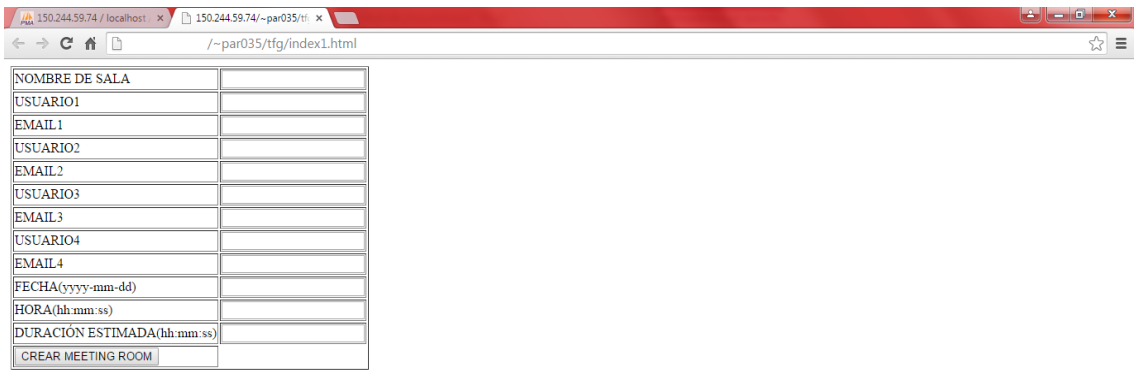

Figura 4-11: Aplicación web

En la figura previa, se observa que el usuario (y administrador) debe rellenar el formulario de forma completa, es decir, no puede dejar ningún campo vacío, lo que fuerza a que el usuario que rellena el formulario siempre cree una sala con otros tres usuarios más. Esta decisión se tomó con el objetivo de probar la audio-conferencia al máximo de su capacidad (lo que permite por el momento el AudioConference Server), aunque en el futuro esta aplicación web deberá permitir la generación de audioconferencias con dos o tres participantes.

Próximamente, la aplicación web dispondrá de una estética diferente y atractiva para el usuario, incorporando una ventana nueva de presentación de la aplicación previa a la ventana de registro anterior.

Otro punto importante es que el administrador debe insertar sus datos en el formulario correspondiente a "usuario1". Como se viene diciendo, esta interfaz es muy simple, y en el futuro se corregirá la aplicación para que el usuario número uno sea asignado como adminisitrador y para que se puedan crear salas con dos y tres participantes.

4.2.3 Aplicación Android

La aplicación Android desarrollada tiene dos partes diferenciadas, una correspondiente a la parte funcional y otra a la interfaz de usuario. La parte funcional se centra en las comunicaciones entre el cliente y los demás servicios, las cuales son llevadas a cabo por peticiones HTTP (en el caso de la gestión de la sala y de los usuarios) y por llamadas SIP (correspondientes al intercambio de flujo de audio durante la audio-conferencia). La parte de interfaz de usuario se centra básicamente en el aspecto de la aplicación, aunque al igual que con la aplicación web también se debe seguir un diseño mínimo que permita utilizar la aplicación.

En lo que concierne a la parte funcional, cabe destacar que la aplicación se ha dividido en tres ventanas posibles denominadas en Android como actividades (activities). La primera actividad se encarga de identificar al usuario y la sala a la que quiera acceder a partir de los identificadores obtenidos del registro de los usuarios.

