ugr | Universidad de Granada

ESTUDIOS DE GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Desarrollo de una aplicación de Android basada en *crowdsourcing* para la recolección de datos de QoE y QoS sobre vídeos de YouTube

CURSO: 2013/2014

José Rafael Suárez-Varela Maciá

El tribunal constituido para la evaluación del trabajo fin de grado TFG titulado:

Desarrollo de una aplicación de Android basada en *crowdsourcing* para la recolección de datos de QoE y QoS sobre vídeos de YouTube

Realizado por el alumno: **José Rafael Suárez-Varela Maciá**
Y dirigido por el tutor: **Jorge Navarro Ortiz**

Ha resuelto asignarle la calificación de:

SOBRESALIENTE (9 - 10 puntos)

NOTABLE (7 - 8.9 puntos)

APROBADO (5 - 6.9 puntos)

SUSPENSO

Con la nota: puntos.

El Presidente:

El Secretario:

El Vocal:

Granada, a de de 2014

ugr | Universidad de Granada

ESTUDIOS DE GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Desarrollo de una aplicación de Android basada en *crowdsourcing* para la recolección de datos de QoE y QoS sobre vídeos de YouTube

REALIZADO POR:

José Rafael Suárez-Varela Maciá

DIRIGIDO POR:

Jorge Navarro Ortiz

DEPARTAMENTO:

Teoría de la Señal, Telemática y Comunicaciones

Granada, 12 de Septiembre de 2014.

Desarrollo de una aplicación de Android basada en *crowdsourcing* para la recolección de datos de QoE y QoS sobre vídeos de YouTube

José Rafael Suárez-Varela Maciá

PALABRAS CLAVE:

Streaming de vídeo, Calidad de Servicio (QoS - *Quality of Service*), Calidad de Experiencia (QoE - *Quality of Experience*), *crowdsourcing*, redes inalámbricas, Wi-Fi, redes móviles, Android, YouTube, base de datos, mySQL.

RESUMEN:

En un mundo como el actual cada vez toma más importancia en internet la compartición de contenidos multimedia. Uno de los servicios que supone un porcentaje muy destacable sobre el tráfico total de internet es el *streaming* de vídeo, cuyo indiscutible líder en la red actualmente es YouTube.

Asimismo el uso de dispositivos móviles en la red en la actualidad está tomando cada vez más protagonismo, especialmente en lo que respecta al *streaming* de vídeo. Según las estadísticas oficiales de YouTube más del veinticinco por ciento del tiempo total de reproducciones de vídeos de este servidor se debe a estos dispositivos.

Tomando en consideración estos aspectos, el Trabajo de Fin de Grado a desarrollar se ha basado en el estudio de la QoS (*Quality of Service*) y QoE (*Quality of Experience*) en YouTube sobre dispositivos móviles. El estudio consiste en obtener por un lado información sobre QoS tanto a nivel de la capa de red como la de aplicación al mismo tiempo que se llevan a cabo reproducciones de vídeos sobre dispositivos móviles en distintas condiciones. Por otro lado, se obtiene una valoración subjetiva del usuario sobre la calidad del vídeo al finalizar la reproducción de éste.

Con esta información se podrá realizar una estimación sobre en qué medida afectan distintos parámetros de QoS sobre la percepción (QoE) del usuario que visualiza el vídeo. Esto tendrá una gran repercusión puesto que se trata de un ámbito que aún no se encuentra muy explotado. En la actualidad sólo existen unos pocos estudios que relacionan la QoS y la QoE sobre el *streaming* de vídeo del servidor de YouTube. Sin embargo estos estudios se refieren a la reproducción

de vídeos desde ordenadores personales, de modo que no serán extensibles al *streaming* de vídeo sobre dispositivos móviles. Esto se debe a que se ha podido comprobar que la generación de tráfico por parte del servidor en este caso difiere respecto de la de los ordenadores personales. Por otro lado, se debe destacar que las expectativas del usuario no suelen ser las mismas al reproducir un vídeo desde un ordenador personal o desde un dispositivo móvil, ya que influyen factores como por ejemplo el tamaño de la pantalla.

Los datos recogidos también tendrán otras aplicaciones, como el trazado de mapas de cobertura Wi-Fi y de redes móviles o la obtención de trazas sobre el tráfico recibido para analizar el patrón de tráfico durante el *streaming* de vídeo.

Para la obtención de la información requerida se ha recurrido al desarrollo de una aplicación sobre el sistema operativo Android que permite monitorizar en segundo plano todos los parámetros de red y de aplicación deseados durante la reproducción de un vídeo. Finalmente se solicita al usuario mediante un formulario una opinión sobre la calidad del vídeo experimentada. Todos los datos se almacenan posteriormente en una base de datos centralizada a la que tiene acceso sólo el administrador del servidor.

Por último, cabe destacar que la aplicación se basa en la filosofía de *crowdsourcing*, lo que se traduce en la obtención de estadísticas con un coste mínimo sobre un gran potencial número de usuarios que disponen en la actualidad de un dispositivo móvil con Android y que podrán descargar de forma gratuita la aplicación.

Development of an Android application based on crowdsourcing for gathering data on QoE and QoS from YouTube's videos.

José Rafael Suárez-Varela Maciá

KEYWORDS:

Video streaming, Quality of Service (QoS), Quality of Experience (QoE), crowdsourcing, Wireless networks, Wi-Fi, mobile networks, Android, YouTube, Database, mySQL.

ABSTRACT:

In today's world, multimedia sharing is becoming increasingly important on the internet. One of the services that represents an outstanding share of the total internet traffic is video streaming, whose undeniable leader is YouTube. YouTube is a renowned service with worth mentioning figures. According to official reports from the company, every month around one million users visit its website and more than 6000 million hours of video are reproduced. With these statistics, YouTube can be considered a highly relevant internet service.

Likewise, the use of mobile devices is gaining paramount importance and, involving video streaming, more than 25% of total time of reproduction in YouTube happens through this kind of appliances. This means that everyday more than one million videos are watched from mobile devices.

Taking these statistics into consideration, our objective with this Degree Thesis is studying QoS (Quality of Service) and QoE (Quality of Experience) in YouTube services through mobile devices. For this purpose, on the one hand, we will aim at obtaining information about QoS, not only at the network layer but also at the application layer, by reproducing videos on mobile devices in different contexts and situations. Simultaneously, we will obtain an assessment of the quality perceived by the user after the reproduction of each video.

With this information, we will be able to estimate to which extend the different QoS parameters are affecting the quality of the videos reproduced as perceived by the user (QoE). This is an issue of vital importance on the field of internet services, as it has not been sufficiently explored yet. As far as we know, there are some studies that explore the relationship between QoS and QoE in video streaming focused on YouTube. However, most of them refer to the reproduction of videos on personal computers (PCs). It has been widely proved that the

generation of traffic from YouTube servers greatly differ for mobile devices compared to personal computers, which means that the conclusions of these studies may not be applicable to mobile devices. In addition, the expectations of a user when a video is being reproduced on a personal computer are not the same as when it is reproduced on a small display of a smartphone. These expectations may affect the user's opinion and, therefore, it is another reason to support the necessity of these new models.

The collected data will also serve to other purposes such as the elaboration of coverage maps for Wi-Fi and mobile networks, or obtaining packet traces that can be analyzed traffic patterns obtained during the video streaming.

In order to collect the necessary data, we developed an application for the Android operating system. This application allow us to perform a background monitoring task that provides network and application statistics during the reproduction of the video. At the end of the reproduction, the user is requested an opinion about the perceived quality of the video by means of a brief survey. All these data are afterwards gathered in a centralized database that can only be accessed by the server administrator.

Finally, it is worth mentioning that this application is based on crowdsourcing. This philosophy consists on the division of tedious tasks into multiple micro-tasks, which in this case are identified with the experiments carried out by the users of our application. In practical terms, crowdsourcing allows us to obtain statistics from a great number of potential users with a minimum cost.

D. Jorge Navarro Ortiz, profesor del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada, como director del Trabajo Fin de Grado de D. José Rafael Suárez-Varela Maciá

Informa de que el presente trabajo, titulado:

Desarrollo de una aplicación de Android basada en crowdsourcing para la recolección de datos de QoE y QoS sobre vídeos de YouTube.

Ha sido realizado y redactado por el mencionado alumno bajo mi dirección, y con esta fecha autorizo a su presentación.

Granada, a de de 2014

Fdo. Jorge Navarro Ortiz

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Grado se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a de de 2014

Fdo. Jorge Navarro Ortiz

Fdo. José Rafael Suárez-Varela Maciá

Agradecimientos

Ha sido un largo camino hasta llegar a este punto.

Quiero aprovechar estas líneas para agradecer a todas las personas que me han acompañado, en los buenos y en los malos momentos, a lo largo de estos 4 años de andadura universitaria y cuyo desenlace tiene lugar con este Trabajo de Fin de Grado.

A mis padres, a mi hermana y a Joaquina por su apoyo incondicional, por sus consejos, por su cariño, por sus valores y por ser todos ellos grandes ejemplos de perseverancia y animarme a seguir hacia delante.

A mis amigos, a los de toda la vida y a todos aquellos que he tenido el placer de conocer durante la carrera. Porque sin vosotros estos años no tendrían el significado que han tenido para mí. Gracias por vuestro apoyo y por los buenos momentos que hemos compartido y que seguiremos compartiendo.

Y, por supuesto, a mi tutor Jorge. Porque sin él no habría sido posible la realización de este trabajo. Me gustaría destacar su apoyo, porque siempre ha estado ahí cuando lo he necesitado. Además siempre me ha proporcionado buenos consejos y se ha preocupado por que se hicieran las cosas de la mejor forma posible.

Gracias a todos.

Índice general

Índice de figuras

Indice de tablas

Capítulo 1

Introducción y referencias bibliográficas

En este capítulo se va a realizar una introducción de todos los aspectos que rodean al presente proyecto. Se comenzará por la definición de conceptos clave de este proyecto como son la calidad de servicio (QoS - '*Quality of Service*') y la calidad de experiencia (QoE - '*Quality of Experience*'). Se expondrá seguidamente un apartado de motivaciones que tratará de plasmar la importancia de este proyecto y de todas las tecnologías y herramientas involucradas en él.

A continuación se incluirá un apartado de revisión del estado del arte en el que se recopilan trabajos de gran relevancia relacionados con lo que se pretende desarrollar en el presente proyecto.

Asimismo, una vez definidos todos los conceptos relativos al proyecto, se exponen de forma genérica los logros y aportaciones. Este apartado permitirá comprender el alcance que tendrá el presente proyecto. Al finalizar este apartado se detallarán de un modo resumido las principales fuentes bibliográficas consultadas en el proyecto.

Por último, se presentará a modo de esquema didáctico un apartado en el que se expondrá cómo se estructura el resto de la memoria, incluyendo una breve descripción de todos los bloques de los que se constituye ésta. Todo ello con la intención de orientar al lector a lo largo de todo el recorrido por la memoria, tratando así de mantener un hilo conductor a lo largo de ésta.

Una vez descrita la estructura de este capítulo, a modo de introducción y antes de comenzar con su desarrollo se plantea el objetivo final del presente proyecto. El fin último de éste consistirá en obtener una base de datos lo más completa posible con información acerca de la QoE y la QoS sobre *streaming* de vídeo del servidor YouTube. Con esta información posteriormente se podrán elaborar modelos que permitan predecir valores de QoE a partir de un conjunto de parámetros de QoS, ya sean a nivel de red o de aplicación, y viceversa. Además, en esta base de datos se pretenderá en todo momento incorporar información que pueda resultar de interés ya no sólo para la creación de estos modelos, sino también para la obtención de otro tipo de estadísticas.

1.1. Definición de los conceptos de QoS y QoE

Los conceptos de calidad de servicio (QoS) y calidad de experiencia (QoE) son dos términos de vital relevancia en el presente proyecto. Se trata de conceptos importantes en el ámbito de los servicios extremo a extremo en redes de datos. Uno de los servicios que tienen una especial relevancia es en la trasmisión de vídeo mediante *streaming*, lo que supone uno de los pilares en torno a los que se desarrolla el proyecto.

La calidad de servicio (QoS) se define como el efecto global de las características de servicio que determinan el grado de satisfacción de un usuario del servicio [1]. Se trata de un conjunto de parámetros que generalmente pueden ser tanto de nivel de red (ancho de banda, *jitter*, retardo...), como de nivel de aplicación (interrupción del servicio, resolución o imágenes por segundo) [2].

La utilización de tecnologías de monitorización de la QoS así como de mecanismos de ingeniería de tráfico, da lugar a la posibilidad de mejorar la experiencia del usuario sobre un servicio reduciendo el coste de una manera efectiva [3]. La ingeniería de tráfico permite priorizar el tráfico de aplicaciones concretas que son más sensibles al retardo, como puede ser el *streaming* de vídeo sobre redes TCP/IP.

Siguiendo el modelo OSI, se puede considerar una extensión de éste en la que por encima de la capa de aplicación, se encuentra una pseudo-capa relacionada con la experiencia del usuario final. En esta pseudo-capa se pueden realizar medidas de su rendimiento mediante el concepto de calidad de experiencia [4].

La calidad de experiencia (QoE) se define como la aceptación general de una aplicación o de un servicio conforme a la percepción subjetiva del usuario extremo [5]. Se trata de un término que en un principio puede aplicarse a cualquier servicio o negocio en el que exista un consumidor o usuario final. Asimismo se trata de una medición subjetiva que depende del usuario, de forma que se encontrarán usuarios que sean más fáciles de satisfacer que otros. La calidad de experiencia en el ámbito de las tecnologías de la información y la comunicación (TICs) no sólo se ve afectada por la QoS, sino que también existirán algunos efectos transversales tales como el coste, eficiencia, privacidad, seguridad o la interfaz [3].

En la actualidad, la transmisión de datos sobre aplicaciones sensibles al retardo, como el *streaming* de vídeo, es una tarea difícil sobre las redes públicas, que típicamente utilizan una planificación de tipo '*best effort*'. Para salvar estas dificultades se puede actuar sobre los parámetros de QoS de forma conveniente con el objetivo de conseguir un nivel deseado de QoE [6].

Las técnicas actuales que tratan de actuar sobre la QoE de los vídeos están basadas en la percepción espacial y temporal humanas. Estos dos tipos de percepción pueden controlarse a nivel de capa de aplicación, mediante la codificación del vídeo. Dentro de los tipos de codificación de vídeo se puede realizar una clasificación en tres grupos, la codificación '*intraframe*', '*interframe*' o técnicas de codificación entrópica. En lo que se refiere a la actuación a nivel de capa de red, se puede intervenir sobre la planificación y priorización de flujos así como en la gestión de la reserva de recursos. Para ello existen algunos protocolos que

actúan sobre estos aspectos, tales como el protocolo IntServ [7] ('*Integrated Services*'), DiffServ [8] ('*Differentiated Services*') o MPLS [9] ('*Multi-Protocol Label Switching*'). Este último está basado en la asociación de etiquetas a distintos flujos que determinarán el enrutamiento a lo largo de la red de los paquetes de forma individual, agilizando así la tarea de planificación y encaminamiento.

En el ámbito de las aplicaciones de vídeo, la métrica más común para la medición subjetiva de la calidad de experiencia es el MOS ('*Mean Opinion Score*'), estandarizado por la ITU-T [10]. Este medidor consiste en la valoración por parte del usuario de la calidad que ha percibido al finalizar la reproducción de un vídeo en una escala de 1 a 5, donde 1 es el peor nivel de calidad y el valor 5 corresponde a un nivel de calidad excelente.

Sin embargo, la medición subjetiva de QoE, mediante el MOS, presenta varios problemas que la hacen inefectiva para su utilización continua e individualizada sobre cada reproducción de vídeo. La utilización de este método implica grandes costes, un consumo de tiempo adicional que podría repercutir negativamente sobre el usuario y la imposibilidad de aplicarlo en tiempo real [11]. Por ello resulta de gran interés poder realizar predicciones de la QoE resultante a partir de un conjunto de parámetros de QoS y viceversa.

Con el fin de obtener aproximaciones que permitan la predicción de la QoE de forma objetiva, existen dos tipos de metodologías: los métodos intrusivos, que hacen uso de la señal de vídeo transmitida, y los métodos no intrusivos, que se basan en parámetros de los niveles de red y aplicación. Los métodos intrusivos son generalmente más precisos, sin embargo no permiten la monitorización del tráfico en tiempo real ya que necesitan la secuencia original del vídeo para poder analizar la calidad. Por otro lado los métodos no intrusivos no necesitan una copia del *stream* original y si permiten la implementación en tiempo real [11].

En el artículo [12] se llega a la afirmación de que el empeoramiento en un parámetro que genere deficiencias sobre la QoS, va en detrimento de la percepción de la calidad por parte del usuario. Así, en un modelo esquemático que recoja las relaciones entre QoE y QoS se plantea la distinción de tres zonas en función del nivel de QoS, tal y como se muestra Figura 1.1.1.

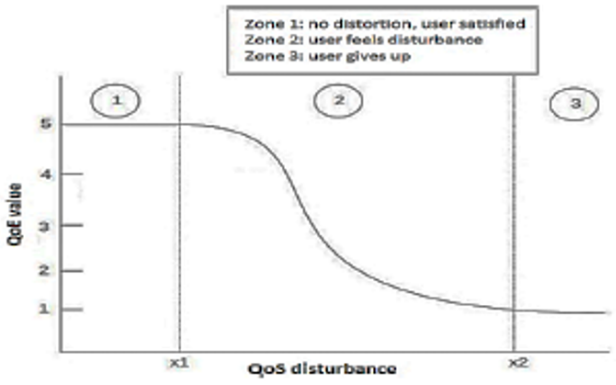

Figura 1.1.1: Curva de mapeo entre QoS y QoE

Cuando las deficiencias en la QoS son pequeñas y se sitúan en la zona 1, la percepción de la calidad por parte del usuario es alta y una variación de la QoS no afecta a la QoE siempre que se mantenga en dicha zona. Cuando la QoS se encuentra en la zona 2 se suele observar un decrecimiento considerable de la QoE conforme se van aumentando las deficiencias sobre la QoS. Por último, se alcanza la zona 3, en la que las deficiencias en la QoS son suficientes para que la valoración del usuario sea muy pobre y la QoE se estabilice en torno al valor mínimo conforme se vaya empeorando la QoS. En este supuesto el usuario se ve tan altamente afectado que el sistema de medición de QoE podría interrumpirse debido a las malas condiciones en la transmisión de datos.

Por último y con el fin de destacar el alcance que tienen los términos de QoS y QoE sobre el *streaming* de vídeo en la actualidad, cabe mencionar que en el ámbito de la investigación pueden encontrarse una gran cantidad de trabajos relacionados con este campo. Estos trabajos tienen como objetivo la consecución de modelos experimentales sobre *streaming* de vídeo que permitan relacionar QoS y QoE en distintas plataformas (ordenadores personales, dispositivos móviles...) y en distintas condiciones.

Entre ellos se encuentra, por ejemplo, el artículo [13] relacionado en especial con la temática del presente proyecto. En este estudio realizado por unos investigadores de la Universidad de Würzburg (Alemania) junto con investigadores del centro de investigación de telecomunicaciones de Viena (Austria), se obtiene un modelado de la QoE en función de distintos parámetros de QoS sobre servicios de vídeo *online* basados en *streaming* transportado sobre TCP. Estas características podrían aplicarse por ejemplo al servicio ampliamente conocido de YouTube. Entre los resultados expuestos, se presenta un modelo en el que se puede observar una curva que relaciona el MOS con el número de interrupciones detectadas a lo largo de la reproducción de un vídeo. Esas interrupciones serían eventos detectados a nivel de aplicación que típicamente se producen cuando hay un cuello de botella en la red por la que se realiza la transmisión de los datos. Además cabe destacar que el estudio sigue la metodología de *crowdsourcing* para la recopilación de los datos, lo que se traduce en un método rentable y flexible a la hora de realizar los experimentos. El método de *crowdsourcing* es un concepto que se considera de gran relevancia y será descrito en mayor detalle más adelante en este mismo capítulo.

1.2. Motivaciones

A continuación se van a suceder una serie de apartados a modo de motivación que van a describir distintos conceptos, tecnologías, plataformas, etcétera. Todos ellos se encuentran dentro del panorama que engloba el presente proyecto.

Estos conceptos son los pilares clave sobre los que se desarrolla el proyecto, así como los principales responsables de que el proyecto se encuentre dentro en un ámbito de gran relevancia en la actualidad dentro del campo de las tecnologías de la información y la comunicación (TICs). Como se podrá observar a lo largo de este apartado, todas las tecnologías y conceptos expuestos son de gran interés en el panorama actual además de tener potencialmente una gran progresión hacia un futuro próximo.

1.2.1. La importancia del *streaming* de vídeo y el servicio de YouTube

El servicio de *streaming* de vídeo es el que genera mayor cantidad de tráfico a lo largo de la red global así como uno de los servicios que están experimentando un amplio crecimiento y evolución en la actualidad.

Según [14] para el año 2017 habrá en Internet un tráfico de un millón de minutos de vídeo aproximadamente cada segundo. Según se expone en este mismo documento se necesitarían 5 millones de años para que una sola persona pueda ver el total de vídeos que atravesarán las redes IP a nivel global a lo largo de un mes. En total se alcanzará un volumen de tráfico de vídeo IP de 88.0 exabytes cada mes, frente a los 26.2 exabytes mensuales que suponía este tipo de tráfico en 2012. Además se estima que para este mismo año, el tráfico de vídeo para consumidores en Internet supondrá un 69 % del tráfico total generado por éstos. Esto supone un crecimiento considerable en el tráfico de vídeo respecto a los datos recogidos en 2012, donde el tráfico de vídeo suponía un 57 % respecto del total. Se debe destacar además que estas cifras no incluyen el volumen de datos de vídeo intercambiado a través de redes *peer-to-peer*, por lo que el tráfico total de vídeo a través de Internet se vería aún más incrementado si se tuvieran en cuenta estas cifras.

Una parte de este tráfico de vídeo se realiza a través de *streaming* sobre protocolos de transporte como TCP o UDP. El *streaming* de vídeo abarca todos aquellos servicios multimedia en los que existe uno o varios servidores con vídeos pregrabados y almacenados. Estos servidores se encargarán de servir sus vídeos a los clientes que los soliciten a través de una red de datos generalmente al mismo tiempo que éstos lo consumen.

Entre las páginas *web* de mayor relevancia que ofrecen un servicio de *streaming* de vídeo se encuentran grandes conocidas como YouTube, Hulu o Netflix, que ya en el año 2010 representaban el 40 % del tráfico total en Internet. En la actualidad, dentro del grupo de sitios *web* que ofrecen contenidos multimedia, YouTube es el líder en tanto que se trata del tercer sitio más visitado a nivel global y el primero dentro de los que ofrecen servicios de audio y/o vídeo [15]. Como información adicional que corrobora la relevancia de este servicio a nivel global, se indica en sus estadísticas oficiales [16] que actualmente (7 de Junio de 2014) más de mil millones de usuarios únicos visitan este sitio cada mes, se reproducen en total más de 6 mil millones de horas de vídeo al mes y cada minuto se suben en torno a 100 horas de vídeo a los servidores de YouTube. Además también en [16] se puede leer que el 40 % del tiempo total de reproducción de vídeos procedentes de este servidor se debe a dispositivos móviles. Esto último indica que este servicio se está introduciendo con gran fuerza en el mercado de los dispositivos móviles, que como es bien sabido se trata de un sector que está experimentando un gran crecimiento en la actualidad [17].

El éxito de YouTube se atribuye al carácter de red social que tiene en combinación con la gran oferta de vídeos que se pueden reproducir [18]. En YouTube los usuarios no sólo consumen vídeos, sino que también suben y comparten sus propios contenidos.

En el ámbito del vídeo sobre *streaming*, los servicios tradicionales se basan en el protocolo de transporte UDP, donde una posible congestión en la red puede

dar lugar a la pérdida de paquetes. Esta pérdida de paquetes a su vez generará efectos indeseables tales como saltos en la secuencia o la visualización de objetos degradados. Sin embargo, YouTube realiza la entrega de vídeos a partir de un sistema de descarga continuo que utiliza TCP como protocolo de transporte. Esta técnica permitirá la reproducción del vídeo original sin ninguna alteración, ya que TCP se encargará de la retransmisión de paquetes corruptos o que no han alcanzado el destino [13]. Además, este método permite que el vídeo pueda reproducirse en el extremo final sin necesidad de esperar hasta que se realice la descarga completa del vídeo.

Todo ello hace de YouTube un servicio muy interesante de analizar desde diferentes perspectivas tales como la caracterización de su tráfico [19, 20], el estudio de su infraestructura de red [21], la evaluación de la QoE con diferentes condiciones de QoS [13, 22], obtención de estadísticas sobre los tópicos de los vídeos reproducidos y/o el número de visualizaciones de éstos [18], etcétera.

Por último cabe destacar que los resultados de cualquier estudio realizado sobre estadísticas de comportamiento de usuarios, número de reproducciones de vídeos o tópicos de vídeos en YouTube podrán extenderse a otros servicios proporcionados por otros servidores de vídeo sobre *streaming*. Si bien, debido al carácter de red social de YouTube, sólo se podrán aplicar a servicios que tengan una naturaleza parecida (por ejemplo Netflix), ya que los servidores de *streaming* convencionales tienen un comportamiento diferente [18].

1.2.2. Actualidad y futuro de las redes móviles y conexiones WLAN (Wi-Fi)

En este apartado se va destacar la importancia de las redes inalámbricas en la actualidad. Dentro de este tipo de redes destacan las redes móviles, que utilizan tecnologías tales como GSM, UMTS o LTE, y redes con puntos de acceso que utilizan la tecnología IEEE 802.11, a menudo denominadas indistintamente como redes con conectividad Wi-Fi.

En el estudio [23] se puede leer que el volumen de tráfico generado en redes fijas con tecnología Wi-Fi supuso un 49 % del tráfico total de Internet en 2012, mientras que en 2017 se prevé que el tráfico total en este tipo de redes alcance un volumen de un 56 % del tráfico total de Internet.

Asimismo, también en [14] se muestran unas estadísticas en las que se realizan unas previsiones para 2017 que indican que el tráfico de redes móviles supondrá un 12 % del tráfico total de Internet, frente al 3 % que suponían en el año 2012. En términos de volumen de datos se prevé que estas redes alcancen un total de 11.2 exabytes mensualmente en el año 2017, frente a 885 petabytes que generaban en el año 2012. Además se indica un dato muy significativo acerca de su evolución futura, y es que en el periodo de 2012 a 2017 el volumen de tráfico en redes móviles crecerá 3 veces más rápido que el tráfico en redes fijas.

Según reza este mismo estudio estadístico, al mismo tiempo las redes fijas cableadas se verán desfavorecidas en lo que a volumen de datos se refiere. Para el año 2017 se prevé que el tráfico debido a este tipo de redes experimente un decremento de 16 puntos porcentuales, de un 48 % a un 32 % del tráfico total de Internet.

Con todos los datos expuestos hasta el momento, parece evidente que la tendencia en los próximos años es de un rápido crecimiento de las redes inalámbricas, ya sean tecnologías IEEE 802.11 (Wi-Fi) o de redes móviles (GSM, UMTS, LTE y tecnologías futuras), en detrimento de las redes fijas cableadas. Para el año 2017, si se suma el tráfico de redes móviles en combinación con el de redes con tecnología Wi-Fi, se alcanzará un total del 68 % del tráfico total de Internet.

En [17] además se pueden observar datos curiosos como que para el año 2018 habrá un total de 177 millones de dispositivos *wearables* conectados en todo el mundo. Esto supone que estos dispositivos, que típicamente se conectan mediante tecnologías inalámbricas, se verán incrementados en un factor multiplicativo de 8 respecto del número de este tipo de dispositivos que se estima que había en 2013. Esto último refuerza la teoría de la importancia creciente que tendrán las redes inalámbricas en los próximos años.

Existen otros múltiples estudios que apoyan el fuerte crecimiento tanto de las redes móviles como de las redes con puntos de acceso Wi-Fi. Entre ellos, se tienen por ejemplo las gráficas de la Figura 1.2.1 en las que se muestra la evolución prevista desde 2009 a 2015 del número total de puntos de accesos inalámbricos públicos [24] y privados [25] que existirán a lo largo del mundo. Como se puede observar, el número de puntos de acceso públicos (a la izquierda), que en la actualidad es considerablemente menor que el número de puntos de acceso privados (a la derecha), experimentará en el próximo año un crecimiento muy acusado. Así, entre los años 2013 y 2015 se prevé un crecimiento de un 75 % en los puntos de acceso públicos respecto de la cifra de 2013, mientras que en el ámbito de los puntos de acceso privados se prevé un crecimiento de un 31 % en este mismo periodo.

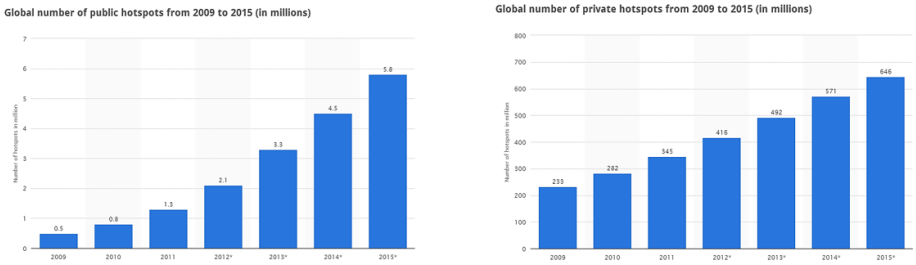

Figura 1.2.1: Evolución de los *hotspots* en el periodo de 2009-2015

Esto permitirá que las redes cuyo acceso se realiza a través de *hotspots* públicos puedan trabajar en colaboración con las redes móviles para soportar la creciente carga de tráfico de dispositivos móviles tales como los *smartphones*, que típicamente tienen soporte para conectarse a ambos tipos de redes. Este efecto se ve reflejado en [17], donde se estima que para el año 2018 el 52 % del tráfico generado por dispositivos móviles se encaminará hacia la red fija mediante conexiones Wi-Fi, frente a un 48 % que lo hará a través de las redes móviles. En la gráfica de la Figura 1.2.2 se puede observar la evolución de ambas tasas de tráfico en los próximos años.

Figura 1.2.2: Evolución del tráfico de dispositivos móviles en redes fijas (*offload*) y móviles

Por último se presenta una gráfica en la Figura 1.2.3 tomada de [26] en la que se demuestra el amplio crecimiento que van a experimentar las tecnologías inalámbricas hasta el año 2020, ya no sólo en cuanto a volumen de tráfico en Internet como demuestran las estadísticas anteriores, sino también en cuanto al número de usuarios que utilizarán dispositivos móviles.

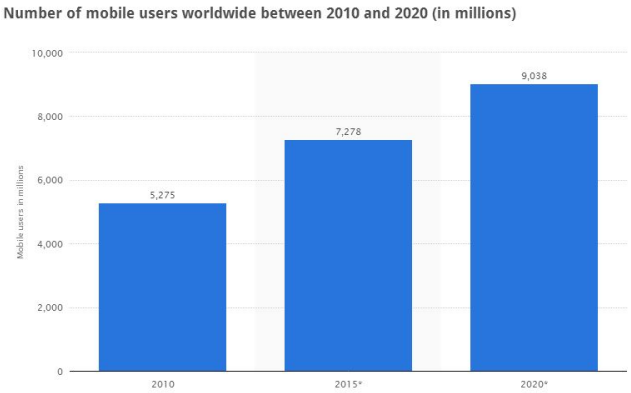

Figura 1.2.3: Evolución del número de usuarios con dispositivos móviles entre 2010 y 2020

Como se puede observar se estima un crecimiento de un 24 % para 2020 respecto de la cifra estimada para 2015. Esto supone que habrá en total unos 9.038 millones de dispositivos móviles en todo el mundo para el año 2020.

1.2.3. ¿Por qué elegir Android?

El sistema operativo Android es uno de los sistemas operativos líderes dentro del mercado de los dispositivos móviles. En concreto se trata del que cuenta en la actualidad con un mayor número de usuarios.

En [27] se puede observar una estadística con el número de activaciones de dispositivos Android diariamente entre Agosto de 2010 y Abril de 2013. Como

se puede observar en la Figura 1.2.4, en el mes de agosto de 2010 se realizaban de media en torno a 200.000 activaciones diariamente, mientras que en Abril de 2013 ya se realizaban en torno a 1.500.000 activaciones diarias. Observando estas estadísticas se puede afirmar que en la actualidad el número de usuarios de Android está experimentando un crecimiento con carácter exponencial.

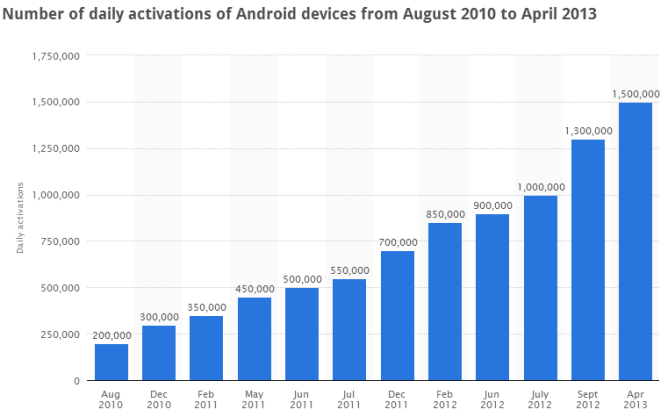

Figura 1.2.4: Activaciones diarias de Android entre Agosto de 2010 y Abril de 2013

Además, si se compara el número de usuarios totales de cada uno de los principales sistemas operativos para dispositivos móviles [28], se tiene, en la Figura 1.2.5, que en los últimos 4 años Android ha experimentado un abultado crecimiento que lo ha encumbrado a la cabeza hasta alcanzar en el último cuatrimestre de 2013 aproximadamente el 80 % del total de los dispositivos móviles. Esto supone una gran ventaja frente a su competidor inmediato (IOS), que abarca aproximadamente un 20 % de los dispositivos móviles del mercado en esa misma fecha.

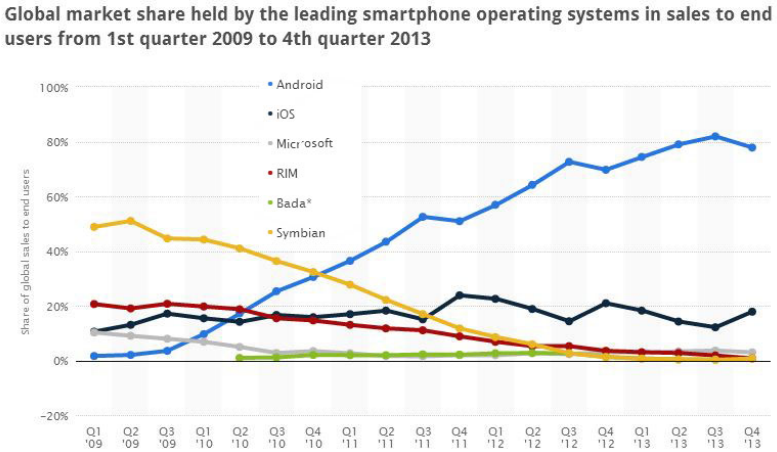

Figura 1.2.5: Porcentaje de ventas de los sistemas operativos líderes en el mercado de los *smartphones* desde 2009 a 2013

Observando todos estos datos parece lógico pensar que seguirá siendo el líder del mercado durante un tiempo.

El éxito de Android puede atribuirse a una combinación de características que lo hacen diferente y lo han convertido en un sistema operativo de gran interés para el consumidor. Entre estas características se destacan las siguientes:

- Se trata de una plataforma de código abierto para el desarrollo y la ejecución de aplicaciones. Esta plataforma está basada en el sistema operativo Linux y fue desarrollada por la '*Open Handset Alliance*', una alianza comercial de 84 compañías liderada por Google. Al ser de código abierto permite la personalización del sistema y su posterior utilización sin necesidad de pagar por los derechos de autor.
- Es independiente de las características *hardware* del dispositivo sobre el que se utiliza. Esto lo hace muy versátil ya que se adapta a una amplia gama de dispositivos, ya sean terminales móviles, tabletas, dispositivos *wearables* o *Smart TVs*.
- Es un sistema operativo especialmente diseñado para dispositivos con recursos limitados tales como el consumo de potencia, la capacidad de procesamiento, la capacidad de memoria, la movilidad o la conectividad. Para ello hace uso de una máquina virtual Dalvik que fue desarrollada con el fin de mejorar el rendimiento y reducir el consumo y que permite ejecutar aplicaciones desarrolladas en Java.
- Las aplicaciones desarrolladas en Java son fácilmente portables a otros sistemas. Esto convierte a Android en un sistema operativo atractivo para el desarrollador, no sólo por la portabilidad en sí, sino también por la facilidad de aprendizaje para aquellos desarrolladores que conocen el lenguaje Java. Además existen entornos de programación específicos para Android como *Eclipse* o *Android Studio* que, junto con las librerías de Android SDK (*Software Development Kit*), facilitan en gran medida las labores de programación.
- Incorpora una amplia gama de servicios como la localización mediante GPS o a través de triangulación con celdas móviles o Wi-Fi, soporte para gráficos y audio de alta calidad y mecanismos de síntesis y reconocimiento de voz bastante robustos.
- La interfaz gráfica de Android está basada en componentes inspirados en Internet, de forma que la descripción mediante XML permitirá que una misma aplicación pueda ejecutarse en un *smartphone*, una tableta o un PC.
- Ofrece mecanismos de seguridad para actuar ante el *malware*, de forma que por ejemplo las aplicaciones permanecen aisladas entre sí mediante el concepto de ejecución dentro de una caja que ya implementa su antecesor Linux. Además gestiona los permisos de cada aplicación de forma que el usuario antes de instalar una aplicación podrá conocer las posibles acciones que podrá llevar a cabo ésta. Por ejemplo, para tener acceso a Internet, a la localización del dispositivo o al almacenamiento interno del dispositivo, el usuario debe aceptar estos permisos de forma explícita.

1.2.4. El modelo de *crowdsourcing*

El término de *crowdsourcing* es un concepto que se ha ido forjando a lo largo de muchos años y cuya definición ha ido variando a lo largo del tiempo. Debido a esto se tratará de realizar un recorrido a lo largo de la historia para analizar las definiciones que se han ido ofreciendo desde que el término fue acuñado por primera vez hasta el momento, tratando finalmente de contextualizarlo en el ámbito en el que se va a explotar en el presente proyecto.

La etimología de la palabra *crowdsourcing* proviene de la composición de las palabras en inglés *crowd* (multitud) y *outsourcing* (externalización), por lo que una definición sencilla podría ser 'externalización de tareas a un grupo de personas'. Sin embargo esta definición quizás no hace total justicia al término en tanto que se trata de una descripción bastante genérica.

El primer autor conocido que se refiere al término de *crowdsourcing* fue Jeff Howe, a menudo denominado el padre del *crowdsourcing*. En una de sus publicaciones [29] este autor expone la siguiente definición:

> "*El crowdsourcing es la externalización, por parte de una empresa o institución, de una función realizada por un empleado a un grupo indefinido (y normalmente grande) de personas mediante una convocatoria abierta. Esta externalización puede tomar la forma de una producción-de-iguales (peer-production) cuando el trabajo se realiza de forma colaborativa, pero también puede llevarse a cabo de forma individual.*"

En esta definición ya se añade un nuevo matiz, y es que el grupo de personas que desempeñarán la función que se pretende externalizar se convoca en forma de llamada abierta. De este modo, cualquier individuo tendrá acceso a dicha convocatoria. Asimismo, la realización de la tarea puede realizarse tanto de forma individual como de forma colaborativa.