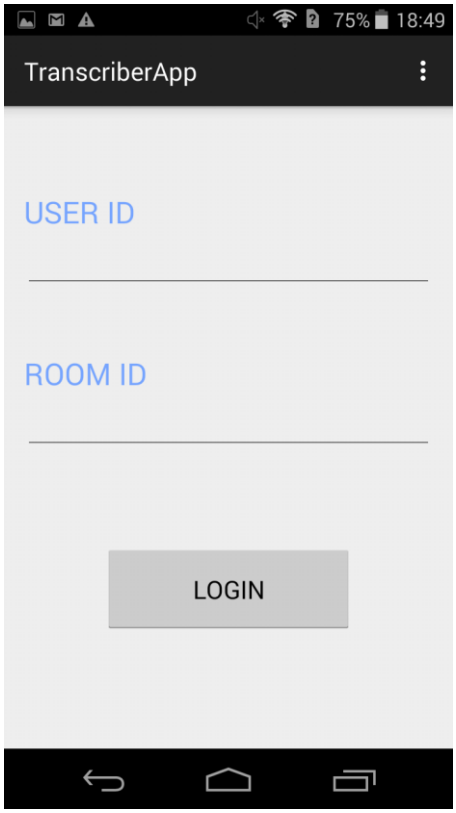

Figura 4-12: Primera activity

Para ello, como se mencionó en el apartado de diseño, se emplean clases definidas en Android que permiten enviar peticiones HTTP y recibir respuestas de las mismas. En el caso de la identificación, este HTTP es de tipo POST, ya que obviamente los identificadores no deben ir en la URL. La respuesta obtenida de esta identificación viene dada por el servidor web en el formato JSON, y en ella se incluyen todos los datos relativos al usuario y a la sala (los nombres y las fechas de inicio y de final de la reunión). Para las próximas peticiones HTTP se emplea el método GET y los nombres de sala y usuario obtenidos anteriormente.

```
private InputStream sendRequest(String myurl, String ID_user, String ID_room){
    client = new DefaultHttpClient();
    localcontext = new BasicHttpContext();
    response = null;
    post = new HttpPost(myurl);

    ArrayList<NameValuePair> parameters = new ArrayList();
    parameters.add(new BasicNameValuePair("username", ID_user));
    parameters.add(new BasicNameValuePair("roomname", ID_room));

    //enviar respuesta, metiendo primero los parametros
    try {
        post.setEntity(new UrlEncodedFormEntity(parameters));
        response = client.execute(post,localcontext);
        return response.getEntity().getContent();
    } catch (Exception e) {
        e.printStackTrace();
    }

    return null;
}
```

Figura 4-13: Extracto de código correspondiente a la petición HTTP POST

Si la identificación del usuario y de su sala ha sido satisfactoria, se pasa a una segunda actividad donde se le da la bienvenida al usuario, mostrándole la sala que le corresponde e indicándole que debe pulsar el botón con el nombre de la misma para unirse e iniciar la audio-conferencia.

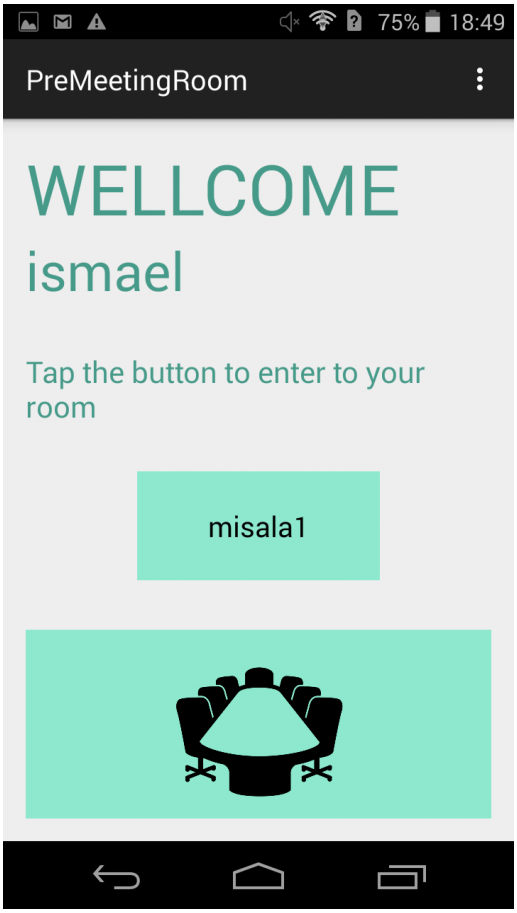

Figura 4-14: Segunda Activity

Tras pulsar el botón anterior, se accede a la tercera y última actividad donde se produce todo el intercambio de información. Nada más saltar esta ventana, el teléfono recoge todo el audio entrante y lo envía mediante la llamada de audio SIP, tal y como se observa en la pantalla del teléfono, donde se muestran las palabras "ON AIR" que indican que se está enviando audio. El usuario dispone de un botón en la parte inferior que le permite silenciar su llamada pero que en ningún caso permite cuelga y acaba la llamada, es decir, simplemente manda silencio. Al pulsarlo sucede lo anteriormente comentado además de cambiar las palabras "ON AIR" a "MUTED", con lo que el usuario podrá saber si está enviando o no audio. Esta funcionalidad se incorporó por dos razones, la primera tiene que ver con que en las reuniones tradicionales es habitual que los participantes silencien su micrófono si éstos no quieren ser grabados, y la segunda es porque si el usuario silencia su audio cuando no habla, el audio grabado en ese momento no contendrá ruido alguno, lo que provoca que a la hora del reconocimiento el segmentador VAD excluya directamente el segmento correspondiente a este silencio y evitando así posibles errores futuros de transcripción.