Con esta definición ya se puede alcanzar una idea bastante exacta del significado de este término. No obstante, posteriormente este concepto ha sido utilizado en múltiples ocasiones en campos de la ciencia bastante diferentes y esto ha dado lugar a que se hayan realizado multitud de definiciones que en algunos casos son difíciles de poner en común para alcanzar una definición única del término. Para ello, con el fin de poner fin a la ambigüedad de este término, dos investigadores de la Universidad Politécnica de Valencia presentaron un artículo [30] en el que hicieron una revisión bibliográfica en profundidad para posteriormente alcanzar como conclusión una definición universal de *crowdsourcing*. En esta revisión se encontraron en total 40 definiciones diferentes de dicha palabra expuestas por diferentes autores y, tras un análisis detallado de éstas y de todos los conceptos que rodean al término en cuestión, presentan la siguiente definición:

> *"El crowdsourcing es una actividad participativa online en la que un individuo, institución, organización sin ánimo de lucro o empresa propone a un grupo de individuos de conocimiento, heterogeneidad y número variable, la realización voluntaria de una tarea a través de una convocatorio abierta flexible. La realización de esta tarea, de complejidad y modularidad variable, y en la que la multitud debe participar aportando su trabajo, dinero, conocimiento y/o experiencia, siempre implica un beneficio mutuo. El usuario recibirá la satisfacción de una necesidad, sea esta económica, de reconocimiento social, de autoestima o de desarrollo de capacidades personales, mientras que el crowdsourcer obtendrá y utilizará en su beneficio la aportación del usuario, cuya forma dependerá del tipo de actividad realizada."*

En esta definición ya se puede observar la evolución que había ido sufriendo a lo largo de los años este concepto. Ahora se trata de una actividad *online* en la que se propone a un grupo heterogéneo de personas la realización de una tarea de forma voluntaria. Además se indica explícitamente que supone un beneficio mutuo de las dos partes, ya sea meramente económico o de satisfacción personal o cualquier otro tipo.

Con ello ya se ha realizado una aproximación bastante cercana al término de *crowdsourcing* que se pretende reflejar en el presente proyecto y que será mencionado frecuentemente a lo largo de la memoria, puesto que se trata de uno de los pilares sobre los que se sustenta el proyecto.

En definitiva, en el presente proyecto se pretende aplicar el concepto de *crowdsourcing* para la obtención de datos de forma masiva, sobre un público heterogéneo que realice encuestas de forma voluntaria y anónima. Así, se desarrollará una aplicación para dispositivos móviles que actuará como plataforma de recolección de datos mientras se está llevando a cabo la reproducción de un vídeo y posteriormente invitará al usuario a completar una encuesta que será enviada a un servidor de base de datos centralizado.

De este modo se está utilizando una metodología económicamente rentable para la obtención de datos de una forma potencialmente masiva. Esto, unido a que la plataforma se desarrolla sobre dispositivos Android, asegurará que la aplicación pueda llegar potencialmente a un gran número de usuarios que además conformarán una muestra bastante heterogénea.

La utilización de esta técnica es cada vez más frecuente en el ámbito de los estudios de QoE y QoS sobre *streaming* de vídeo y existen algunos estudios que la avalan. Por ejemplo, en [13] se realiza un estudio de QoE sobre servicios de vídeo *online* basados en *streaming* utilizando TCP en la capa de transporte. Para la obtención de la base de datos utilizada se recurre a una plataforma mediadora que aplica la filosofía de *crowdsourcing* y que convoca en forma de llamada abierta a todos los usuarios registrados que quieran participar. En este estudio en concreto se invita al usuario a visualizar vídeos sobre PCs mientras se monitorizan una serie de parámetros de QoS y posteriormente se realiza una encuesta a éste con el fin de obtener posibles relaciones entre parámetros de QoS y QoE. La plataforma de *crowdsourcing* utilizada se denomina 'microworkers' y contaba ya con 800.000 usuarios a finales de 2010. En este caso, aquellos que participaban en cada una de las pequeñas tareas propuestas recibían una pequeña compensación económica a cambio. En el artículo mencionado se describe

textualmente la técnica utilizada como "altamente rentable, rápida y una forma flexible de realizar experimentos".

1.3. Revisión del estado del arte

El fin último de este proyecto consistirá en obtener una base de datos importante que posteriormente permita la obtención de estadísticas para predecir valores de QoE a partir de ciertos parámetros de QoS, ya sean a nivel de red o de aplicación, y viceversa.

Por ello, en este apartado se tratará de ofrecer una perspectiva global del estado del arte de todas las tecnologías y conceptos que se engloban en el proyecto.

Con este fin, en este apartado se realizará una recopilación de artículos que tratan sobre estudios de calidad de servicio y calidad de experiencia así como de obtención de estadísticas e inferencias sobre relaciones entre parámetros de estas dos. Para ello se analizarán distintos escenarios que como se podrá ver se desarrollarán sobre diferentes paradigmas de programación (Flash, Java...) así como diferentes plataformas (dispositivos móviles y ordenadores personales).

A continuación se realizará una recopilación de todas las aplicaciones basadas en el sistema operativo Android existentes en la actualidad relacionadas con la monitorización y recolección de distintos datos basándose en la filosofía de *crowdsourcing*. Se podrá observar que todas estas aplicaciones abarcan una amplia gama de datos recopilados, desde niveles de señal en redes móviles (relacionado con la QoS) hasta datos de consumo de batería o datos meteorológicos. Todas estas aplicaciones irán acompañadas de una descripción general que destacará sus principales funcionalidades así como las principales aportaciones que pueden tener para el proyecto.

1.3.1. Estudios experimentales basados en mediciones de QoS y QoE

En primer lugar se presentan dos estudios que han emergido de las Universidades de Roma (Italia) y Hong Kong (China) en los que, mediante distintas metodologías, se han obtenido mediciones de QoS y QoE sobre la visualización de múltiples vídeos con diferentes parámetros de calidad con el objetivo de estudiar las relaciones entre ambos factores.

Estos estudios tienen en común que analizan la problemática asociada que tiene el *streaming* de vídeo debido a deficiencias en la red (interrupciones, pérdida de paquetes, *jitter*...). El primero de ellos se constituye en torno al ámbito de los dispositivos móviles con una particularización de aquellos que utilizan el sistema operativo Android. Por otro lado, el segundo de ellos se realiza sobre ordenadores personales en los que el *streaming* de vídeo se realiza mediante HTTP y la visualización se hace mediante una aplicación Flash.

"*Modelling quality of Experience in future internet networks*"

En este estudio [31] de un grupo de investigadores de la Universidad de Roma (Italia) se presenta un modelo en el que se trata de analizar las desviaciones que se producen en la visualización de un vídeo entre lo que se define como QoE activo y QoE pasivo. El último de estos términos se refiere a la parametrización de la QoE a partir de una serie de parámetros de QoS, donde la función que los relaciona se toma de estudios realizados con anterioridad. Entre estos parámetros de QoS que afectan a la QoE se encuentran el *jitter*, el retardo, el ancho de banda, la pérdida de paquetes o el BER (*Bit Error Rate*).

Por otro lado se describe la QoE activa como la realimentación aportada por un usuario que visualiza el vídeo, es decir, una valoración subjetiva asociada al comportamiento humano que se define normalmente como no lineal, no estacionario y que puede ser modelado por un proceso estocástico.

Para la realización de este estudio se desarrolló una API con funciones que permitían medir la QoE activa (aportada por el usuario) en función de dos posibles parámetros: el MOS (*Mean Opinion Score*) y 'OneClick'.

Mean Opinion Score no es más que una valoración entre 1 (pobre) y 5 (excelente) de la calidad de experiencia [10]. Mientras que 'OneClick' es un método descrito en [32] que consiste en recoger la calidad de experiencia del usuario a partir del número de *clicks* que éste realiza sobre un botón que se le pone a su disposición. De este modo, a mayor número de *clicks* y con mayor frecuencia el usuario estará indicando una mala percepción del vídeo y por lo tanto habrá un decremento en la QoE.

Una vez desarrollada esta API, los autores realizaron una aplicación para el sistema operativo Android desde la que se podían visualizar videos en *streaming* descargados de un servidor '*Darwin Streaming Server*' (DSS). Esta aplicación contiene una funcionalidad mediante la cual el usuario podrá hacer *click* sobre un botón cuando perciba que la calidad no es adecuada y podrá ir observando un emoticono que irá variando desde una cara muy feliz (calidad excelente) a una cara de enfado (calidad pobre) en función del número de *clicks* totales y la frecuencia con la que pulse el botón. Con esto se pretende obtener la QoE activa del usuario que se enviará de forma periódica al servidor de *streaming* para que este pueda almacenar la información. Estos datos se compararán con los obtenidos de la QoE pasiva calculada a partir de una serie de parámetros de QoS que la aplicación irá monitorizando a la vez que se está realizando el *streaming*.

En la Figura 1.3.1 se puede observar la interfaz gráfica de dicha aplicación. En la imagen de la izquierda se tiene un ejemplo en el que se puede observar una buena calidad sobre el vídeo y el usuario así lo ha percibido y consecuentemente no ha efectuado ningún *click*. Sin embargo, a la derecha se puede observar un caso en el que la visualización del vídeo es de una calidad cuestionable y el usuario ha efectuado un total de 27 *clicks* que demuestran su descontento con la calidad percibida. Asimismo, se puede observar que hay un emoticono que expresa el nivel de QoE percibido por el usuario en ambos casos.

Cabe destacar que esta aplicación desarrollada para Android ha sido utilizada para la obtención de información para el estudio presentado en el artículo y como

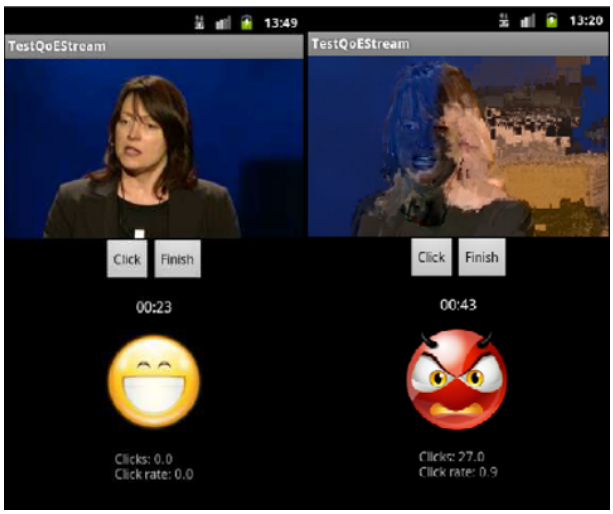

Figura 1.3.1: Aplicación desarrollada en el artículo "*Modelling quality of experience in future internet networks*"

ejemplo de aplicación de la API que se había desarrollado. No obstante, esta aplicación no está disponible para su descarga de forma que cualquier individuo pueda hacer uso de ésta para medir la calidad de experiencia activa sobre su propio servidor de *streaming*.

"*Measuring the quality of experience off HTTP video streaming*"

En el estudio [22] de un grupo de investigadores de la Universidad Politécnica de Hong Kong (China) se describe un método para medir la QoE sobre *streaming* de vídeos sobre HTTP. En dicho estudio se pretende obtener relaciones entre múltiples parámetros de QoS y QoE sobre la visualización de vídeos con diferentes niveles de calidad. Para ello, se desarrolla una plataforma en Flash en la que se llevan a cabo reproducciones de vídeos previamente seleccionados con diferentes niveles de QoS. Estos vídeos tendrán una duración de 20 segundos y al final de la reproducción de cada uno de ellos se le demandará al usuario final que haga una valoración entre 1 y 5 (escala MOS) sobre la calidad percibida.

Finalmente se presenta un estudio estadístico en el que se pueden observar relaciones entre parámetros como el MOS y la tasa de pérdidas de paquetes. Este estudio se realiza únicamente sobre 10 individuos a los que se les presentan 27 muestras de distintas calidades con el fin de obtener una base de datos que incluya estadísticas de vídeos de diversas calidades. Por ello se puede considerar que se trata de un estudio de laboratorio más que un modelo de *crowdsourcing* ya que las características intrínsecas de este último modelo se basan en una muestra de individuos amplia y heterogénea.

Como nota adicional se indica que esta aplicación en Flash que se utiliza como herramienta para la realización de encuestas y la recopilación de datos no se ha desplegado públicamente, por lo que no puede ser utilizada para que quien lo desee pueda realizar posteriores estudios con dicha plataforma.

1.3.2. Trabajos basados en *crowdsourcing* y mediciones de QoE

En este apartado se listan de forma genérica algunas aplicaciones que realizan labores de monitorización de parámetros, cada una de ellas de una naturaleza distinta. Este tipo de aplicaciones tienen además, como característica común, que se utilizan como plataformas de recolección de datos en dispositivos móviles sobre el sistema operativo Android. Como principal aportación de estas aplicaciones al proyecto, se destaca el hecho de que todas ellas siguen la filosofía de *crowdsourcing* para la recolección de datos, lo que se traduce en que se trata de obtener una base de datos potente en cuanto al número y heterogeneidad de individuos de la muestra con un bajo coste que además es prácticamente fijo. Es decir, una vez que se realiza una inversión inicial para disponer de la plataforma de recolección de datos y de un servidor de base de datos (cuyo coste de mantenimiento es muy bajo) que almacene estos datos para su posterior posprocesado, la cantidad de observaciones distintas de las que dispone la muestra no influirá significativamente en el coste final.

No obstante, sí hay que tener en cuenta que, en términos de almacenamiento de datos en el lado del servidor y de ancho de banda de conexión en sentido ascendente (hacia éste), se puede producir un cuello de botella. En este caso habrá que asumir algunos costes adicionales que sean acordes con las dimensiones y la escalabilidad de dicho proyecto.

Sin embargo, la técnica de *crowdsourcing* se utiliza con gran recurrencia en la actualidad debido a que suele resultar de gran interés en lo que se refiere a rentabilidad en la obtención de bases de datos que sean de dimensiones notables. Como ejemplo de la rentabilidad del modelo de *crowdsourcing*, en el estudio [13] se puede observar que, una vez descartadas todas las muestras que no se consideran fiables, el coste final de cada una de las encuestas utilizadas para el análisis se encuentra por debajo de 1 dólar americano.

A continuación se exponen las aplicaciones de interés relacionadas con el presente proyecto que se han encontrado sobre dispositivos móviles de la plataforma Android hasta la fecha.

Portolan Network Tools

La aplicación *'Portolan Network Tools'* [33, 34] forma parte de un proyecto de investigación desempeñado por un grupo de investigadores de la Universidad de Pisa (Italia) en conjunto con algunos trabajadores del Instituto Italiano de Tecnología (IIT). Este proyecto tiene como fin descubrir por un lado la topología y estructura de Internet a escala global y por otro lado obtener mapas de cobertura de redes móviles realizando continuamente medidas que incluyen el nivel de señal (RSSI - *Received Signal Strength Indication*) y el proveedor de servicios que ofrece dicha cobertura.

Para ello distribuyen esta aplicación disponible para Android que se ejecuta en segundo plano y monitoriza continuamente parámetros de tráfico de red sobre Internet y realiza mediciones de la señal de la red móvil para generar un mapa de cobertura. Estas últimas mediciones se realizan en conjunción con la localización

geográfica del dispositivo móvil y con el operador móvil para así poder ubicar esta medición y generar un mapa en el que se puedan filtrar las medidas en función del proveedor.

Además, según reza la introducción que se puede encontrar en [33], el consumo de batería debido a esta monitorización continua es prácticamente nulo y el consumo de datos para realizar las mediciones sobre tráfico de datos es inferior a 2 MB por día. Asimismo, los datos enviados carecen de cualquier identificador, por lo que son totalmente anónimos para preservar la privacidad del usuario.

En las imágenes de la Figura 1.3.2 se puede observar la interfaz gráfica de dicha aplicación.

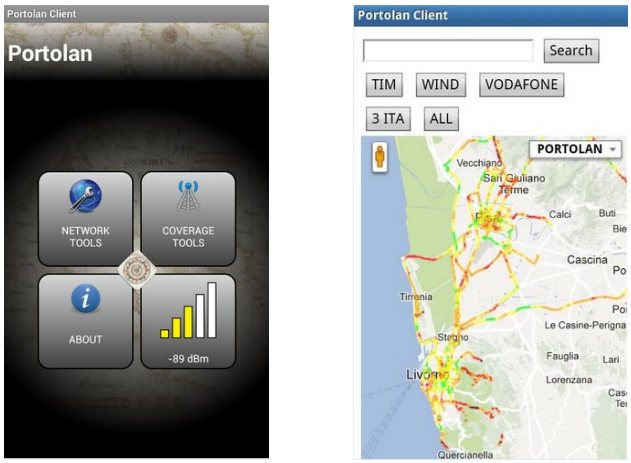

Figura 1.3.2: Aplicación *'Portolan network tools'*

Por último cabe a destacar que, como incentivo para la descarga y utilización de esta aplicación a parte de colaborar con dicho proyecto de investigación, se ofrecen algunas herramientas de red útiles y ampliamente conocidas en el ámbito de las redes de comunicación sobre TCP/IP. Algunas de estas herramientas son *'ping'*, *'traceroute'*, estimación del *throughput* máximo con la conexión actual, etcétera.

En definitiva, lo más relevante en relación con el presente proyecto es que se trata de un sistema basado en el modelo de *crowdsourcing*, ya que se realiza el proceso de recolección de datos en el que cada uno de los usuarios finales realizan una pequeña aportación ofreciendo sus estadísticas de monitorización pero potencialmente pueden obtenerse una gran cantidad de datos a escala global y sobre una muestra heterogénea con un coste mínimo.

Device Analyzer

La aplicación *'Device Analyzer'* [35, 36] forma parte de un proyecto de investigación de la Universidad de Cambridge (Reino Unido). Esta aplicación se ha desarrollado para el sistema operativo Android y se ejecuta en segundo plano enviando periódicamente datos a un servidor localizado en dicha universidad.

La información enviada es de carácter anónimo y se utiliza para realizar posteriormente inferencias estadísticas sobre todos los datos recolectados.

Según se puede leer en [35], los datos recolectados se dirigen principalmente a estadísticas de uso del dispositivo móvil por parte del usuario tales como las aplicaciones que se utilizan y con qué frecuencia, el número de mensajes cortos enviados, el número de llamadas perdidas, estadísticas de duración de llamadas entrantes y salientes, estadísticas de tráfico de datos entrante y saliente, etcétera. En definitiva, una serie de datos que permiten conocer los patrones de uso de una gama muy amplia de usuarios a lo largo del mundo.

En la imagen de la Figura 1.3.3 se puede observar la interfaz gráfica de dicha aplicación.

Figura 1.3.3: Aplicación *'Device analyzer'*

Como incentivo para que el usuario final se descargue la aplicación y haga uso de ella, ésta permite que el propio usuario pueda ver todos los datos que han sido recolectados de modo que podrá conocer una gran cantidad de estadísticas del uso que ha realizado sobre su propio dispositivo móvil.

Se trata de una aplicación que, al igual que la anterior, se basa en la filosofía de *crowdsourcing*. En este caso se trata de un caso en el que la utilización de *crowdsourcing* supone un gran acierto debido a las dimensiones del proyecto y al gran potencial de usuarios con el que cuenta una universidad de gran reconocimiento a nivel mundial. Actualmente, según [35] cuenta con un total de 17.906 usuarios (28 de Abril de 2014) que contribuyen al proyecto a lo largo de todo el mundo.

Carat

La aplicación *'Carat'* [37, 38] es parte de un proyecto de un grupo de investigadores de la Universidad de Berkeley (California, EEUU) con la colaboración de algunos investigadores de la Universidad de Helsinki (Finlandia). Esta aplicación ha sido desarrollada para los sistemas operativos Android e IOS.

El objetivo de esta aplicación es recopilar datos sobre el consumo de batería de dispositivos móviles que se enviarán a un centro de *big data*. Estos datos serán procesados para posteriormente poder obtener algunas recomendaciones personalizadas sobre los distintos usuarios que permitan que se realice un consumo de energía más eficiente. Principalmente se ofrecen estadísticas sobre el consumo de energía debido a cada una de las aplicaciones del dispositivo móvil.