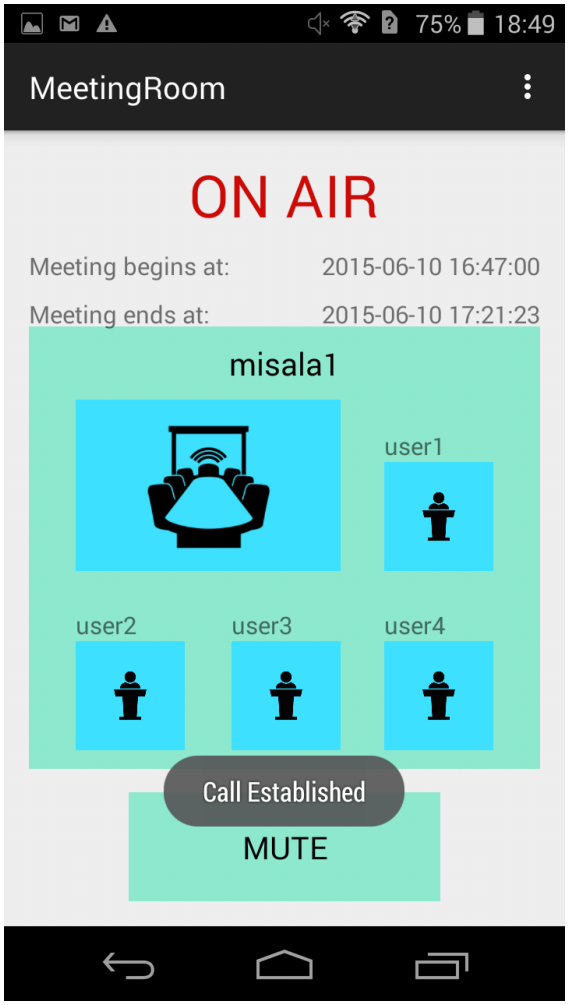

Figura 4-15: Tercera Activity

En cuanto al intercambio de datos entre los servidores y el cliente, cabe reseñar que estas comunicaciones se inician nada más acceder a esta actividad y son parte del back-end. En primer lugar, como se mencionó en el apartado de diseño, el cliente se conecta y registra directamente al servidor SIP situado en la plataforma de Vocalia. Para ello hace uso de la librería SIP integrada en la API de Android Studio, con la cual se manda una petición de registro a la plataforma. No obstante, es necesario que el cliente tenga una dirección SIP válida, la cual se forma automáticamente a partir del nombre del usuario, del dominio SIP con el que se está comunicando y del puerto de conexión empleado. Por ejemplo, si el usuario es Luis, el domino del servidor SIP es ssl.dict2me.com y el puerto asignado es el 5061 entonces la dirección SIP del cliente generada será sip:luis@ssl.dict2me.com:5061.

```
try {
    //a SipProfile object is created
    SipProfile.Builder builder = new SipProfile.Builder(this.user,SIP_PROXY_SERVER_IP);
    //builder.setPassword("passwoooord");
    builder.setPort(SIP_PROXY_SERVER_PORT); //Fixed port
    profile = builder.build();

    //opens the local profile for making SIP calls
    Intent i1 = new Intent();
    PendingIntent pi = PendingIntent.getBroadcast(this, 0, i1, Intent.FILL_IN_DATA);
    manager.open(profile, pi, null);

    if(manager.isOpened(profile.getUriString())==true){
        Log.e("This profile is opened", profile.getUriString());
    }
```

Figura 4-16: Extracto de código correspondiente al registro SIP

Tras establecerse el registro SIP, la aplicación ejecuta ahora otra petición HTTP con destino al servidor web con el objetivo de que este servidor informe a su vez a la plataforma de Vocalia para así obtener la dirección SIP a la que llamar. Esta petición es de tipo GET y ejecuta un script PHP que reenvía (mediante otra petición HTTP GET) a la plataforma la información necesaria para la reserva de recursos. Estos datos requeridos por la plataforma son el nombre del usuario en cuestión, el nombre de la sala a la que va a acceder y un número entero indicando el número de usuarios que existe en la reunión, y con ellos el servidor logra establecer un mismo nodo de servicio para los usuarios de una misma sala. En este punto sí que se añadió la información relativa al número de clientes, por lo que para que el sistema soporte audio-conferencias de dos o tres usuarios, solo habría que reestructurar levemente la base de datos y evidentemente reajustar la aplicación web.

```
private void Login(){
    //conectar con la base de datos MYsql, comprobar que los IDs introducidos son iguales que los de las tablas
    //hacer uso de un hilo independiente para hacerlo de forma asÃ-ncrona.

    InputStream resp;
    String url_prueba = "http://150.244.59.74/~par035/tfg/joinroom.php";
    //"http://ssl.dict2me.com/services/dict2me/dev/reuniones/joinroom.php"
    //resp = sendRequestGET(url_prueba, room, user);
    //String sResponse = responseInString(resp);
    //Log.e("PRUEBA",sResponse);

    //final ArrayList<String> dataJson = ResponseOfSIPserver(sResponse);
    String myurl1=url_prueba+"?roomname="+this.room+"&username="+this.user;
    ArrayList<String> dataJson = new ArrayList();
    try{
        dataJson = new Http_service().execute(myurl1).get();
    }
    catch(Exception e){
        e.printStackTrace();
    }
```

Figura 4-17: Extracto de código correspondiente a una petición HTTP GET

La respuesta obtenida por parte de la plataforma contiene tres campos importantes: el más significativo es la dirección SIP del nodo de servicio asignado a los dispositivos de una misma sala con el cual el usuario ya registrado puede realizar la llamada de VoIP, después entrega un listado con los usuarios conectados en ese preciso momento, y por último entrega una bandera indicando si la transacción de información ha sido satisfactoria.

Una vez obtenida la dirección SIP a la que llamar, el dispositivo llama utilizando la misma librería SIP empleada para el registro.

```
private void initiateCall(){
    //initiate a SIP call
    try {
        SipAudioCall.Listener listener = new SipAudioCall.Listener() {
            @Override
            public void onCallEstablished(SipAudioCall call) {
                call.startAudio();
                //call.setSpeakerMode(true);
                if(call.isMuted()){
                    call.toggleMute();
                }

                updateStatus("Call Established");
            }

            @Override
            public void onCallEnded(SipAudioCall call) { updateStatus("Call Ended."); }

            @Override
            public void onCalling(SipAudioCall call) { updateStatus("Calling..."); }

            @Override
            public void onCallBusy(SipAudioCall call) { updateStatus("The peer is busy.");

            @Override
            public void onError(SipAudioCall call, int errorCode, String errorMessage){
                updateStatus(errorMessage);
            }
        };
        call = manager.makeAudioCall(profile.getUriString(), sipAddress, listener, 30);
```

Figura 4-18: Extracto de código correspondiente a la petición de inicio de llamada

Si el servidor de audio-conferencia esta en escucha (hecho que siempre se produce si se han asignado los recursos pertinentes), se inicia el proceso de audio-conferencia entre el resto de usuarios, almacenando los audios de cada uno de ellos y pudiendo hablar unos con otros de forma simultánea.

De forma simultánea, y también en el back-end de la aplicación, se produce el polling anteriormente comentado con el objetivo de saber aproximadamente en tiempo real el estado de los usuarios. Básicamente se manda una petición HTTP cada 3 segundos en el que se pregunta al servidor web sobre el campo de conexión “status” de cada uno de los usuarios de una misma sala (que si se recuerda, estaba situado en la tabla user_room), devolviendo una lista con los usuarios conectados. Esta lista luego será empleada para cambiar el layout en función de los mismos, tal y como se refleja en la siguiente figura.

No obstante, la funcionalidad de la aplicación durante la llamada no se queda aquí, pues se ha establecido que si, aún con la llamada aún activa, existen unos auriculares (con micrófono incorporado) conectados al dispositivo, la aplicación accede automáticamente al micrófono de los mismos para enviar la voz en vez de acceder al micrófono y altavoz del móvil, algo que es esencial pues en este trabajo también se analizará el efecto de los diferentes auriculares sobre los audios grabados y sobre la calidad de la reunión.

```
public void onRegistrationDone(String localProfileUri, long expiryTime) {
    updateStatus("Ready");
    Log.e("ExpiryTime",String.valueOf(expiryTime));
    if(expiryTime!=-1){
        Login();
        Log.e("PRUEBA", "LOGUEOOO.....");
        if (audioManager.isBluetoothA2dpOn()) {
            // Adjust output for Bluetooth.
            audioManager.setMode(AudioManager.MODE_IN_COMMUNICATION);
            audioManager.setSpeakerphoneOn(false);
            audioManager.setBluetoothScoOn(true);
            audioManager.startBluetoothSco();
            Log.e("AUDIO","bluetooooothhhh");
        } else if (audioManager.isSpeakerphoneOn()) {
            Log.e("AUDIO","altavooooozzzz");
            // Adjust output for Speakerphone.
        } else {
            // If audio plays and noone can hear it, is it still playing?
            Log.e("AUDIO","no suena naaaah");
        }
        initiateCall();
```

Figura 4-19: Extracto de código correspondiente a la configuración de auriculares Bluetooth

Por último, entorno a la desconexión de la sala de reuniones, cabe decir que hay dos formas de terminarla: la más simple consiste en que si el usuario que es administrador abandona la sala, inmediatamente se expulsa al resto de usuarios (que saben de la desconexión de éste por el polling que realizan constantemente) y se envía una petición HTTP al servidor web y a la plataforma informando de la conclusión de la reunión; y la otra forma de eliminarla es de forma manual pulsando el botón de "atrás" del dispositivo. Al pulsarlo, aparecerá una pequeña ventana de diálogo como el de la siguiente imagen que preguntará al usuario si quiere cambia de usuario, lo que se traduce en colgar la llamada y salir de la reunión para no poder volver a entrar.