La arquitectura del sistema queda reflejada en la imagen de la Figura 1.3.4.

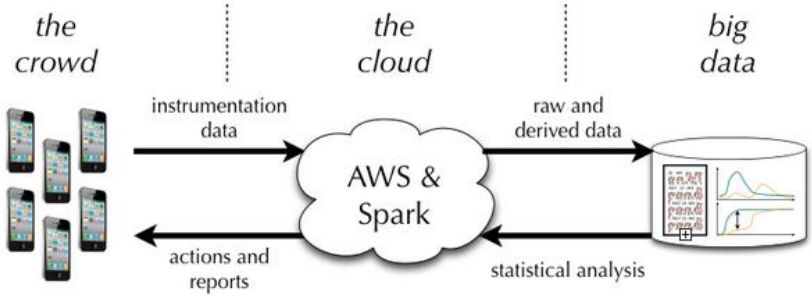

Figura 1.3.4: Arquitectura de la aplicación *'Carat'*

Según se indica en [37], el envío de datos es totalmente anónimo aunque estos datos si podrán ser utilizados en publicaciones académicas ya que se trata de un proyecto de investigación.

Nuevamente nos encontramos ante una aplicación que hace uso del modelo de *crowdsourcing* para la obtención masiva de datos ofreciendo ciertas funcionalidades al usuario final que incentivan la descarga y uso de la aplicación. En la página *web* se puede ver que actualmente cuentan con un total de 725.050 usuarios (28 de Abril de 2014).

En la Figura 1.3.5 se presenta una muestra de la interfaz gráfica de esta aplicación.

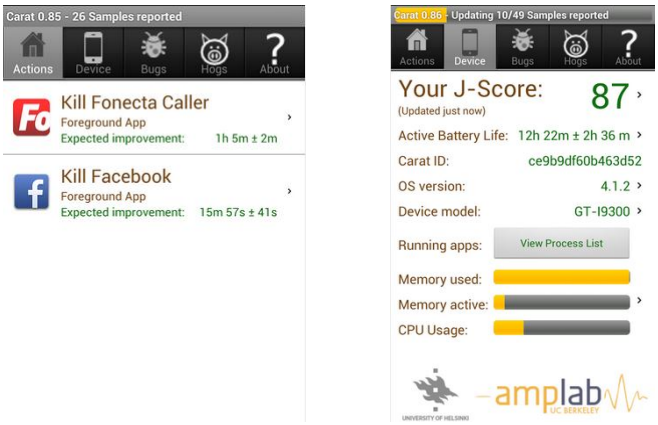

Figura 1.3.5: Aplicación *'Carat'*

OpenSignal

'*OpenSignal*' [39, 40] es una aplicación desarrollada a partir de un proyecto de investigación de un equipo de físicos de la Universidad de Oxford (Reino Unido). Se trata de una aplicación con soporte tanto para Android como para IOS.

Esta aplicación recoge anónimamente mediciones de nivel de señal de celdas de operadores móviles de una gran cantidad de usuarios así como mediciones de redes Wi-Fi. Estas mediciones se realizan junto con su localización asociada. Con ello buscan generar un mapa global en el que se puedan conocer los niveles de señal de celdas móviles para 2G, 3G o 4G y para los distintos operadores. En el caso de las mediciones de señal Wi-Fi análogamente se pretende generar una base de datos con tantas mediciones como sea posible.

Dicho proyecto alcanza una magnitud considerable tal y como se anuncia en su página *web* [39] en la que aparecen unas cifras de 5.186.324.530 mediciones de señal de cobertura móvil y 1.230.834.497 mediciones de puntos de acceso Wi-Fi (28 de Abril de 2014).

En la ilustración de la Figura 1.3.6 se puede observar un mapa de Granada y alrededores que cuenta con una cantidad de mediciones más que considerable.

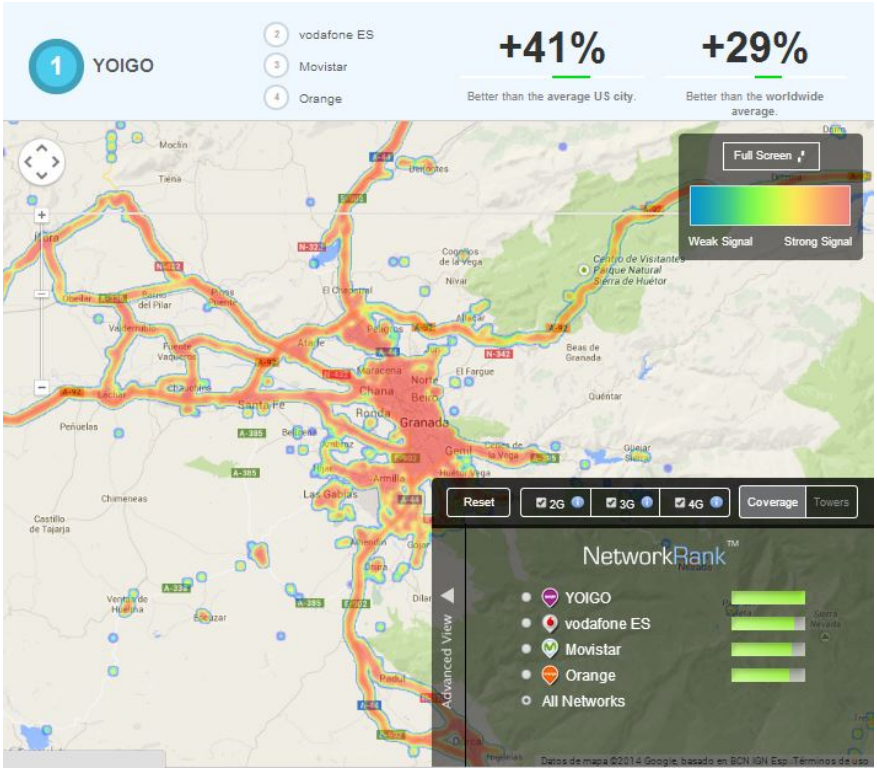

Figura 1.3.6: Mapa de cobertura obtenido por *'OpenSignal'*

Como se puede observar en la imagen también se listan en orden de mayor a menor nivel de cobertura las distintas operadoras móviles que ofrecen servicios en dicha área.

En las imágenes de la Figura 1.3.7 se puede observar que cuenta con una interfaz gráfica intuitiva y estética.

Figura 1.3.7: Aplicación *'OpenSignal'*

Este proyecto, que hace uso de la filosofía *crowdsourcing*, ha tenido una gran expansión y cuenta con empresas patrocinadoras muy relevantes a nivel internacional como las cadenas de televisión 'CNN' y 'BBC' o la revista 'TIME'.

pressureNet

El proyecto '*pressureNET*' [41, 42] nace en Toronto (Canadá) y es de código abierto. Se trata de una aplicación con soporte para Android que sólo puede ponerse en funcionamiento en dispositivos móviles que incorporen un barómetro.

El objetivo de esta aplicación es obtener un mapa de predicciones meteorológicas de gran resolución y gran precisión. Para ello, en los dispositivos móviles que tengan instalada esta aplicación se tomarán periódicamente mediciones de la presión atmosférica y serán enviados a una base de datos. Con ello se pretende obtener a lo largo de todo el mundo datos que permitan crear un mapa muy detallado de la presión atmosférica medida de forma precisa así como de predicciones relacionadas con este parámetro (anticiclones, borrascas, etcétera).

Al tratarse de un sistema de *crowdsourcing*, la resolución de este mapa depende del número de usuarios que colaboren utilizando esta aplicación y de lo repartidos que estén a lo largo de la geografía mundial.

Además de existir la aplicación para que se la pueda descargar cualquier usuario, ofrecen una API que permite integrar este servicio de medición en cualquier otra aplicación para así tratar de que algunos desarrolladores interesados con colaborar con este proyecto puedan incluirla en sus propias aplicaciones. Esto dotará al proyecto de un gran potencial de usuarios colaboradores.

En su página *web* [41] se muestra un mapa en el que se puede observar la evolución de la presión atmosférica en el periodo de días que el usuario desee y en la zona geográfica que este indique. A modo de ejemplo se presenta una

imagen en la Figura 1.3.8 en la que aparece la evolución de la presión atmosférica en la ciudad de Granada y alrededores a lo largo de 6 días (22 a 27 de Abril de 2014).

Figura 1.3.8: Mapa de presión atmosférica de *'pressureNET'*

En la Figura 1.3.9 se puede observar la interfaz gráfica de la aplicación desarrollada para dispositivos móviles Android.

Figura 1.3.9: Aplicación *'pressureNET'*

1.3.3. Síntesis de la información recogida

A modo de resumen de todos los trabajos recogidos en la revisión del estado de arte, se detallan a continuación las aportaciones que pueden tener todos éstos al presente proyecto.

En primer lugar, cabe destacar que como fuente principal de inspiración para la elaboración de este proyecto se tomó la aplicación '*Portolan Network Tools*'. Esta aplicación contiene una serie de herramientas que permiten realizar una localización periódica del dispositivo al mismo tiempo que se realizan medidas de nivel de señal de la red móvil. Esto será de gran utilidad ya que servirá de modelo de referencia para incluir esta misma funcionalidad en la aplicación que se va a desarrollar en el proyecto. Asimismo se trata de una aplicación basada en

la filosofía de *crowdsourcing*, cuyo despliegue es totalmente gratuito y se podrán recoger datos anónimamente y con un coste mínimo.

Si se analizan el resto de aplicaciones recogidas en la revisión del estado del arte, se puede afirmar que todas éstas siguen el modelo de *crowdsourcing* y se trata de plataformas que surgen en diversos proyectos de investigación de distintas universidades distribuidas alrededor del mundo. Todas ellas tienen como fin último la obtención de datos de ciertos de interés de forma masiva y económica por medio de dispositivos móviles. Por ello resultan de interés para la elaboración del presente proyecto, ya que el objetivo de éste es obtener una gran base de datos que posteriormente dé lugar a posibles estudios en los que se obtengan diversos resultados de carácter estadístico. En este caso los resultados se pretenden extraer de parámetros de QoS, de QoE, relaciones entre ambas, creación de mapas de cobertura a partir de los niveles de señal obtenidos o estadísticas de reproducción de vídeos sobre YouTube.

Por último, cabe realizar una especial mención al primero de los artículos expuestos. Éste es quizás el que plasma el objetivo que más se acerca al del presente proyecto. Sin embargo, se pueden encontrar algunas diferencias destacables. Por un lado, se trata de una plataforma en la que sólo se visualizan vídeos de un servidor de *streaming* concreto que utiliza el *software 'Darwin Streaming Server'*, mientras que en el presente proyecto se podrán buscar y visualizar vídeos del servidor de vídeos con mayor afluencia del mundo (YouTube). Concretamente se encuentra en el puesto número 3 de las páginas más visitadas en todo el mundo [15]. Además, mientras que en la aplicación de dicho artículo se utiliza sólo un método de monitorización de QoS y obtención de la calidad percibida por el usuario, en el presente proyecto se pretende también tomar datos de niveles de cobertura de red asociados a la localización actual así como otra serie de datos. Entre otros, se obtendrán parámetros de QoS a nivel de aplicación y datos estadísticos sobre tópicos, número de reproducciones, etcétera, que no se contemplan en la otra aplicación. Todo ello añadido a que, además, la aplicación descrita en el artículo no ha sido liberada públicamente. Únicamente se ha liberado una API que contiene todas las funciones que han utilizado en la aplicación tal y como se expone en [31].

1.4. Logros y aportaciones

En un principio la principal motivación de este proyecto consistía en la obtención de una base de datos suficiente como para obtener modelos que permitieran realizar estimaciones del nivel de QoE a partir de una serie de características de QoS dadas y viceversa. De este modo, la principal aportación sería la obtención de una base de datos importante que diera pie a un futuro análisis de los datos. Además se trataría de un modelo basado en *crowdsourcing* en el que el coste se reduciría prácticamente al mantenimiento del servidor de base de datos. Esto supondría la utilización de una plataforma de recolección de datos muy rentable económicamente.

Hasta el momento puede parecer que no se está aportando ninguna novedad, ya que si uno se remite a la revisión del estado del arte, podrá apreciar que actualmente existen numerosos estudios en los que se obtienen modelos que relacionan

la QoE y la QoS sobre el *streaming* de vídeo. Sin embargo, el presente proyecto difiere de todos éstos en cuanto a varios factores distintos. Por un lado en este proyecto se podrán realizar estudios concretos sobre el servicio de *streaming* de YouTube, actualmente líder mundial en su ámbito. Esto resulta importante ya que se ha podido observar que en la actualidad la QoE sobre este servidor en concreto no ha sido estudiada aún en profundidad. Por otro lado, algunos estudios que se han podido encontrar relativos a este servidor se refieren al *streaming* de vídeo sobre ordenadores personales, lo que no será extensible al *streaming* sobre dispositivos móviles. Esto se debe entre otros factores a que, como se ha podido comprobar en la revisión bibliográfica, la generación de tráfico de YouTube para dispositivos móviles difiere respecto de la convencional utilizada para ordenadores personales.

Estos modelos serán de gran utilidad en tanto que se utilizan a menudo en ingeniería de tráfico. Apoyándose en estos modelos, los proveedores de servicios y los propios servidores de contenidos pueden tratar de buscar una serie de políticas que busquen un punto óptimo en el que se ofrezca el nivel de servicios deseado con la mayor eficiencia posible. Esto supone grandes reportes económicos puesto que se utilizarán técnicas que conseguirán un aprovechamiento máximo de los recursos. De este modo se evitará llevar a cabo una política de sobredimensionamiento que garantice el correcto funcionamiento de los servicios con su consecuente coste adicional.

Después de lo expuesto en el apartado de motivaciones, es más que sustentable la idea de que los dispositivos móviles juegan actualmente y jugarán en un futuro un papel muy importante en Internet. Asimismo, el *streaming* de vídeo es el servicio con mayor porcentaje de tráfico total de Internet y tiene una tendencia al alza. Por ello, con estos datos no se puede negar que la combinación de estas dos tecnologías hace de este proyecto un trabajo de gran alcance en cuanto al potencial número de usuarios que abarca y a las grandes cantidades de tráfico que se manejan. Por ende será también potencialmente importante en lo que a rentabilidad económica se refiere puesto que, como ya se ha explicado anteriormente, los modelos que se pretenden obtener tratan de buscar el punto óptimo de compromiso entre el rendimiento y el nivel de recursos necesarios.

Posteriormente se han recogido otros datos además de los inicialmente previstos que permitirán obtener otro tipo de resultados y mediciones. Por ejemplo, al mismo tiempo que se está reproduciendo el vídeo se medirán los niveles de cobertura de la red móvil, de la red Wi-Fi o de ambas, según a la red que se esté conectado en ese momento. Esto permitirá posteriormente trazar un mapa en el que se muestren todas las mediciones de los niveles de cobertura para cada una de las tecnologías móviles (GSM, UMTS, LTE...) y para las células Wi-Fi.

Otra posible funcionalidad sería realizar estudios estadísticos en los que se manejen variables tales como el número de reproducciones, los tópicos relacionados, el número de 'me gusta' y 'no me gusta' para cada vídeo. Asimismo, los datos van asociados al sexo y edad del usuario, por lo que también podrían entrar en el modelo estadístico dichas variables demográficas.

Por último también cabe destacar que se tendrán también ciertas estadísticas sobre el tiempo de *buffering* antes de iniciar el vídeo o el número de interrupciones y la duración de estas. Esto permitirá ya no sólo realizar estudios entre QoS y

QoE, sino también realizar algunos estudios que relacionen estas características con el tipo de red que se está utilizando.

En definitiva, se trata de creación de una base de datos realizada con la pretensión de incluir todos los datos que puedan resultar de interés de modo que sea muy versátil a la hora de realizar posibles estudios. No se descarta de hecho que conforme se vayan analizando los datos almacenados puedan ir surgiendo algunos estudios que no se tenían contemplados *a priori* con las variables de las que se disponen.

Todos los datos que se recopilan se podrán consultar en el Capítulo 4 de esta memoria donde se encuentran debidamente descritos.

1.5. Principales fuentes bibliográficas

Se exponen seguidamente las principales fuentes bibliográficas que se han utilizado para la realización de este proyecto y que han servido de fuentes de inspiración y apoyo para el mismo.

Artículos en los que se realizan estudios sobre el *streaming* de vídeo sobre TCP

Estos artículos servirán de apoyo para encontrar algunos posibles resultados que puedan ser de utilidad en el presente proyecto. También servirán para valorar qué parámetros pueden resultar interesantes para recopilar en la base de datos.

- [13]T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE), Dana Point, CA, USA, 2011.

- [22]R. Mok, E. Chan, and R. Chang, "Measuring the quality of experience of HTTP video streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, pp. 485-492*, 2011.

- [31]F. Delli Priscoli, V. Suraci, A. Pietrabissa, and M. Iannone, "Modelling Quality of Experience in Future Internet networks," in *Future Network Mobile Summit (FutureNetw), 2012, pp. 1-9*, 2012.

- [18]X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos," in *Proc. IEEE Int. Workshp Quality of Service (IWQoS), pp. 229–238*, 2008.

- [33]University of Pisa, "Portolan network sensing architecture." [cited 2014 April 28]. Available from: http://portolan.iet.unipi.it/ .

Páginas *web* para programación

Estas páginas *web* se utilizarán como documentación oficial para el desarrollo de la aplicación en Android para dispositivos móviles y para el programa en Java en el lado del servidor.

- [43]Android Developers webpage. [cited 2014 April 28]. Available from: http://developer.android.com/ .
- [44]Java Platform, "Standard Edition 7 API Specification." [cited 2014 May 30]. Available from: http://docs.oracle.com/javase/7/docs/api/ .

APIs para Android de YouTube

Estas referencias se consideran de gran interés en el presente proyecto. Se trata de las principales APIs, excluyendo las propias del SDK (*Software Development Kit*) de Android, que se han utilizado para la programación de la aplicación Android. Con ellas ha sido posible incorporar un reproductor embebido de YouTube en la aplicación y obtener datos de QoS a nivel de aplicación así como otro tipo de datos descriptivos sobre el video reproducido.

- [45]Google Developers, "YouTube Android Player API." [cited 2014 July 22]. Available from: https://developers.google.com/youtube/android/player/ .
- [46]Google Developers, "YouTube Data API (v3)." [cited 2014 July 23]. Available from: https://developers.google.com/youtube/v3/ .
- [47]Google Developers, "YouTube Analytics API." [cited 2014 July 23]. Available from: https://developers.google.com/youtube/analytics/ .

Estas referencias representan las que el autor de este proyecto ha considerado esenciales. No obstante esta lista podría ampliarse considerablemente añadiendo el resto de referencias que se incluyen al final de la memoria.

1.6. Organización de la memoria

En este apartado se definirá la estructura de la memoria con el fin de orientar al lector. La presente memoria consta de cinco capítulos y cuatro anexos que se describen a continuación:

Capítulo 1: Introducción y referencias bibliográficas

En este capítulo se describen todos los conceptos relacionados con el proyecto. Se incluye un apartado de motivaciones que destaca las principales tecnologías que toman parte en el mismo. Además, se expone una amplia revisión bibliográfica con trabajos relacionados realizados hasta el momento, que servirán de apoyo para la elaboración del diseño. Finalmente se realiza una recopilación de todos

los logros y aportaciones con la que se trata de plasmar el alcance que tendrá el proyecto.

Capítulo 2: Análisis de objetivos y metodología

En este capítulo se definirán todos los objetivos que se pretenden alcanzar en un principio y una valoración final evaluando aquellos que se han cumplido. Además se definen todos los requisitos funcionales y no funcionales que se deben considerar en la realización del proyecto.

Una vez definidos los objetivos y requisitos, se desarrolla un segundo bloque en el que se reúnen todos los aspectos teóricos del diseño que se va a elaborar. Se incluirán todas las herramientas y tecnologías utilizadas justificando su uso y se describirán todas las consideraciones y criterios tomados en el proceso de diseño.

Capítulo 3: Planificación y estimación de costes

En este capítulo se exponen de forma detallada todos los aspectos relacionados con la planificación temporal del proyecto, definiendo todas las fases de las que consta y realizando estimaciones del tiempo que se invertirá en cada una de ellas.

Además se identificarán todos los recursos necesarios y se estimarán los costes totales asociados a éstos para finalmente realizar un presupuesto con el coste total del proyecto.

Capítulo 4: Diseño y resolución del trabajo

Este capítulo recoge todos los aspectos relacionados con la implementación del diseño descrito en el capítulo anterior. Se describirán los programas desarrollados y las posibilidades que éstos ofrecen. Se incluirán diagramas de clases UML y diagramas de flujo de estos programas con el fin de que el lector conozca el funcionamiento de éstos sin necesidad de tener que examinar el código.

Posteriormente se recogen resultados obtenidos durante un periodo de fase de pruebas y se realizará una evaluación de éstos.

Capítulo 5: Conclusiones y vías futuras

Este último capítulo con el que concluye la memoria se utilizará a modo de resumen para destacar todos los principales aspectos del proyecto. En este resumen se incluyen consideraciones de aspecto profesional y académico en conjunción con algunas de ámbito más personal.

También se propondrán posibles trabajos futuros que se podrán realizar utilizando como base este proyecto para conseguir otras funcionalidades y resultados.

Anexo A: Manual de instalación

Se trata de un manual en el que se describe en gran detalle todo el proceso necesario para la instalación y puesta en marcha del sistema. Se divide en dos grandes bloques, uno dedicado a la instalación de todas las herramientas necesarias en el servidor y otro dedicado a la instalación y configuración de la aplicación Android para dispositivos móviles.

Anexo B: Manual de usuario

En este anexo se realiza un resumen de cómo se utilizan cada uno de los programas desarrollados en el proyecto. Así, una vez instalados siguiendo las indicaciones del Anexo A, se podrá utilizar este anexo para utilizar correctamente todas las herramientas.

Anexo C: Obtención de trazas utilizando *TCPDump*

Este anexo describe cómo podría realizarse de un modo adicional una captura de paquetes utilizando *TCPDump*. Esto podría resultar de interés para un usuario que desee recoger las trazas de paquetes enviados y recibidos al mismo tiempo que se realizan todas las tareas de monitorización desde la aplicación de Android mientras se reproduce un vídeo.

Anexo D: Consultas básicas en *mySQL*

En este anexo se incluirán algunas nociones básicas del lenguaje SQL (*Structured Query Language*) necesario para llevar a cabo consultas en la base de datos desarrollada en el presente proyecto. Posteriormente se incluyen algunos ejemplos prácticos que resultarán de interés para el lector ya que se trata de consultas que pueden ser necesarias frecuentemente para extraer información concreta de la base de datos.

Capítulo 2

Análisis de objetivos y metodología

El presente capítulo representa todas las cuestiones previas que se han abordado antes de comenzar con el diseño y la implementación del presente proyecto.

Para ello, en primer lugar, se tratarán de recoger todos los objetivos principales y secundarios que se desean alcanzar con el fin de establecerlos como guía para el proceso de diseño e implementación. Entre los objetivos se incluirán algunos opcionales cuya consecución se irá valorando a lo largo del desarrollo del proyecto para determinar si finalmente son factibles o no.

En una segunda parte se identificarán todos los requisitos funcionales y no funcionales que se deben cumplir de un modo genérico en el proyecto y para cada una de las herramientas individuales desarrolladas en concreto. Estos requisitos serán producto de los objetivos que se marcan en el apartado anterior.

Se incluye a continuación una breve valoración donde se describe a posteriori el alcance de los objetivos que se marcaron en un inicio. Así, se podrá contemplar el cumplimiento de éstos y la inclusión de nuevos objetivos que han surgido adicionalmente a lo largo de la etapa de diseño e implementación.

Finalmente, se desarrolla un apartado en el que se describe la metodología que se ha seguido a lo largo de todo el proyecto para su realización satisfactoria. Se describirán a grandes rasgos todas las etapas de las que ha constado y los aspectos más esenciales de cada una de estas etapas.

2.1. Objetivos

El objetivo último de este proyecto es obtener una base de datos con toda la información posible que pueda resultar de interés para posteriores estudios estadísticos de *streaming* sobre vídeo del servidor YouTube.

Para ello se propone diseñar una aplicación para dispositivos móviles Android que permita la monitorización de parámetros de QoS al mismo tiempo que se

reproducen vídeos del servidor YouTube sobre una interfaz gráfica atractiva y sencilla de manejar. Al finalizar la reproducción del vídeo, se pretende invitar al usuario a realizar una breve encuesta que permita obtener una serie de datos acerca de la calidad percibida por éste. Al término de esta encuesta, se podrán enviar todos los datos conjuntamente a un servidor de base de datos. Esta aplicación debe ser para el usuario final una alternativa a la aplicación oficial de YouTube que ofrezca alguna funcionalidad extra que no contemple esta última y que no conlleve un gran coste de adaptación a su interfaz. Esto facilitará en gran medida la distribución de la aplicación y la colaboración del usuario en el proyecto.

El desarrollo de esta aplicación permitirá obtener una base de datos lo más completa posible en la que tratarán de incluirse todos los parámetros de QoS conocidos y medibles tanto a nivel de red como a nivel de aplicación. A nivel de red se tratarán de recopilar mediciones de indicadores de las condiciones de la red tales como tipo de conexión, ancho de banda, *jitter*, nivel de señal, etcétera. A nivel de la capa de aplicación se pretenderán capturar eventos que ocurran durante la reproducción de un vídeo, como pueden ser las interrupciones causadas por congestión, fallo en la red o por interrupción forzada por el usuario. También será importante conocer el tiempo de *buffering* que transcurre después de cada interrupción o el tiempo de carga al inicio del vídeo.

No obstante se pretende realizar una recopilación de datos lo suficientemente completa como para que puedan surgir gran cantidad de oportunidades a la hora de manejar estos datos y poder realizar estudios interesantes que quizás no habían sido contemplados al inicio de la realización de este proyecto.

Por otro lado se pretende que la aplicación del dispositivo móvil y el sistema desarrollado en su conjunto sea en la medida de lo posible escalable y fácilmente adaptable a la evolución de las tecnologías. Con ello se tratará de facilitar el funcionamiento de la aplicación sobre nuevas tecnologías de red, versiones de sistemas operativos o políticas de envío de tráfico de YouTube. Esto último es importante en tanto que en un mundo como el de las TICs, que se encuentra en continua evolución, puede resultar interesante realizar nuevos estudios ante la aparición de una nueva tecnología sin que esto suponga realizar un nuevo diseño para la construcción de la base de datos.

En la Figura 2.1.1 se puede observar una ilustración esquemática que simplifica el funcionamiento del sistema que se pretende diseñar.

2.2. Especificación de requisitos

En este apartado se va a llevar a cabo un análisis profundo de todo el proceso de diseño tratando así de realizar una previsión lo más completa posible que incluya un listado de todos los requisitos funcionales y no funcionales que se deben cumplir a lo largo de todo el diseño.

Este proceso será importante ya que se deberán tener en mente todos estos requisitos durante el diseño. Con ello se conseguirá que al término del diseño, el trabajo realizado sea exitoso y no haya que realizar grandes remodelaciones con el respectivo consumo de tiempo adicional que ello conlleva.

Figura 2.1.1: Esquema del sistema a diseñar en el presente proyecto

2.2.1. Requisitos funcionales

En este apartado se listan los requisitos mínimos funcionales que se deben incluir en el diseño final con el objetivo de alcanzar satisfactoriamente los objetivos planteados en la Sección 2.1:

- **Búsqueda y reproducción de vídeos del servidor YouTube:**

  La aplicación móvil debe posibilitar la búsqueda de vídeos a partir de una descripción escrita por el usuario tal y como ocurre en la aplicación oficial de YouTube, y a continuación ofrecer un listado con todos los vídeos de modo que el usuario pueda seleccionar uno de ellos para reproducirlo.

- **Monitorización de parámetros de QoS a nivel de red y de aplicación:**

  La aplicación debe recopilar tantos datos como sea posible que ofrezcan alguna información relevante sobre la calidad de servicio (QoS), ya sea a nivel de red o a nivel de aplicación. Para ello se monitorizarán parámetros indicadores de la calidad del tráfico de red y se tratarán de detectar los posibles eventos que se producen a nivel de aplicación durante la reproducción.

- **Realización de una breve encuesta:**

  Al término de la reproducción, cuando el usuario desee salir de dicha pantalla, se le invitará a realizar una breve encuesta que trate de recopilar alguna información sobre la QoE. Esta encuesta deberá ser sencilla de rellenar para evitar disuadir al usuario de su realización.

- **Almacenamiento de datos y envío automático:**

  En el caso de que el usuario haya accedido a realizar la encuesta se le presentará la opción de enviar a un servidor los datos de monitorización

previamente almacenados junto con los datos de la encuesta. Todos estos datos se enviarán de un modo anónimo.

- **Creación y actualización de base de datos:**

  En el lado del servidor debe programarse una rutina que permita crear una base de datos junto con las tablas necesarias de forma automática y cuya ejecución permita ir añadiendo los nuevos datos que se vayan recibiendo. Esto dará lugar a una base de datos bien ordenada que permita el fácil acceso y procesado de los datos.

2.2.2. Requisitos no funcionales

Una vez detallados los requisitos funcionales, se presentan seguidamente una serie de requisitos no funcionales que serán en su mayoría de carácter opcional pero no por ello serán menos importantes.

- Proceso de monitorización, almacenamiento y envío de datos lo más transparente posible para el usuario. Aunque el usuario esté informado sobre estos procedimientos que se realizarán en segundo plano, éste no deberá percibir que haya alguna repercusión derivada de estos procesos sobre el normal funcionamiento de la aplicación para la reproducción de vídeos.
- La aplicación debe programarse de modo que abarque el máximo número posible de versiones de Android, aunque en algunas de ellas las funciones sean más limitadas. Con ello se tratará de alcanzar potencialmente un mayor número de dispositivos móviles, lo que favorecerá que se consiga un mayor volumen de datos recopilados. Un dato a tener en cuenta es que Android mantiene una política de compatibilidad hacia atrás, por lo que esta aplicación deberá funcionar correctamente en las versiones futuras de Android.
- La interfaz debe ser atractiva, sencilla e intuitiva para el usuario final. Esto favorecerá el uso de la aplicación frente a otras alternativas como puede ser la propia aplicación oficial de YouTube.
- Se tratarán de incluir en la medida de lo posible algunas funcionalidades que no se ofrezcan en la aplicación oficial de YouTube tratando así de dotar a la aplicación de alguna ventaja competitiva que compense otros aspectos tales como el tiempo invertido en la realización de la encuesta.
- El formulario de la encuesta será breve y conciso con el fin de que no suponga un gran coste para el usuario la realización de ésta. Con ello se obtendrá una cantidad de encuestas presumiblemente más abultada que en el caso de que se ofrezcan formularios más extensos. No obstante tratarán de recopilarse los datos más importantes de QoE y otros datos estadísticos que puedan resultar interesantes.
- Se debe facilitar en la medida de lo posible una modificación futura tanto de los módulos de monitorización como del formulario de la encuesta. Esto permitirá que en un futuro se puedan ir realizando adaptaciones de la aplicación en función de nuevos datos que puedan interesar o de nuevas funcionalidades que se ofrezcan en las versiones más recientes de Android.

- Se debe incluir una sección dentro de la aplicación de Android que informe al usuario sobre los objetivos del proyecto donde además se incluya el tipo de datos que serán enviados al servidor. En esta descripción se debe destacar que el envío y almacenamiento de datos tiene un carácter anónimo, ya que carece de cualquier identificador personal. Esto motivará además la participación del usuario en el proyecto.

- El consumo adicional de datos debidos al envío de la información recopilada al servidor se tratará de minimizar en la medida de lo posible. Esto evitará que el usuario decida dejar de utilizar la aplicación por este motivo, ya que en la actualidad existen una gran cantidad de planes de datos sobre Internet para dispositivos móviles sobre los que se imponen restricciones de volumen de datos enviados y recibidos mensualmente.

- Se debe preservar en la medida de lo posible la privacidad de los datos almacenados. Para ello será necesario utilizar canales seguros para el envío de la información recopilada desde los dispositivos móviles al servidor. Asimismo, en el propio servidor, la información debe estar suficientemente protegida para restringir el acceso de terceros a esta información.

- Tanto la aplicación desarrollada para el cliente como el servidor deben ser robustos y tolerantes a fallos. Para ello se tratará en la medida de lo posible que, ante un fallo, el sistema pueda volver a su normal funcionamiento fácilmente. Además se facilitará el diagnóstico de posibles fallos recurrentes que pueden ocurrir. En el Anexo B se recogen errores comunes que pueden producirse y se describe cómo resolverlos rápidamente.

En definitiva, en estos puntos se trata de recoger una serie de directrices que se deberán tener en cuenta a lo largo del diseño para conseguir un acabado los más completo y exitoso posible.

2.3. Valoración del alcance de los objetivos

Este apartado se escribe al finalizar la implementación del presente proyecto con el fin de evaluar el cumplimiento de los objetivos previstos en un inicio.

Una vez completado todo el proceso de implementación se puede afirmar que todos los objetivos planteados en el Apartado 2.1 de esta memoria se han alcanzado por completo.

Esto indica que se han cumplido las expectativas previstas. Aunque eso no es todo, ya que se han alcanzado algunos objetivos adicionales. Los principales objetivos nuevos que se han alcanzado se deben a nuevos datos que en un principio no se contemplaron y que a lo largo del proceso de implementación se han considerado interesantes. Estos datos se listan seguidamente junto con las aplicaciones que pueden tener:

- Obtención de mediciones de RSSI de la red móvil y de la red Wi-Fi. Durante la reproducción de un vídeo se tomarán medidas de RSSI (*Received Signal Strength Indication*) asociadas a la localización del dispositivo móvil

en ese momento. Cada vez que se detecte una nueva posición, se grabarán nuevas mediciones de RSSI. Las mediciones se realizarán sobre la red móvil en cualquier caso y adicionalmente sobre la red Wi-Fi si el dispositivo móvil está conectado mediante esta tecnología. Estas mediciones permitirán trazar mapas de cobertura tanto para redes móviles como para redes Wi-Fi. Además, podrán realizarse consultas en función de la localización y del tipo de tecnología de red (GSM, UMTS, LTE...). Este tipo de trabajos ya se realizan en proyectos de investigación de gran envergadura [33, 39].

- Obtención de datos relacionados con el carácter de red social que caracteriza a YouTube. Para ello se recogen estadísticas como el número total de reproducciones de los vídeos, los números de 'Me Gusta' y 'No Me Gusta' de los usuarios y los tópicos principales y secundarios asociados a los vídeos. Estos datos permitirán obtener estadísticas sobre la propagación de los vídeos en el servidor de YouTube o la clasificación por tópicos de éstos. También permitirán inferir algunos aspectos comportamentales de los usuarios de este servidor. En la revisión bibliográfica ya se incluía algún estudio relacionados con estos parámetros [18].

- Se ofrece la posibilidad de obtener trazas con el tráfico entrante y saliente. Esto permitirá analizar cómo el servidor de YouTube gestiona el tráfico para el *streaming* de vídeos sobre TCP. Para la obtención de estas trazas se necesitará tener el dispositivo móvil conectado a un ordenador mediante un cable USB y no se realizarán de un modo automático ya que no ha sido posible. Sin embargo, en el Anexo C se describe en gran detalle cómo podría realizarse esto. Estas trazas permitirán por ejemplo conocer los códecs utilizados o estudiar los patrones de envío de YouTube, como se analiza en el artículo [20].

En estos puntos sólo se incluye un resumen de las nuevas aportaciones no contempladas en un inicio. No obstante, en el Apartado 4.4 de esta memoria podrán observarse con mayor claridad y en mayor detalle todos los resultados finalmente conseguidos.

2.4. Metodología

A modo de resumen de todo el trabajo realizado en el presente proyecto, se presenta en este apartado el desarrollo que se ha seguido a lo largo de éste. Para ello se describirá cronológicamente toda la elaboración del proyecto en su conjunto comenzando por las fases previas de formación.

En primer lugar, antes de comenzar con el diseño de las herramientas del proyecto, se ha realizado una revisión bibliográfica. Esta revisión ha permitido contemplar el panorama actual en el ámbito de la investigación de temáticas relacionadas con el proyecto. En ella se recopilan artículos y herramientas que han servido de apoyo posteriormente en la fase de diseño.

Respecto a la elaboración de la aplicación de Android, ha sido necesario un periodo de formación para comenzar a programar en este sistema operativo y comprender en un nivel bastante avanzado este paradigma de programación.

Una ventaja para el autor del presente proyecto es que se trata de una plataforma basada en el lenguaje Java, con el que ya se encontraba familiarizado.

Una vez completado el periodo formativo, se ha procedido en primera instancia a diseñar la aplicación Android para el cliente. En este proceso ha sido necesario identificar aquellos parámetros y datos que se iban a recopilar. Además se ha realizado un diseño detallado de la interfaz de la aplicación para establecer cómo puede interactuar el usuario con ella. Asimismo, se ha necesitado diseñar un método para almacenar toda la información recopilada y enviarla hacia el servidor de un modo seguro y con un formato estándar para que pueda ser procesada automáticamente.

Durante el proceso de implementación han ido surgiendo nuevas problemáticas y nuevas proposiciones de diseño hasta finalmente alcanzar una versión estable tras un periodo de pruebas en busca de posibles fallos. En esta misma etapa han ido surgiendo nuevos objetivos que no habían sido contemplados al inicio y que se consideran interesantes.

Al término de la aplicación para el cliente, ha sido necesario poner en marcha toda la infraestructura del lado del servidor. Se ha creado un servidor de base de datos y un programa que actualiza automáticamente la información nueva que se va recibiendo. En este servidor se han tenido en cuenta cuestiones relacionadas con la seguridad para evitar posibles ataques y preservar la información recopilada.

Finalizada esta última fase, se ha procedido a utilizar el sistema en su conjunto para comenzar a recopilar datos durante una fase de pruebas que ha servido para detectar posibles fallos y para recoger algunas muestras que permitan mostrar algunos resultados. Estos resultados permitirán motivar el alcance que puede tener esta base de datos.

Como conclusión final, se han evaluado los resultados finales obtenidos. Con ello se han podido valorar los objetivos propuestos a priori y los objetivos alcanzados finalmente. Además se han presentado posibles mejoras o extensiones para un trabajo futuro que pueda tener como base lo realizado en el presente proyecto.

Capítulo 3

Planificación y estimación de costes

En el presente capítulo se van a tratar todos los aspectos relacionados con la planificación del diseño y la estimación de costes *a priori*.

Así, en primer lugar se tratará de diferenciar las etapas de las que constará el proyecto incluyendo una descripción del trabajo que implicará cada una de ellas. A continuación se procederá a realizar una estimación de la temporización de cada una de las etapas para finalmente realizar un diagrama de Gantt que represente gráficamente el desarrollo del proyecto de inicio a fin.

Una vez analizado todo lo relativo a la temporización del proyecto, se tratarán de identificar todos aquellos recursos que estarán involucrados en éste. En esta previsión de recursos se realizará una diferenciación entre los recursos humanos, *hardware* y *software*.

Por último, una vez realizada una estimación temporal e identificados todos los recursos que tomarán parte, se procederá a realizar una estimación de los costes que se necesitarán para abordar el proyecto completo.

3.1. Planificación

En este apartado se describe una planificación en la que se distinguirán cada uno de los paquetes de trabajo en los que se dividirá el proyecto. Para ello se ha llevado a cabo un profundo análisis de cómo debe desarrollarse el proceso con el fin de que este se ajuste lo máximo posible al transcurso real del proyecto.