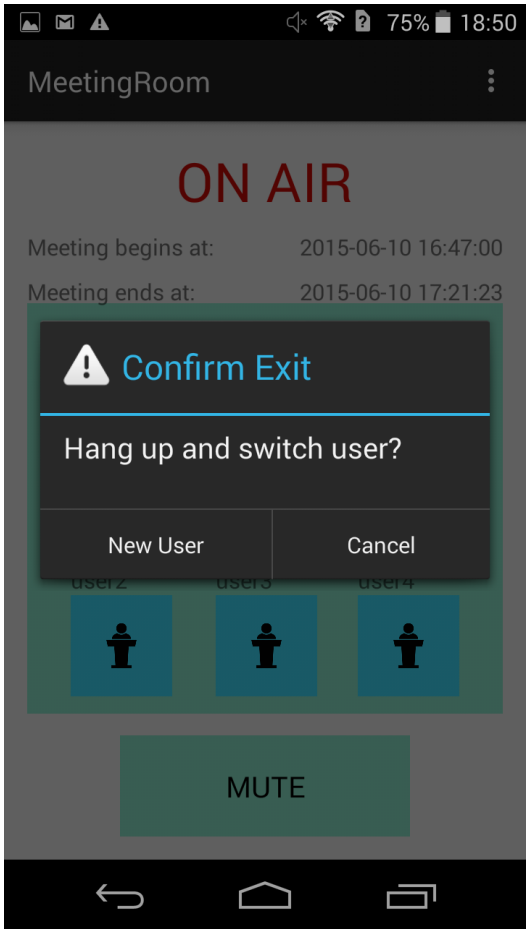

Figura 4-20: Solicitud de desconexión de usuario

Como se ha mencionado al final del anterior párrafo, este sistema está diseñado para que el usuario desconectado no pueda volver a iniciar otra llamada. Esto se debe a que el servidor de audio-conferencia recoge tantos audios como conexiones se hayan establecido, es decir, si un usuario se conectase tres veces a la audio-conferencia se obtendrían tres audios del mismo usuario que luego no se podrían recomponer temporalmente al realizar la transcripción.

5 Pruebas y resultados

5.1 Pruebas

En este capítulo se comentarán las pruebas realizadas sobre este sistema de transcripción de reuniones. En primer lugar, el primer ensayo obviamente es verificar el correcto registro del cliente en cuestión al servidor SIP, y posteriormente comprobar si el audio ha sido enviado correctamente desde el dispositivo al servidor y que éste ha sido grabado por uno de los módulos de grabación de audio. En la imagen inferior se muestra además del registro y desconexión SIP, el establecimiento e inicio de la llamada de VoIP entre un nodo de servicio (cuya dirección SIP es sip:phone-36-4@ssl.dict2me.com) y un cliente Android (sip:ismael@ssl.dict2me.com).