Los paquetes de trabajo en los que se ha dividido este proyecto son los siguientes:

PT1: **Revisión bibliográfica**

La primera etapa consistirá en realizar una profunda investigación sobre las publicaciones científicas que hay hasta el momento relacionadas con la temática del proyecto. Esto tendrá dos funciones principalmente: por un lado será necesario conocer lo que hay en el mundo

científico hasta el momento para evitar realizar algo que ya existe. Por otro lado todas estas publicaciones servirán como fuente de inspiración y orientación para la elaboración del presente proyecto, ya que se aprenderán nuevos conceptos y nuevas técnicas utilizadas que el autor podría desconocer.

PT2: Toma de contacto con la programación en Android

Esta segunda etapa, de gran relevancia, consistirá en aprender las nociones básicas de la programación en Android. Puesto que el autor de este proyecto conoce el lenguaje de programación Java no supondrá un gran coste. Sin embargo será necesario un proceso de adaptación a las nuevas APIs del SDK de Android frente a las que ofrece Java. Asimismo también será importante conocer cómo se utiliza el lenguaje XML para definir las interfaces gráficas y cómo el usuario interactúa con los diferentes elementos gráficos.

PT3: Estudio de las APIs de YouTube

Deberán analizarse en profundidad las funcionalidades que ofrecen las APIs de YouTube para Android con el fin de sacar el máximo partido. Con ello el autor podrá ser consciente de qué tipo de datos y parámetros se pueden recopilar. Además serán necesarias para el proceso de búsqueda y reproducción de vídeos. Por ello podría decirse que estas APIs proporcionarán la mayoría de las funciones clave que se desean implementar en la aplicación de Android.

PT4: Estudio de los datos y parámetros que se van a recopilar

En esta fase se utilizarán como base las APIs de Android y YouTube analizadas en las etapas anteriores junto con los artículos de la revisión bibliográfica para realizar una recopilación de todos los parámetros que es posible monitorizar y que puedan resultar de interés en el presente proyecto. En este proceso se hará especial hincapié en la búsqueda de parámetros relacionados con la conectividad de red que permitan obtener datos de QoS a nivel de red y en la detección de eventos en el reproductor de vídeos a partir de las APIs de YouTube con el fin de obtener datos relacionados con la QoS a nivel de aplicación. No obstante se añadirá todo tipo de parámetros relacionados con el proyecto. Además deben elegirse convenientemente las escalas que se van a utilizar así como las unidades de medición.

PT5: **Desarrollo de la aplicación de Android del lado del cliente**

Éste será el proceso más largo y en el que se debe trabajar con mayor esmero, ya que se trata de la principal cara visible del proyecto y la que actuará como plataforma de recolección de datos. Se estima que esta etapa constará a su vez de las siguientes etapas:

- Diseño manual de la interfaz gráfica y de las diferentes pantallas de las que dispondrá la aplicación. Habrá además que diseñar cómo se interactúa con el usuario en cada una de estas pantallas y cómo se intercomunicarán éstas. Para ello se debe diseñar un pequeño diagrama de flujo de la ejecución completa del programa.

- Desarrollo de la aplicación tratando de ser lo más fiel posible a los requisitos funcionales y no funcionales expuestos en el Capítulo 1 de esta memoria. Dentro de este proceso de implementación se deberá trabajar con especial criterio para ir realizando posibles adaptaciones respecto del diseño contemplado al inicio. En la aplicación se pretenden incluir diferentes modos de funcionamiento, al menos uno para el usuario final en el que se cuidará más la estética y otro para el desarrollador en el que se mostrará por pantalla y en tiempo real todo lo que se está monitorizando y los datos que serán enviados finalmente al servidor.

PT6: **Desarrollo de la aplicación del lado del servidor**

Esta etapa engloba todas las aplicaciones utilizadas del lado del servidor necesarias para la recolección de datos y su posterior ordenamiento en una base de datos. Para ello, se necesitarán desarrollar los siguientes elementos:

- Servidor SFTP (*Secure File Transfer Protocol*) que se encargará de la recepción de ficheros desde los dispositivos móviles que contendrán toda la información recopilada en cada envío. Puesto que se utiliza el protocolo SFTP, la comunicación entre ambas partes se establecerá a través de un canal seguro. Con ello se evitarán posibles ataques del tipo *man-in-the-middle*, haciendo así más robusto y seguro el sistema.

- Servidor de base de datos *mySQL*. Se ha elegido este servidor en base a que se trata de un servidor de código abierto y sencillo en cuanto a la tarea de configuración e instalación. Además se trata de un servicio optimizado en cuanto a consumo de datos y velocidad en la realización de las distintas operaciones. La utilización de la base de datos facilitará en gran medida el posterior procesado de los datos cuando se deseen realizar estudios sobre éstos.

Dentro de esta etapa también se incluirá un periodo de familiarización con el lenguaje *mySQL* que se necesitará posteriormente para gestionar la base de datos.

- Desarrollo de un programa escrito en Java para la lectura automática de ficheros, la creación de la base de datos y sus tablas y,

por último, la inserción de los datos. Esta inserción se efectuará de modo que sólo se añadirán aquellos elementos que no se encuentren en la base de datos, tratando así de optimizar el proceso de inserción. La utilización de este programa resultará de gran utilidad ya que se tratará de una herramienta que realizará de modo totalmente automático la conversión de la información de los ficheros recibidos en el servidor SFTP sobre una base de datos bien ordenada.

PT7: **Fase de pruebas**

Una vez realizadas todas las implementaciones necesarias para el funcionamiento del sistema en su conjunto, se establecerá un periodo de pruebas. Este periodo tendrá dos finalidades: por un lado la detección de posibles fallos que no se hayan contemplado a lo largo del periodo de implementación (*debug*), y por otro lado la recolección de datos que permitan mostrar una serie de resultados como producto del sistema implementado.

PT8: **Elaboración de una memoria técnica del proyecto**

Este último apartado corresponde a la elaboración de la presente memoria. Con esta memoria se pretende documentar todo lo relativo al proyecto, recogiendo así desde los aspectos más teóricos, hasta las técnicas y procedimientos utilizados para finalmente mostrar una serie de resultados que avalen la utilidad del proyecto. Se trata de un paquete de trabajo que se desempeñará de modo transversal a lo largo de toda la realización del proyecto con el fin de ir documentando todo desde el inicio.

Una vez identificados y descritos todos los paquetes de trabajo de los que constará el proyecto será necesario realizar una planificación. Esta planificación tiene un carácter totalmente orientativo aunque se tratará de respetar en la medida de lo posible para así ir cumpliendo los distintos plazos y alcanzar el término del proyecto de un modo satisfactorio. En la Tabla 3.1.1 se pueden observar las estimaciones temporales para cada uno de los paquetes de trabajo definidos.

En la Figura 3.1.1 se muestra un diagrama de Gantt que representará gráficamente el periodo sobre el que se extenderán cada una de las tareas.

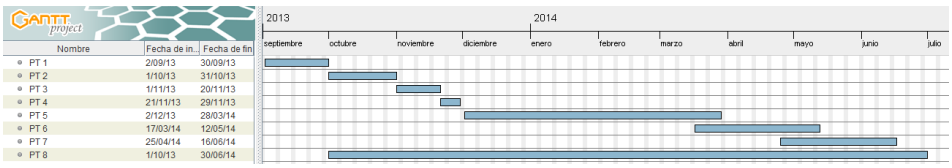

Figura 3.1.1: Diagrama de Gantt estimado del proyecto

Paquetes de trabajo	Descripción	Tiempo estimado
PT1	Revisión bibliográfica	50 horas
PT2	Programación en Android	60 horas
PT3	APIs de YouTube	20 horas
PT4	Elección de datos y parámetros	10 horas
PT5	Aplicación en Android	130 horas
PT6	Aplicación del lado del servidor	50 horas
PT7	Fase de pruebas	30 horas
PT8	Memoria técnica	100 horas
Total		**450 horas**

Tabla 3.1.1: Distribución temporal del proyecto

En el diagrama de Gantt puede observar que a lo largo de la realización del proyecto hay periodos en los que se realiza más de una tarea paralelamente. En estos casos se estima que será necesario sincronizar ambas etapas para que puedan completarse correctamente. Por ejemplo, una vez que la aplicación móvil esté llegando a su fase final, se necesitará empezar a programar el *software* del lado del servidor para comprobar que la comunicación entre ambos se establece exitosamente.

3.2. Recursos utilizados

A continuación se van a tratar de identificar todos los recursos involucrados en el proyecto. Para ello en primer lugar va a realizarse una clasificación en la que se diferencien los recursos de tipo humano, *hardware* y *software*. Las subsecciones que se desarrollan seguidamente corresponderán de forma ordenada a cada uno de estos tipos de recursos.

3.2.1. Recursos humanos

- D. Jorge Navarro Ortiz, Contratado Doctor de la Universidad de Granada en el Departamento de Teoría de la señal, Telemática y Comunicaciones, en calidad de tutor del proyecto.
- José Rafael Suárez-Varela Maciá, alumno del Grado en Ingeniería de Tecnologías de Telecomunicación y autor del presente proyecto.

3.2.2. Recursos *hardware*

- Ordenador portátil Acer Extensa 5635G, con procesador Intel Core 2 Duo T6600 a 2.20 GHz, memoria RAM de 4 GB y disco duro de 285 GB de capacidad. Este ordenador se utilizará principalmente para el desarrollo de la aplicación en Android.
- Ordenador de sobremesa con procesador Intel Core 2 Duo E7500 a 2.93 GHz, memoria RAM de 4 GB y disco duro con una capacidad de 465 GB.

Este ordenador tendrá como funcionalidad principal actuar como servidor SFTP y de base de datos.

- Smartphone LG Nexus 4, con procesador quad-core Qualcomm Snapdragon S4 Pro a 1.5 GHz, 2 GB de RAM, 16 GB de almacenamiento interno y una versión de Android 4.4 (Android Kitkat). Este dispositivo se utilizará para comprobar el funcionamiento de la aplicación y al mismo tiempo que se realiza el proceso de *debug*.
- Línea de acceso a Internet con un ancho de banda moderado. Será necesaria para que el servidor sea accesible desde Internet y así los dispositivos móviles puedan enviar la información recopilada.

3.2.3. Recursos *software*

- Sistema operativo *Windows 8* (64 bits), utilizado en el ordenador portátil sobre el que se van a realizar las tareas de programación de la aplicación Android.
- Sistema operativo *Android 4.4.x* (Kitkat), incluido en el dispositivo móvil sobre el que se realizarán pruebas de la aplicación.
- Sistema operativo *Linux Ubuntu 12.04 LTS*, utilizado en el servidor SFTP y de base de datos donde se almacenará y se podrán consultar todos los datos.
- *Eclipse Kepler* (IDE para desarrolladores de Java), entorno de programación para el desarrollo de la aplicación de Android.
- Android SDK (*Software Development Kit*), conjunto de herramientas proporcionadas por Android necesarias para el desarrollo de aplicaciones.
- *Plug-In* Android para *Eclipse* ADT (*Android Development Tools*), utilizado como herramienta de apoyo para el desarrollo de la aplicación Android.
- *Netbeans* 8.0 para Ubuntu, entorno de programación gratuito para el desarrollo del programa del servidor de base de datos en Java.
- *mySQL workbench*, *software* gratuito para realizar la gestión y consultas sobre la base de datos.
- *phpMyAdmin*, *software* gratuito que servirá como alternativa para gestionar y consultar la base de datos desde una interfaz *web*.
- *mySQL server* y *mySQL client*, *software* gratuito para la instalación del servicio de base de datos.
- *Apache*, servidor HTTP con licencia gratuita para la gestión y consulta de la base de datos desde una interfaz *web*.
- Servidor SFTP *openSSH* que se instalará en el servidor y en el que se almacenarán remotamente todos los ficheros de información recibidos de los dispositivos móviles.

- *Gantt Project*, *software* gratuito para la elaboración de diagramas de Gantt. Se utilizará para diseñar gráficamente la planificación temporal estimada para cada una de las fases o paquetes de trabajo de los que consta este proyecto.
- *Team Viewer*, *software* con una versión gratuita para conectarse mediante escritorio remoto al servidor. Se utilizará para la ejecución remota del programa de actualización de la base de datos del servidor y para realizar consultas a la base de datos también remotamente.
- *Filezilla*, *software* gratuito para la conexión el servidor SFTP. Se utilizará para comprobar remotamente el contenido del servidor SFTP y así poder gestionar los ficheros almacenados e incluso solucionar algunos posibles problemas asociados a estos ficheros en los que se amacena la información recopilada.
- Procesador de textos *Lyx*, de código abierto que utiliza el lenguaje LaTeX para la elaboración de textos. Se utilizará para la elaboración de la presente memoria permitiendo al autor abstraerse en gran medida del lenguaje LaTeX.

Exceptuando el sistema operativo Windows que viene incluido en la compra del ordenador portátil, el resto del *software* utilizado para el proyecto es totalmente gratuito.

3.3. Estimación de costes

En esta sección se tratará de realizar una estimación de los costes que conllevará abordar el presente proyecto.

Como se podrá observar los costes no son elevados. Esto se debe entre otros factores a que el software utilizado es mayoritariamente gratuito.

Recursos humanos

Los costes relativos a recursos humanos se van a calcular en base a las horas que se necesitarán invertir en cada uno de los paquetes de trabajo. Para ello se utilizarán los datos de la estimación temporal recogidos en la Tabla 3.1.1 y se establecerán las siguientes premisas:

- El sueldo medio de un graduado en ingeniería en tecnologías de telecomunicación es de 20 euros / hora.
- El sueldo de un ingeniero Contratado Doctor de la Universidad de Granada se estima en torno a 50 euros / hora. Se considera que en total ha podido invertir entre tutorías para orientar al alumno y la posterior revisión del trabajo unas 15 horas.

De este modo se estipula el presupuesto recogido en la Tabla 3.3.1.

Paquetes de trabajo	Descripción	Coste estimado
PT1	Revisión bibliográfica	1000 euros
PT2	Programación en Android	1200 euros
PT3	APIs de YouTube	400 euros
PT4	Elección de datos y parámetros	200 euros
PT5	Aplicación en Android	2600 euros
PT6	Aplicación del lado del servidor	1000 euros
PT7	Fase de pruebas	600 euros
PT8	Memoria técnica	2000 euros
Trabajo de ingeniero Doctor		750 euros
Total		**9750 euros**

Tabla 3.3.1: Costes estimados de recursos humanos

En la Figura 3.3.1 se representa gráficamente el coste asociado a cada uno de los paquetes de trabajo de los que consta el proyecto.

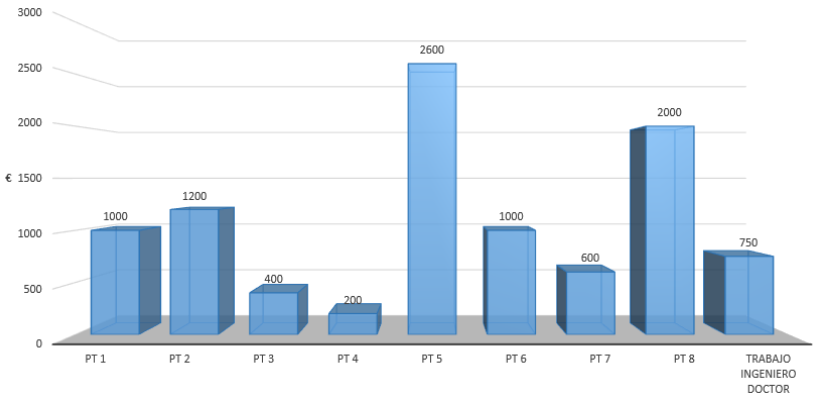

Figura 3.3.1: Coste de recursos humanos estimado para cada paquete de trabajo

Recursos *hardware*

En la Tabla 3.3.2 se recoge un presupuesto aproximado de los recursos *hardware* utilizados.

Recurso	Coste estimado	Vida media
Ordenador portátil	450 euros	36 meses
Ordenador sobremesa	350 euros	48 meses
Smartphone Nexus 4	250 euros	24 meses
Línea acceso a Internet	30 euros/mes	-

Tabla 3.3.2: Costes estimados de recursos *hardware*

Recursos *software*

En cuanto a los recursos *software* empleados, todos los programas y herramientas utilizadas son gratuitos. La única excepción es el sistema operativo Windows del ordenador portátil, cuyo coste no se valorará puesto que la licencia viene incluida en la compra del propio ordenador que ya ha sido referenciada en el apartado anterior.

3.4. Presupuesto final

Por último se presenta el presupuesto final estimado. En la Tabla 3.4.1 se recogen todos los costes previstos considerando una amortización de los recursos *hardware* de 10 meses, que es aproximadamente el periodo previsto para la realización completa del proyecto.

Concepto	**Coste**
Recursos humanos	9750 euros
Ordenador portátil (450 euros x 10 meses/36 meses)	125 euros
Ordenador sobremesa (350 euros x 10 meses/48 meses)	72,92 euros
Smartphone Nexus 4 (250 euros x 10 meses/24 meses)	104,17 euros
Línea de acceso a Internet (30 euros/mes x 10 meses)	300 euros
Total	**10.352,09 euros**

Tabla 3.4.1: Presupuesto final estimado

Siguiendo el presupuesto final estimado se puede observar que el coste se debe en su mayoría a los recursos humanos. En concreto, éstos suponen un 94.18 % del presupuesto total aproximadamente. Esto indica que el proyecto no requiere de grandes esfuerzos económicos en cuanto a *hardware* y/o *software*.

3.5. Consideraciones finales sobre la planificación

Debido a cierta problemática que ha ido surgiendo a lo largo del desarrollo del proyecto, finalmente se ha decidido postergar su entrega hasta Septiembre. Los principales motivos que han motivado este retraso han sido los siguientes:

- Durante un tiempo considerable se intentó implementar un módulo adicional con el que se pudieran realizar capturas de tráfico utilizando una adaptación de *TCPDump* para dispositivos Android. Sin embargo, tras una larga etapa en la que se trataron de explotar todas las alternativas, no fue posible conseguir exitosamente la realización de este módulo. Esta problemática se explica en mayor detalle en el apartado 4.2.4 de esta memoria. Esto produjo que la versión final de la aplicación Android se hiciera realidad más tarde de lo previsto en la planificación estimada al inicio.
- Al tratarse de un proyecto en el que el principal resultado es el conjunto de datos almacenado en la base de datos, resulta de gran interés tener

un conjunto de muestras de un tamaño importante. Esto motivó ampliar la fase de pruebas de modo que se pudiera obtener el máximo número de muestras de tantos usuarios diferentes como fuera posible. Todo ello con el fin de mostrar en el apartado de resultados el potencial que puede alcanzar el sistema desarrollado en este proyecto. No obstante cabe destacar que el fin último de este proyecto no es obtener resultados estadísticos concluyentes, sino conseguir una plataforma que permita la recopilación de datos de un modo económico y con un gran número potencial de usuarios colaboradores.

Finalmente, la planificación se ha realizado de un modo aproximado conforme al diagrama expuesto en la Figura 3.5.1.

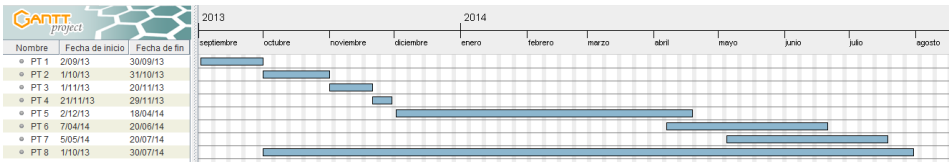

Figura 3.5.1: Diagrama de Gantt con la planificación final del proyecto

En realidad, pese a la entrega del proyecto en Septiembre, la finalización de éste ha tenido lugar a principios del mes de Agosto.

Capítulo 4

Diseño y resolución del trabajo

En el presente capítulo se van a abordar todos los aspectos referentes al diseño y la implementación del sistema que se va a desarrollar en este proyecto y que se ha descrito en términos teóricos en capítulos anteriores.