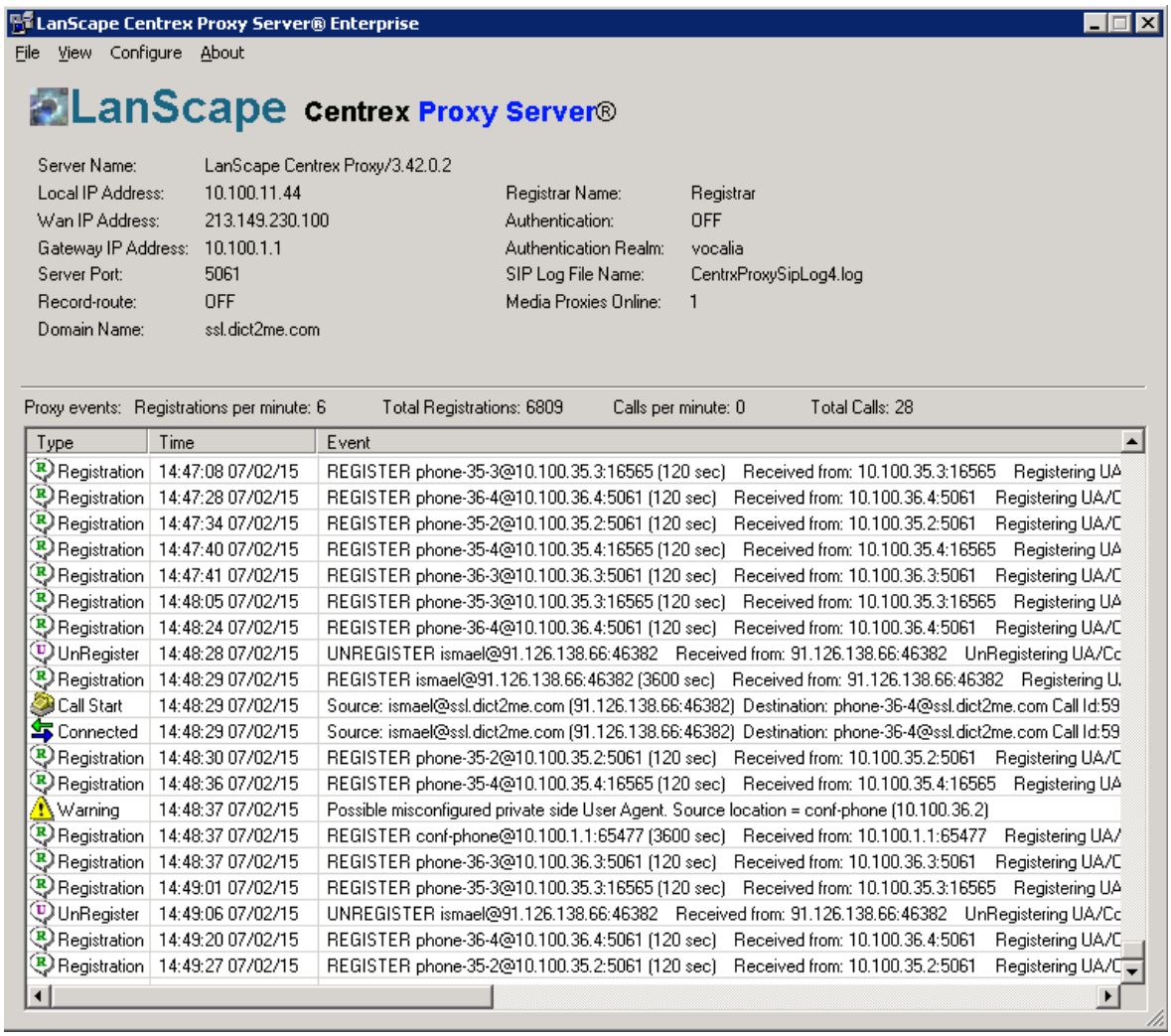

Figura 5-1: Visualización del servidor del registro SIP y del inicio de llamada

Cabe destacar que se ha probado a grabar el audio con diferentes tipos de auriculares con micrófono incorporado, tanto inalámbricos (Bluetooth) como cableados. Los resultados obtenidos tras estas pruebas de grabación se comentarán en el próximo apartado.

En la imagen siguiente, se puede observar la grabación del audio mediante un módulo de grabación disponible en la plataforma. Este audio ha sido enviado desde un cliente Android a un nodo de servicio.

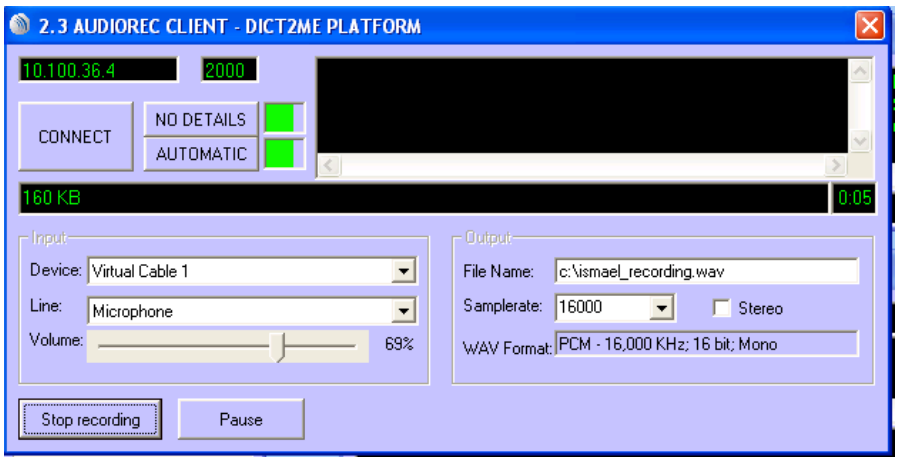

Figura 5-2: Grabación de un audio

Después se verifica la conexión de múltiples clientes a este servidor de audio-conferencia, los cuales pueden escucharse unos a otros a través de sus auriculares. En la próxima figura se muestra la audio-conferencia establecida entre cuatro clientes de este sistema, donde todos ellos intercambian diálogos simultáneamente.