En primer lugar se va a describir en detalle la arquitectura elegida, motivando la elección de ésta y describiendo todos los elementos que la componen.

El diseño realizado en el proyecto puede dividirse en dos grandes bloques funcionales. Por un lado se tiene la aplicación Android en el lado del cliente, desde la que los usuarios podrán reproducir vídeos y colaborar con el proyecto completando una encuesta y enviando todos los datos monitorizados. Por otro lado se tiene la infraestructura desarrollada en el lado del servidor. El servidor se compondrá de una serie de herramientas y servicios no necesariamente alojados en un mismo *host* que se encargarán de recoger la información enviada desde los dispositivos móviles y de procesarla para insertarla ordenadamente sobre una base de datos. Estos dos bloques descritos corresponderán respectivamente a los apartados 4.2 y 4.3 de este capítulo.

A continuación se incluirá un apartado con los resultados obtenidos tras una fase de pruebas. Estos resultados incluirán algunas estadísticas extraídas de la información recogida en la base de datos con el fin de mostrar el potencial que puede tener dicha base de datos.

4.1. Arquitectura del sistema

La arquitectura genérica del sistema diseñado queda gráficamente descrita en la Figura 4.1.1. Se trata de la alternativa que se ha considerado más adecuada para la implementación que se desea realizar en el presente proyecto.

Esta arquitectura constará de dos servidores principales en los que se gestionará la información recopilada: un servidor SFTP (*Secure Transfer File Protocol*) y un servidor de base de datos *mySQL*. El primero de ellos se encargará de

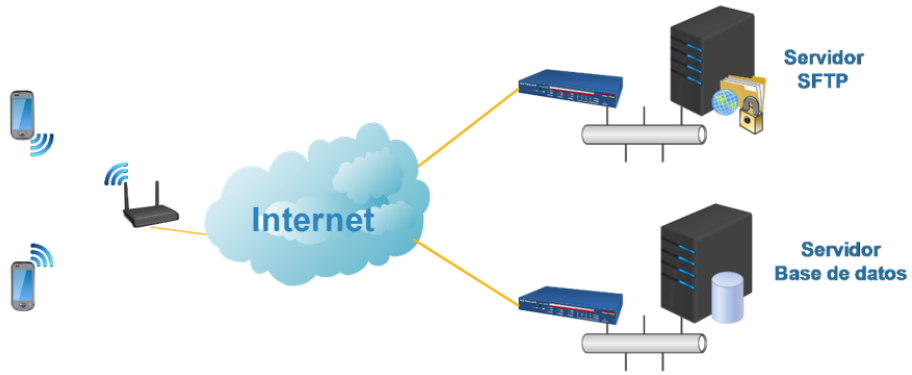

Figura 4.1.1: Arquitectura del sistema diseñado

recopilar todos los ficheros recibidos desde los dispositivos móviles, mientras que en el segundo de ellos se almacenará la información en una base de datos constituida por diversas tablas que sigue un modelo relacional. Esta base de datos permitirá que se realicen consultas de modo que los datos puedan ser procesados fácilmente.

En cuanto a lo que a seguridad se refiere, al utilizar el servidor SFTP en la transmisión de datos, se establece un canal seguro que evitará que se pueda observar el contenido de la información enviada a lo largo de su recorrido por Internet. Por otro lado, el acceso a la base de datos se ha restringido únicamente a la máquina en la que se aloja dicho servicio. Con ello se preservará la privacidad de los datos que ésta contiene.

Aunque se trata de dos servidores independientes que podrían encontrarse en diferentes redes con conectividad entre sí, en el presente proyecto se ha decidido implementarlos sobre una misma máquina. Esta decisión se ha tomado por simplicidad y siguiendo uno de los objetivos que se plantean en este proyecto, la minimización de los gastos económicos. Con ello los gastos de mantenimiento y consumo de potencia eléctrica se reducirán a los de una sola máquina en lugar de dos. Asimismo, se necesitará contratar una línea de acceso a Internet para cada conexión a Internet que tome parte en el diseño, por lo que esto añade una razón más a favor para motivar la unión de estos dos servicios. En contrapartida no se considera que existan claras desventajas que puedan fomentar la separación de ambos servidores, por lo que se recomienda la arquitectura implementada en el presente proyecto.

No obstante, todo el diseño desarrollado se ha realizado con la posibilidad de separar físicamente ambos servidores realizando sólo cambios en la configuración de los programas. Toda la documentación relativa a la instalación de las herramientas puede consultarse en el Anexo A de esta memoria, donde se recoge también una explicación sobre cómo podrían separarse los servidores.

Por otro lado, en el lado del cliente se ha diseñado una aplicación específica de Android para la reproducción de vídeos de YouTube al mismo tiempo que se monitorizan gran cantidad de parámetros. En esta aplicación es donde reside la clave del proyecto, ya que ha permitido integrar la filosofía de *crowdsourcing*

que se intenta imponer en el presente proyecto. Esta herramienta hará posible la recolección de datos de un modo totalmente gratuito a partir de la colaboración del usuario. En aras de conseguir un gran nivel de participación, esta aplicación se diseñará de un modo atractivo para el usuario y se tratará de minimizar en la medida de lo posible el coste de adaptación a ésta. Resulta también de gran relevancia realizar una encuesta lo más breve y concisa posible para favorecer que el usuario pueda completarlas sin realizar un gran esfuerzo.

Siguiendo la arquitectura propuesta para el diseño, cabe destacar las siguientes consideraciones implícitas del modelo:

- El servidor SFTP debe ser universalmente accesible desde cualquier lugar de Internet. Para ello será necesario disponer de una IP pública o de un dominio de acceso universal, de modo que cualquier dispositivo móvil pueda ponerse en contacto con dicho servidor.
- La dirección IP o dominio público del servidor SFTP debe ser permanente. Esta dirección se incluye en la configuración de la aplicación para móviles Android, por lo que un cambio de ésta supondrá que el envío de datos no se pueda producir satisfactoriamente. Cada vez que se realice un cambio sería necesario reconfigurar el acceso al servidor SFTP en la aplicación y volver a instalarla en todos los dispositivos móviles.
- Si el servidor SFTP y el de base de datos *mySQL* se encuentran físicamente en distintas máquinas, se debe garantizar la conectividad entre dichas máquinas. Asimismo, habría que comprobar que no exista en el camino intermedio de la conexión ningún cortafuegos que impida el acceso al puerto del servidor de base de datos (este puerto es el 3306 por defecto).
- No es necesario que el servidor de base de datos sea accesible desde cualquier máquina ajena a la del servidor SFTP, puesto que la única máquina que se comunicará con la base de datos será ésta. Se debe recordar que la información recopilada de los dispositivos móviles se almacena en el servidor SFTP, mientras que en este último servidor es donde se procesa la información y se inserta en la base de datos. Por ello, restringiendo el acceso a la base de datos desde el exterior se podría conseguir preservar la seguridad de los datos.

Otra alternativa que se consideró en la fase preliminar del diseño consistía en almacenar los datos en los dispositivos móviles en ficheros con estructura de base de datos. De este modo se simplificaría la implementación en el lado del servidor, ya que podría omitirse el servidor SFTP e insertar directamente los datos conectándose al servidor de base de datos. Sin embargo esta opción fue descartada posteriormente debido a que los datos monitorizados precisaban un posprocesado una vez que se finalizaba la reproducción del vídeo. Esto motivó que, para agilizar el proceso de envío de los datos hacia el servidor una vez rellenada la encuesta y reducir la carga computacional en el dispositivo móvil, se enviaran ficheros de texto que se iban rellenando al mismo tiempo que se monitorizaban los datos. Esto dio lugar a la incorporación de un servidor SFTP necesario para recibir estos ficheros que serían procesados posteriormente para ser insertados en la base de datos. Para el traslado de los datos desde los ficheros

de texto hacia la base de datos sería a su vez necesario el desarrollo un programa. Este programa se encargaría básicamente de leer la información de los ficheros y procesarla convenientemente para obtener todas las estadísticas que podrán encontrarse finalmente en la base de datos.

4.2. Desarrollo de la aplicación Android del cliente

En el presente epígrafe se van a describir todos los aspectos relacionados con el desarrollo de la aplicación Android que actuará como plataforma de recolección de datos basada en la filosofía de *crowdsourcing*.

Se tratará de realizar una documentación detallada para que el lector pueda comprender en profundidad el funcionamiento de la aplicación sin necesidad de tener que conocer el código utilizado propiamente para la aplicación.

Uno de los principales objetivos fijados sobre esta aplicación además de los meramente funcionales, es diseñar una interfaz limpia, sencilla e intuitiva para el usuario. Es un concepto que se destaca en varias ocasiones a lo largo de la memoria y que resulta de gran relevancia, ya que se trata de favorecer su utilización en la medida de lo posible. Por ello, todo el diseño de la aplicación se verá ampliamente influenciado por estos factores.

4.2.1. Introducción al diseño de la aplicación

En primer lugar, a modo de introducción al diseño, se va a describir de un modo sintético el funcionamiento de la implementación que se desea desarrollar y que tratará de cumplir todos los requisitos funcionales y no funcionales expuestos en el apartado 2.2 de esta memoria. En esta descripción se incluirán todos aquellos procesos que resultan fundamentales en la aplicación y se realizan de un modo transparente para el usuario.

La utilización de la interfaz por parte del usuario se encuentra debidamente documentada en el Anexo B de esta memoria, concretamente en el apartado B.2. En este anexo se indican todas las pantallas o actividades (término que se define en la documentación oficial de Android) de las que dispone la aplicación y cómo puede el usuario final interactuar con ella. Además se recoge un apartado con los posibles errores o problemáticas que podrían producirse junto con su diagnóstico.

La aplicación permitirá en primer lugar que el usuario pueda buscar cualquier vídeo en el servidor de YouTube. Para ello se escribirá una palabra o frase y se listarán los resultados encontrados. Alternativamente podrá pulsar desde cualquier otra aplicación del móvil sobre un enlace *web* que referencie un vídeo de YouTube para ejecutar directamente la reproducción del vídeo a través de esta aplicación.

Una vez elegido un vídeo, comienza el proceso de mayor interés en relación al proyecto. En ese momento se abrirá una nueva pantalla o actividad. En esta

pantalla, justo un momento antes de comenzar la reproducción, se ejecutarán tres hebras que realizarán un trabajo específico en segundo plano.

Estas tres hebras se encargarán de recopilar información de distinta naturaleza relacionada con la calidad de servicio (QoS) durante la reproducción del vídeo. En la siguiente lista se describe la utilidad de cada una de ellas:

- **Hebra 'InformacionVideo'**

  Recopilación de información relacionada con la descripción del vídeo. Se recogen datos como el título, su descripción, el número de reproducciones o los tópicos relacionados con el vídeo. Una vez que se recogen todos estos datos, serán almacenados y esta hebra dará por terminada su ejecución.

- **Hebra 'MonitorizacionRed'**

  Recopilación de datos referentes al estado de la red. Estos datos pretenden recoger una perspectiva global de la QoS durante la reproducción. La hebra ejecutada detectará en primer lugar si el tipo de conexión es Wi-Fi o es de la red móvil. Posteriormente recogerá un conjunto de información bastante completo en función del tipo de red. En caso de que se esté conectado a la red Wi-Fi para el acceso a Internet, podrán recogerse datos tanto referentes a la celda de la red móvil como de la red Wi-Fi. Sin embargo, si sólo se está conectado a la red móvil, se recogerán datos de la celda de la red móvil a la que se está conectado.

  Al mismo tiempo se ejecutan dos procesos que serán interrumpidos únicamente cuando el usuario decida finalizar la reproducción y salir de la actividad del reproductor. El primero de estos procesos se encargará de ir detectando cambios en la tecnología de red móvil, por ejemplo si se realiza un cambio de la tecnología GPRS a UMTS. Cuando esto ocurra se registrará con una marca temporal para poder conocer en qué momento y durante cuánto tiempo se ha utilizado cada tecnología. El segundo de estos procesos se encargará de ir realizando mediciones de nivel de señal (RSSI) tanto de Wi-Fi (si está activado) como de la cobertura red móvil. Estas mediciones se tomarán asociadas a la localización del teléfono móvil en ese momento y se realizarán cada vez que se detecte una nueva posición. Con ello se podrá conocer el nivel de señal, la tecnología de la celda móvil, la localización y el operador móvil en cada medida.

- **Hebra 'MonitorizacionReproductor'**

  Recopilación de datos relacionados con la QoS a nivel de aplicación. Para ello la hebra actuará como un capturador de eventos en segundo plano cuya ejecución se extenderá a lo largo de toda la aplicación. Así, se podrán detectar los eventos de carga inicial del vídeo, interrupción, reanudación, inicio del proceso de *buffering*, finalización del proceso de *buffering* y algunos otros eventos que permitirán obtener información relevante sobre la QoS tras un procesado de los datos.

En este resumen de las tres hebras se recopilan algunos otros datos que no han sido mencionados pero que por el momento no son de gran relevancia para el

lector, ya que en el apartado 4.4 se detallan todos los datos que se obtienen finalmente en la base de datos tras un procesado de toda la información recopilada conjuntamente.

El funcionamiento básico de estas tres hebras se basa en un mismo mecanismo. Al inicio de cada una de ellas se genera un fichero de texto independiente al de las otras y con el mismo nombre que tiene la hebra (excepto el de la hebra 'InformaciónVideo', que se denomina '*DescripciónVideo.txt*'). Este fichero de texto se rellenará con un formato predefinido para facilitar el procesado de la información posteriormente en el servidor. Para ello se ha decidido generar una especie de fichero de registro, también denominado comúnmente fichero de tipo *log*. En este fichero se irán añadiendo líneas con distinta información conforme se va recogiendo. De este modo, en el caso de las hebras 'MonitorizacionRed' y 'MonitorizacionReproductor', cada línea se iniciará con una marca temporal (tomada como el tiempo de sistema en Android) y seguidamente incluirá un dato. Esto permitirá gestionar los tiempos en los que se producen diferentes eventos o se realizan mediciones, lo que tendrá también su utilidad posteriormente en el posprocesado de los datos. Así en la Figura 4.2.1 se presenta un ejemplo de un fichero de la hebra 'MonitorizacionRed', donde se observa información recopilada sobre la red.

```
Estadisticas de red:
1406903938044 -> Conectado via WiFi
1406903938046 -> SSID: [redacted]
1406903938046 -> BSSID: [redacted]
1406903938046 -> Ancho de banda teórico(Mbps): 65
1406903938046 -> dir. IP: [redacted]
1406903938124 -> Conectado a celda Wcdma
1406903938125 -> Mcc = 214
1406903938125 -> Mnc = 85
1406903938126 -> Lac = 1804
1406903938126 -> Cell ID = 7764823
1406903938126 -> Tipo de conexion: UMTS(3G)
1406903938126 -> Operador: ONO
                    ⋮
```

Figura 4.2.1: Ejemplo de fichero '*MonitorizacionRed.txt*'

Una vez finalizada la reproducción del vídeo y cuando el usuario decida salir de dicha actividad, pueden ocurrir tres casos distintos:

- Si el usuario pulsa el botón 'atrás' propio del sistema operativo Android y se ha reproducido un tiempo igual o superior al tiempo umbral configurado, se le invitará a realizar una encuesta.
- Si el usuario pulsa el botón 'atrás' y se ha reproducido un tiempo inferior al tiempo umbral configurado, se volverá a la actividad anterior y se desechará toda la información recogida.
- Si el usuario sale de la aplicación mediante el botón '*home*' propio del sistema operativo Android o mediante cualquier otro método distinto al botón 'atrás', se volverá a la actividad anterior y se desechará toda la información recogida.

La utilización del umbral pretende conseguir que los datos recogidos tengan cierta fiabilidad y para ello se trata de garantizar que el usuario al menos haya reproducido este tiempo umbral antes de realizar la encuesta y enviar los datos al servidor. Todos los aspectos relativos a la configuración de este umbral pueden consultarse en el Anexo A de esta memoria, concretamente en el apartado A.2. Por defecto este valor ha sido fijado a treinta segundos.

En el caso de que se invite al usuario a realizar una encuesta, éste podrá aceptar y se le enviará a una nueva actividad en la que aparecerá un formulario que se debe rellenar. Este formulario es bastante breve y rápido de completar pero se considera que contiene datos de bastante interés. Estos datos son:

- Edad del usuario
- Sexo del usuario
- Valoración general del usuario (entre 1 y 5)

Los dos primeros datos servirán para almacenar datos de carácter demográfico, que podrán utilizarse en posibles estudios en los que tomen parte este tipo de variables. El tercero de ellos quizás es el de mayor relevancia en lo que al proyecto se refiere. Con esta simple valoración de 1 a 5, se trata de realizar una medición del MOS (*Mean Opinion Score*) [10]. Este medidor que ya ha sido mencionado con anterioridad en el capítulo 1, es uno de los que se utilizan con mayor asiduidad para la evaluación de la QoE, es decir, de la calidad que ha experimentado el usuario al visualizar el vídeo.

Esto último permitirá realizar una asociación entre el nivel de QoE evaluado por el usuario y los diferentes parámetros de QoS a nivel de red y a nivel de aplicación recopilados.

Por último, una vez que se completa la encuesta, estos datos se incluirán en un nuevo fichero de texto, por lo que en total se tienen cuatro ficheros con la información recopilada. Tres de ellos serán correspondientes a las hebras descritas arriba y un último fichero correspondiente a la encuesta.

Al finalizar la encuesta se le proporcionará al usuario la posibilidad de enviar los datos al servidor. Para ello se utilizará una conexión a un servidor SFTP. Con el protocolo SFTP se permitirá que se puedan enviar de una forma segura al servidor los cuatro ficheros que se han generado. Antes del envío, en el título de los cuatro ficheros se añadirá al inicio un identificador obtenido mediante la fecha y hora de envío y un identificador único del dispositivo móvil, con lo que se podrá diferenciar entre los ficheros que han sido enviados en cada experimento.

El hecho de que los ficheros identifiquen a un experimento concreto no está relacionado con que se pueda identificar el dispositivo móvil desde el que se ha enviado. Esto se debe a que en el algoritmo utilizado para la consecución del identificador único, se utiliza una función *hash* que no permite obtener el identificador mediante ingeniería inversa. Este mecanismo se ha diseñado con el fin de preservar la privacidad del usuario tal y como se planteaba en los objetivos del proyecto, ya que los datos son de carácter anónimo.

En la Figura 4.2.2 se puede observar el directorio SFTP con los cuatro ficheros relativos a un mismo envío. Como se puede observar todos ellos comienzan con

un mismo identificador compuesto por fecha, hora y el identificador único y posteriormente se indica de cuál de los cuatro ficheros se trata. Esto permitirá que al utilizar el programa que lleva a cabo automáticamente la lectura de los ficheros, se puedan identificar los ficheros de un mismo experimento con facilidad para insertarlos ordenadamente en la base de datos.

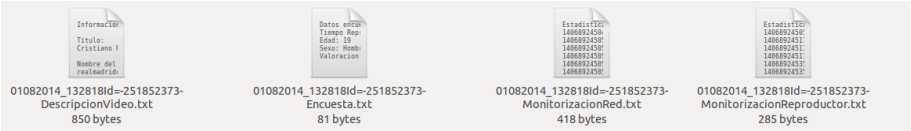

Figura 4.2.2: Ficheros enviados al servidor

Modos de la aplicación

Una vez descrito todo el proceso de diseño, se van a describir las posibles modalidades de la aplicación que se han contemplado. La aplicación consta en total de cuatro pantallas o actividades diferentes. Sin embargo, se podrá configurar con distintos modos que darán lugar a que el comportamiento y la interfaz de algunas actividades se modifiquen.

Los modos diseñados son los siguientes:

- **Modo '*debug* en la actividad del reproductor'**

  Dentro de la actividad en la que se reproduce el vídeo, se podrá observar en diferentes cuadros de texto toda la información que se está monitorizando. En concreto habrá dos cuadros de texto. En uno de ellos se expone la información referente al fichero de la hebra 'MonitorizacionRed' y en el otro el de la hebra 'MonitorizacionReproductor'. Estos cuadros de texto disponen de una barra para deslizarse a lo largo de toda la información contenida en el fichero. En ellos se podrá comprobar la estructura de registro, en el que todas sus líneas están conformadas por una marca temporal seguida de un mensaje o dato.