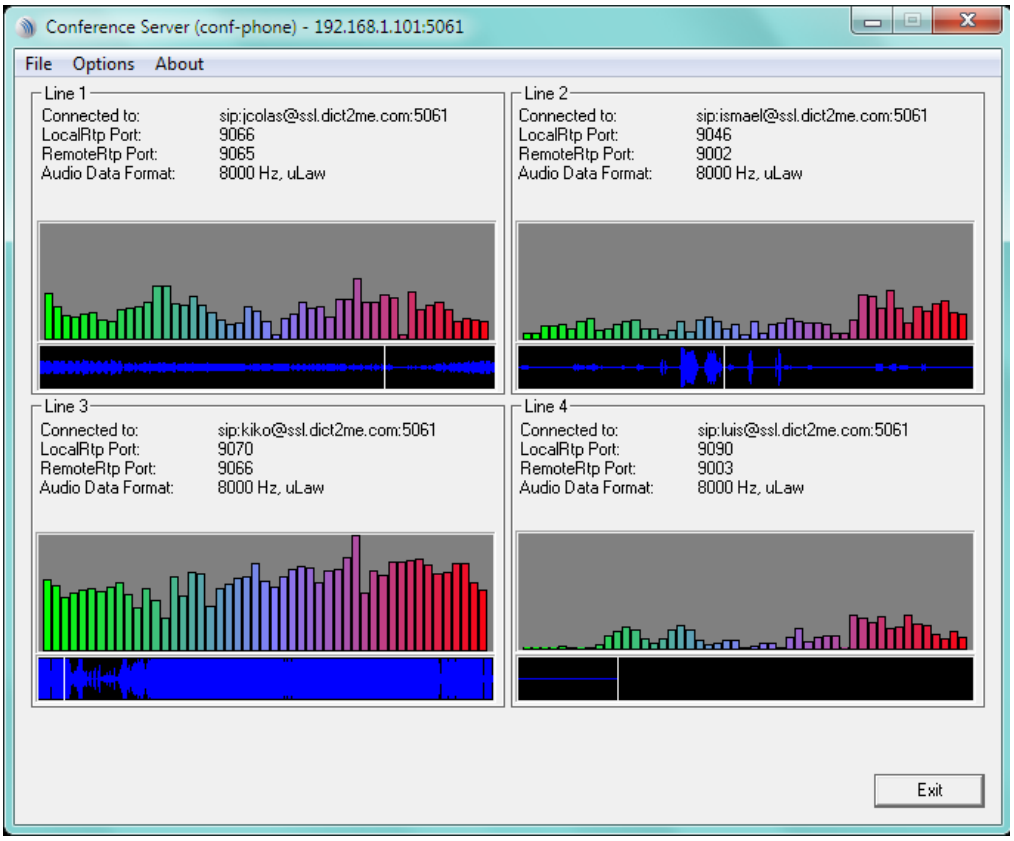

Figura 5-3: AudioConference Server en ejecución

En este punto se comprueban dos cosas: al igual que antes, se comprueban los audios grabados por cada usuario (situados en un directorio del nodo de servicio asignado) y generados al terminar la audio-conferencia, los cuales han sido grabados con diferentes micrófonos para discutir posteriormente la calidad de grabación de los mismos. La segunda comprobación consiste en verificar de forma práctica el correcto diálogo entre los usuarios cuando estos utilizan el sistema como conferencia remota, es decir, se comprobará si todos los integrantes (que se encuentran físicamente separados) pueden hablar y escucharse entre sí sin problemas. Esta última prueba se realizará varias veces, pues en este caso interesa comprobar la audio-conferencia utilizando todos los participantes el mismo modelo de auriculares.

5.2 Resultados

Después de realizar todas las pruebas pertinentes y anteriormente comentadas, se comentarán los resultados experimentales obtenidos tras su ejecución.

Los dispositivos empleados en estas pruebas son los siguientes:

- Auriculares Acer Liquid
  - Auriculares cableados
  - Sin cancelador de ruido

- LG HBS 730
  - Auriculares de cuello inalámbricos Bluetooth
  - Con cancelador de ruido

- Logitech HS 110
  - Auriculares cableados
  - Con cancelador de ruido

En primer lugar, se destaca que los tres dispositivos probados registran correctamente el audio generado por cada hablante, es decir, los ficheros de audio grabados no contienen ningún tipo de distorsión ni corte, aunque bien es cierto que existen diferencias entre los micrófonos, pues a pesar de que estos audios no se corten, el ruido captado por cada uno es diferente entre sí. En el caso de los auriculares Logitech, el micrófono que incorporan es muy poco direccional, de tal manera que capta con gran intensidad el ruido de fondo, lo cual provocará una ineficaz segmentación y reconocimiento. El auricular cableado de Acer cancela algo más el ruido, pero no tiene ni punto de comparación con el auricular de cuello LG. El micrófono de este último no detecta el más mínimo ruido cercano, incluso aun teniendo a otro participante hablando a escasos centímetros de otro, permitiendo así una correcta segmentación y reconocimiento futuras.

El problema principal radica en la audio-conferencia. Para probar que se puede establecer una reunión a distancia, se probó a realizar esta audio-conferencia con los participantes separados en habitaciones diferentes y con diferentes auriculares. El usuario que utilizaba los cascos LG no escuchaba fluidamente a los demás participantes puesto que se producían de vez en cuando cortes en la voz que hacían por momentos ininteligible el audio recibido. Sin embargo, los otros tres usuarios que utilizaron los cascos Logitech y Acer si permitían tener una conversación fluida, ya que en rara ocasión el audio se cortaba. El hecho de que con los cascos LG no se pueda seguir la conversación se debe probablemente a dos razones: la primera y más probable es que la librería estándar de Android (utilizada en este trabajo) que permite incorporar la tecnología Bluetooth a una aplicación deja bastante que desear, y la segunda razón menos probable es que el protocolo Bluetooth en determinadas circunstancias especiales puede desechar con facilidad sus paquetes de voz. Aun así, en todos los casos el audio grabado por el módulo de audio-conferencia era de peor calidad que en el escenario anterior donde los usuarios estaban físicamente juntos debido precisamente a estos cortes.

Parecerá extraño que los audios grabados por el AudioConference Server tengan una calidad diferente a la de los mismos audios escuchados por el resto de usuarios, pero esto tiene una explicación. Resulta que el audio inicialmente pasa por el grabador que tiene incorporado este módulo, lo cual supone que sólo exista un único intercambio de datos (entre hablante y servidor). De forma simultánea, este audio es reenviado al resto de dispositivos mediante la tecnología VoIP (lo cual supone dos intercambios de datos, entre hablante y servidor, y entre servidor y oyentes), la cual sabemos que en ciertas ocasiones no cumple cierta calidad de servicio (QoS), tanto por lado del que habla como por el lado de los que escuchan. Por tanto, estos cortes (que muy raramente aparecen en los audios grabados) aparecen con mucha facilidad en el lado del oyente.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Tras haber analizado el estado del arte en cuanto a transcripción de reuniones y técnicas de procesamiento de voz, se llega a la conclusión de que es factible la implementación de un módulo automático de transcripción situada en la red.

Para ello, inicialmente se diseñó una arquitectura basada en cliente/servidor que permitía la comunicación con la plataforma de Vocalia, la cual incorporaba los módulos de audio-conferencia, segmentación VAD y reconocimiento de voz entre otros muchos módulos.

En el capítulo 4 se explicó la implementación de este sistema de acceso a la plataforma mediante una aplicación web y dispositivos Android. Tras esto, se probó el funcionamiento del sistema, obteniendo unos resultados muy positivos en cuanto a la funcionalidad del sistema implementado, el cual se puede utilizar con otras plataformas de procesamiento en la nube similares a la de Vocalia.

Sin embargo, la calidad obtenida de las grabaciones no es del todo la esperada, pues en un futuro pueden aparecer errores en la transcripción de estos audios. En cuanto a la calidad obtenida de la audio-conferencia, como se comentó anteriormente, es muy difícil tener una conversación fluida con el resto, lo que supondrá cambiar en un futuro alguna implementación llevada a cabo en este trabajo con el objetivo de solucionar este problema.

Después de analizar los resultados en el apartado anterior, como última conclusión cabe decir que sería interesante diseñar un módulo de cancelador de ruido para añadirla al nodo de servicio empleado para la transcripción, con lo que se podría evitar los ruidos que se pueden producir durante la reunión.

6.2 Trabajo futuro

Dado los resultados anteriores, evidentemente el trabajo futuro reside fundamentalmente en la mejora de la calidad de las grabaciones y de la audio-conferencia. Para ello en este trabajo se proponen dos líneas de trabajo: la primera sería estudiar la implementación de las llamadas VoIP con otros paquetes software que existen en la actualidad, como puede ser Linphone o SipDroid, en vez de utilizar la librería SIP que implementa por defecto el entorno de Android Studio. La segunda línea de trabajo consistiría en hacer otro estudio en profundidad sobre la tecnología VoIP aplicada a tareas de reconocimiento y procesado de voz, pues sería interesante solucionar de forma automática esos defectos que pueden aparecer en la voz debido a la pérdida de paquetes o al jitter.

Otra línea de trabajo que se seguirá es la mejora de la interfaz de usuario, pues en este trabajo no se ha desarrollado demasiado esta parte, sobre todo en lo que concierne a la aplicación web. En cuanto a la aplicación Android, es perfectamente funcional y libre de fallos, aunque se informa de que su interfaz de usuario sólo está disponible para aquellos móviles que tengan un tamaño de pantalla determinado, por tanto se hace imprescindible su adaptación al resto de móviles Android, independientemente de la pantalla.

Referencias

[1] J.M. Pardo, X. Anguera, C. Wooters, "Speaker Diarization for Multi-microphone Meetings Using Only Between-Channel Differences", MLMI 2006, LNCS 4299, pp. 257 – 264, 2006.

[2] S.S. Chen, P.S. Gopalakrishnan, "Speaker, environment and channel change detection clustering via the Bayesian information criterion", "http://www.itl.nist.gov/iad/mig/publications/proceedings/darpa98/html/bn20/bn20.htm", 1998.

[3] M. Huijbregts, "Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled", "http://eprints.eemcs.utwente.nl/15023/01/thesis_Marijn_Huijbregts.pdf", CTIT Ph.D. thesis Series No. 08-123, 2008.

[4] "Transana Official Web Page", "http://www.transana.org/about/tour/index.htm".

[5] Transcriber, "Extensions to Transcriber for Meeting Recorder Transcription", "ww1.icsi.berkeley.edu/Speech/mr/channeltrans.html", 2004.

[6] Vocapia, "Vocapia Speech-to-text conversion", "http://www.vocapia.com/speech-to-text.html".

[7] Nuance, "Dragon Speech Recognition", "http://www.nuance.com/dragon/index.htm".

[8] A. Sangwan, Chiranth M.C., H.S. Jamadagni, R. Sah, R.V. Prasad y V. Gaurav ,"VAD Techniques for Real-Time Speech Transmission on the Internet", "http://homepage.tudelft.nl/w5p50/pdffiles/VAD%20Techniques%20for%20Real-Time%20Speech%20Transmission%20on%20the%20Internet.pdf".

[9] "Android Studio Overview", "http://developer.android.com/tools/studio/index.html".

Glosario

API	Application Programming Interface
SIP	Session Initiation Protocol
VoIP	Voice over IP
HTTP	HyperText Transfer Protocol
JSON	JavaScript Notation Object
HTML	HyperText Markup Language
ASR	Automatic Speech Recognition
OVV	Out-of-vocabulary
VAD	Voice Activity Detector
LVCSR	Large vocabulary Continuous Speech Recognition
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
ZCD	Zero Cross Detection
AED	Adaptive Energy Detector
WFD	Wave Fricative Detector
BIC	Bayesian Information Criterion
MFCC	Mel Frequency Cepstral Coefficients