  Este modo servirá para realizar tareas de depuración durante el periodo de desarrollo de la aplicación. Además puede resultar útil para el usuario que desee conocer en mayor profundidad el funcionamiento de la aplicación. De esta manera se podrá observar en tiempo real y de un modo didáctico la información que se va añadiendo ante diferentes eventos que se van produciendo a lo largo de la reproducción de un vídeo.

  En la Figura 4.2.3 se muestra un ejemplo de la interfaz de la actividad del reproductor cuando este modo se encuentra activo. En ella se puede observar debajo del vídeo, el fichero de 'MonitorizacionReproductor', e inmediatamente debajo el de 'MonitorizacionRed'.

- **Modo '*debug* en la actividad de la encuesta'**

  Este modo se aplicará sobre la actividad en la que se realiza una encuesta al usuario. El comportamiento difiere de su funcionamiento normal en el

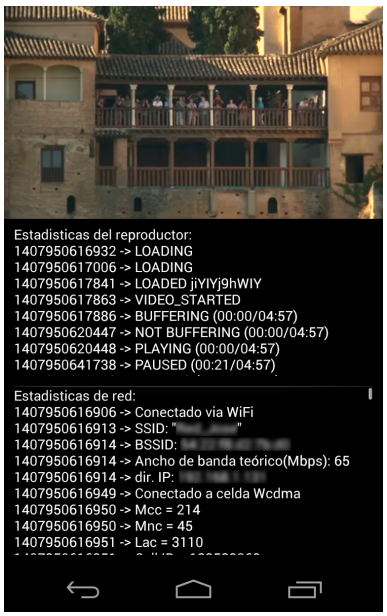

Figura 4.2.3: Interfaz del modo *debug* en la actividad del reproductor

momento en el que el usuario ha rellenado la encuesta y pulsa el botón para enviar los datos al servidor. En este momento, en lugar de producirse el envío hacia el servidor, aparecerá una pantalla con toda la información que se ha recogido a lo largo del experimento completo. Con ello podrá observarse el contenido de los cuatro ficheros que se han descrito anteriormente.

Se trata de un modo creado para comprobar todos los datos que han sido recopilados y que se enviarían hipotéticamente al servidor. Por lo tanto, este modo no se debe activar a menos que se utilice en tareas de desarrollo y validación del funcionamiento de la aplicación, ya que los datos no serán enviados al servidor y no podrán ser por tanto recogidos en la base de datos.

En la Figura 4.2.4 se muestra a modo de ejemplo una captura de pantalla de la interfaz de usuario cuando se utiliza este modo. Se observa un extracto de la información recopilada una vez finalizada la encuesta. Puesto que habitualmente se trata de ficheros bastante extensos, será necesario deslizar el cuadro de texto para poder observar todo el contenido.

Estos dos modos podrán configurarse previamente a la instalación, mediante la modificación del código de la aplicación. Únicamente será necesario modificar dos variables de tipo *booleano* que indicarán si se activan (*true*) o no (*false*) estos modos. Cabe destacar que ambos son compatibles en todas sus combinaciones posibles puesto que afectan a distintas pantallas o actividades de la aplicación.

No obstante, la versión final contemplada para el cliente se definirá mediante el modo por defecto de la aplicación. Este modo simplemente se podrá configurar desactivando los dos modos descritos arriba. Se recomienda distribuir al cliente

Figura 4.2.4: Interfaz del modo *debug* en la actividad de la encuesta

la aplicación en este modo, ya que es aquel en el que más se cuida la estética de la interfaz además de que la monitorización se realiza de un modo totalmente transparente. Todo ello facilitará que el cliente utilice la aplicación como si se tratase de un reproductor corriente de vídeos.

Para más información sobre cómo configurar estos modos, en el apartado A.2 de esta memoria se recoge toda la documentación referente al proceso de configuración e instalación de la aplicación Android.

4.2.2. Descripción de la implementación

En el presente apartado se va a realizar una descripción en términos generales de la implementación realizada para la aplicación de Android que llevará a cabo la reproducción de vídeos y recopilación de datos.

El desarrollo de la implementación requiere de una formación previa específica para la programación en Android, ya que se trata de una filosofía de programación que difiere de los lenguajes convencionales orientados a objetos, tales como Java. De un modo bastante resumido, en Android la programación puede dividirse en tres grandes bloques:

- Por un lado se generarán una serie de ficheros Java en los que se desarrollará toda la parte lógica de la aplicación. Es decir, se gestionará el funcionamiento de cada una de las pantallas o actividades y todos aquellos procesos debidos a la actuación del usuario sobre un elemento de la interfaz o simplemente procesos que se ejecutan de un modo programado. Todos estos ficheros conforman un proyecto con características sintácticas

muy similares a las de un proyecto convencional realizado con el lenguaje de programación Java. La principal diferencia que reside en la programación en Android, es que se utilizan librerías específicas del SDK (*Software Development Kit*) de Android generadas la mayoría a partir de adaptaciones de librerías propias de Java. Además existen una serie de métodos intrínsecos de la interfaz gráfica de Android que definen por ejemplo una rutina al iniciar una nueva pantalla (método *OnCreate()*) o *listeners* que se encargan de capturar una gran diversidad de eventos posibles.

Este conjunto de ficheros con extensión *java* se encuentran almacenados en el directorio '*/src*' dentro del proyecto.

- Por otro lado, en el diseño tomarán parte una serie de ficheros XML de gran relevancia. En estos ficheros se describirán todos los aspectos visuales de la aplicación. Recogerán toda la información necesaria para describir el aspecto visual de todas las pantallas o actividades de la aplicación así como de algunos elementos visuales concretos. Todo ello dotará de gran libertad al programador para que éste pueda implementar sus propios diseños. Estos ficheros XML serán referenciados en los ficheros desarrollados en Java, donde se describirá el funcionamiento de los diferentes elementos visuales que constituyen el diseño.

  Todos estos ficheros se encuentran alojados en el directorio '/res/layout' del proyecto.

- Por último, se incluye un fichero XML especial denominado 'AndroidManifest'. Este fichero es una especie de contrato o manifiesto de la aplicación tal y como su nombre indica. En éste se recogen aspectos como el nombre de la aplicación, la versión mínima de Android que soporta, los permisos especiales que se necesitan, las distintas actividades o pantallas de las que dispone y sus respectivos nombres, etcétera.

  Este fichero puede encontrarse en el directorio raíz del proyecto.

Además, existen algunas restricciones también propias del sistema operativo Android y relacionadas con la interfaz táctil del teléfono. Por ejemplo, una restricción que se ha puesto de manifiesto en multitud de ocasiones a lo largo del desarrollo de la aplicación, es que cualquier tarea o método que implique un acceso a Internet o conexión a un servidor remoto no puede ejecutarse en un código correspondiente a alguna de las interfaces de usuario o actividades de la aplicación. Este tipo de procesos deberán ejecutarse típicamente en hebras que actúen en segundo plano o, en su defecto, en clases adicionales que se ejecuten independientemente del código de la interfaz de usuario. Con esta restricción se trata de evitar bloqueos en la interfaz gráfica debidos al procesado de actividades en red, que habitualmente son tareas que tienen un tiempo de ejecución difícil de acotar ya que están ligadas a las condiciones instantáneas de la red y pueden verse afectadas por fallos de conectividad. En lo que se refiere a la realización de este proyecto, esta restricción se ha podido salvar fácilmente mediante la ejecución de hebras de tipo *AsyncTask* [48] propias del SDK de Android. Estas hebras son relativamente sencillas de programar y permiten interactuar en momentos puntuales con la interfaz gráfica para realizar algún tipo de tarea que afecte a ésta.

Descripción de la parte lógica de la aplicación

A continuación, una vez que se han descrito los tres bloques principales que constituyen una aplicación, se procederá a describir el código debido a la parte lógica de la aplicación. Esta parte corresponde al primero de los bloques descritos arriba, donde realmente se describe el funcionamiento de la aplicación y todos los procesos que permiten la reproducción de vídeos y la recolección de información.

En Android, las clases que implementan una actividad o interfaz de usuario típicamente extienden la clase *Activity* [49]. Esta clase contiene una serie de métodos que gestionarán distintos comportamientos derivados de la interacción del usuario con la interfaz táctil. Entre ellos, algunos que se han utilizado de forma recurrente en el desarrollo de la aplicación son:

- *protected void onCreate(Bundle savedInstanceState)*

  Este método se debe utilizar obligatoriamente en todas las clases que correspondan al comportamiento de una interfaz de usuario o actividad. En él se describe la rutina que se realizará al iniciarse dicha actividad. Entre los procesos que se pueden encontrar en este método, se incluye una referencia al fichero XML en el que se describe el diseño visual de la interfaz.

- *public boolean onCreateOptionsMenu(Menu menu)*

  Este método inicializa el menú de opciones propio de las interfaces del sistema operativo Android.

- *public boolean onOptionsItemSelected(MenuItem item)*

  Se define el comportamiento al pulsar una de las opciones del menú de opciones de Android.

- *public void onBackPressed()*

  Se define el comportamiento cuando el usuario pulsa el botón 'atrás' incorporado en todos los sistemas Android.

- *public void onPause()*

  Este método se ejecuta cuando una actividad pasa a estar en segundo plano pero no se ha destruido aún.

- *protected void onRestart()*

  Método que se ejecuta cuando una actividad que había sido reemplazada por una nueva que se ha ejecutado encima, vuelve a mostrarse por pantalla debido a que la nueva se ha finalizado.

Por otro lado, como se ha indicado con anterioridad, todas las hebras implementadas se han programado mediante la utilización de la clase *AsyncTask* [48]. Para ello se han creado clases que extienden dicha clase. Con ello, se podrán utilizar entre otros, los siguientes métodos que convierten en sencilla y flexible la programación de la hebra:

- *protected void onPreExecute()*

  Este método será el primero en ejecutarse al crearse la hebra. Se podrán realizar acciones que afecten a la interfaz de usuario inmediatamente antes de llevar a cabo el proceso en segundo plano.

- *protected abstract Result doInBackground(Params... params)*

  Este método se ejecuta a continuación del anterior. Dentro de esta rutina que actúa en segundo plano no se pueden realizar acciones que afecten directamente a la interfaz de usuario. Para ello se puede ejecutar cuando se desee el método *publishProgress(Progress... Values)*, que llamará a la rutina que se defina en el método *onProgressUpdate(Progress... Values)*.

- *protected void onProgressUpdate(Progress... Values)*

  Este método será llamado tantas veces como se desee desde el método *doInBackground(Params... params)* y típicamente se encargará de realizar procesos que afecten a la interfaz de usuario y que no puedan por tanto realizarse en segundo plano. La llamada a este método se realiza mediante el método *publishProgress(Progress... Values)*.

- *protected void onPostExecute(Result result)*

  Este método se ejecutará al finalizar el método *doInBackground(Params... params)* y típicamente se utilizará para publicar los resultados obtenidos tras realizar las tareas en segundo plano. Permite realizar acciones que afecten a la interfaz de usuario.

Seguidamente se incluye una recopilación de las clases Java que componen el proyecto tratando de explicar las relaciones existentes entre éstas para comprender cómo finalmente funciona la aplicación en su conjunto.

Las ficheros Java que constituyen este proyecto son:

- **DescriptoresAplicacion.java**

  En esta clase se recogen una serie de variables referentes a la configuración de la aplicación. En ella se pueden ajustar algunos parámetros y personalizar la aplicación activando o desactivando sus diferentes modalidades. Asimismo, se incluyen algunas variables que habrá que modificar convenientemente para el acceso satisfactorio al servidor SFTP. En el código de esta clase se describe el significado de cada uno de los atributos que contiene. No obstante se podrá consultar el Anexo A de esta memoria, en el apartado A.2, donde se desarrolla de forma detallada un manual de instalación de la aplicación de Android incluyendo el significado de estos atributos.

- **ActividadInicial.java**

  Se trata de la primera clase desarrollada en el proyecto, es decir, aquella en la que se define el comportamiento de la primera actividad o pantalla que

se ejecuta en la aplicación. En esta clase se define el proceso de leer desde un campo de texto (objeto *EditText* [50]) aquella palabra o frase sobre la que el usuario desea buscar. Además se recoge el máximo número de resultados (mediante un objeto *NumberPicker* [51]) que el usuario desea buscar. Una vez que el usuario ejecute la búsqueda de vídeos, se llamará a una nueva actividad correspondiente a la clase '*ResultadosBusqueda*', a la que se le enviarán estos datos.

Todo este proceso descrito se implementa en el método *Buscar(View view)* , que se trata de un *listener* que se ejecutará cuando el usuario pulse el botón *Buscar* de la interfaz.

En la Figura 4.2.5 se puede observar la interfaz cuyo comportamiento se define en esta clase. La composición visual de ésta se define en el fichero '*/res/layout/actividad_inicial.xml*' del proyecto.

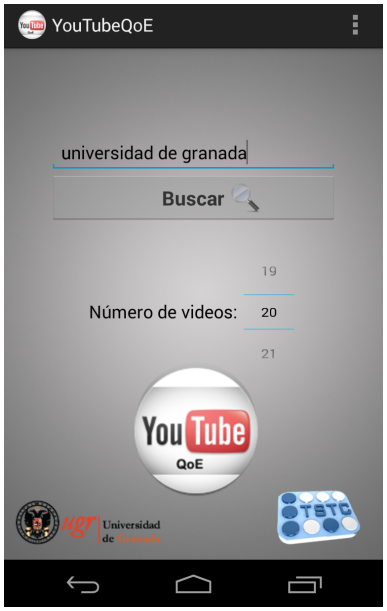

Figura 4.2.5: Interfaz correspondiente a la clase '*ActividadInicial*'

- **ResultadosBusqueda.java**

  En esta actividad se recogen los términos de búsqueda recibidos desde la actividad correspondiente a la clase '*ActividadInicial*' a partir de un objeto denominado *Bundle* [52]. Una vez que se tienen estos datos almacenados, se ejecuta una hebra denominada '*BusquedaYoutube*' que se ha implementado como una clase interna dentro de esta misma clase. En esta hebra se ejecutará en segundo plano la solicitud de búsqueda al servidor de YouTube. Mientras se ejecuta esta búsqueda se despliega un diálogo que indica al usuario el progreso de la búsqueda y, al recibir los resultados, se listan éstos de forma ordenada indicando el título, una imagen asociada al vídeo y la fecha de publicación de éste. La disposición visual de cada elemento de la lista viene descrito en el fichero '*/res/layout/elemento_lista.xml*'.

Esta misma hebra contiene un método de tipo *listener* que se encargará de capturar el evento de que se pulse sobre alguno de los vídeos de la lista. En el momento en el que se produce dicho evento, se abrirá una nueva actividad correspondiente al código recogido en la clase '*Reproductor*' y se le enviarán algunos datos de la descripción del vídeo (identificador único del vídeo, título, descripción, fecha de publicación y canal de YouTube).

La interfaz cuya lógica se define en esta clase queda ilustrada en la Figura 4.2.6, donde se observa un ejemplo de los resultados obtenidos en una búsqueda. Asimismo, se podrá analizar la disposición visual de la interfaz consultando el contenido del fichero '*/res/layout/actividad_resultados_busqueda.xml*'.

Figura 4.2.6: Interfaz de resultados de la búsqueda de la aplicación

- **DescriptorVideo.java**

  Se trata de una clase auxiliar de la clase '*ResultadosBusqueda*' creada para almacenar toda la información relativa a un vídeo. Esta clase almacena los siguientes datos del vídeo:

  - Identificador único del vídeo
  - Título
  - Fecha de publicación
  - Imagen descriptiva
  - Descripción resumida
  - Canal de YouTube

- **Reproductor.java**

  Esta clase corresponde a la actividad de mayor relevancia de la aplicación. En esta clase se define todo el comportamiento relativo a la actividad donde se reproducen los vídeos.

  En primer lugar, se define la interfaz visual que se va a ofrecer. Ésta dependerá de si el modo '*debug* en la actividad del reproductor' se encuentra activo o no. En el caso se encontrarse activo, se utilizará la disposición visual descrita en el fichero '*/res/layout/actividad_reproductor_debug.xml*', en caso contrario se referenciará la composición definida en '*/res/layout/actividad_reproductor.xml*'.

  En la Figura 4.2.7 se observa la interfaz de esta actividad cuando el modo '*debug* en la actividad del reproductor' se encuentra desactivado, es decir, la interfaz contemplada para el usuario final.

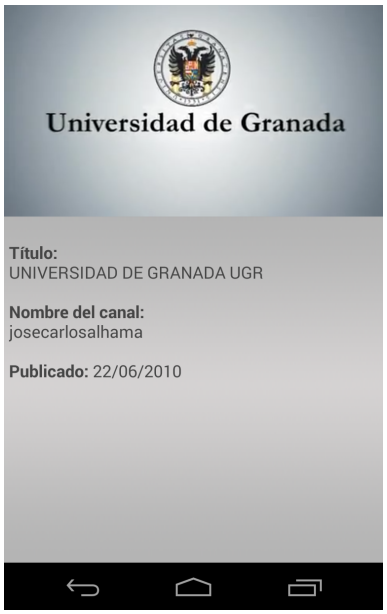

Figura 4.2.7: Interfaz del reproductor de la aplicación

En la Figura 4.2.3 se puede observar la interfaz cuando el modo '*debug* en la actividad del reproductor' se encuentra activo.

A continuación se almacenan todos los datos de la descripción del vídeo recibidos desde la clase '*ResultadosBusqueda*'. Estos datos permitirán enviar una solicitud al servidor de YouTube para comenzar el *streaming* del vídeo a partir de su identificador único.

Una vez que se inicializa el objeto del reproductor, se ejecutará el método '*onInitializationSuccess()*'. En este método se ejecutarán varios procesos esenciales en el desarrollo de la aplicación. En concreto se inician por orden las hebras '*MonitorizacionReproductor*', '*MonitorizacionRed*' e '*InformacionVideo*'. Estas hebras se describirán seguidamente puesto que se trata de otras clases que constituyen el proyecto. Posteriormente se comenza-

rá la carga del vídeo para iniciar su reproducción una vez que todos los mecanismos de monitorización se han ejecutado.

Otro método importante de esta misma clase es '*doLayout()*', que se encargará de componer la interfaz de la actividad en función de si el dispositivo móvil se encuentra en posición vertical u horizontal. De este modo, se detectará cuándo se produce un cambio de orientación y se mostrará por pantalla la interfaz correspondiente a la posición vertical (Figura 4.2.7 o Figura 4.2.3) u horizontal (Figura 4.2.8).

Figura 4.2.8: Interfaz del reproductor de la aplicación en posición horizontal

Por último cabe destacar el método '*onBackPressed()*', en el que se gestiona el comportamiento cuando el usuario pulsa el botón de 'atrás' propio del sistema operativo Android. En este método se realiza una comprobación sobre si el tiempo de reproducción supera el umbral de tiempo establecido como se había explicado en el apartado 4.2.1 de esta memoria. Si se ha superado este tiempo umbral, se le invitará mediante un diálogo a realizar la encuesta. Para ello se llamará a una nueva actividad cuya lógica se define en la clase '*Encuesta*'.

- **ReproductorFromURL.java**

  Se trata de una clase con un código muy similar en su mayoría al de la clase '*Reproductor*'. Esta clase se ha generado con el fin de poder reproducir vídeos a partir de enlaces *web*. La principal diferencia con la anterior clase reside en que al principio del método *onCreate()* de esta clase, se captura el enlace *web* para poder obtener el identificador del vídeo. De este modo se podrá comenzar con la reproducción de éste habiendo ejecutado previamente las tres hebras de monitorización, tal y como ocurre en la clase anterior.

  Esta clase contiene una hebra denominada '*BusquedaVideoFromURL*' que se encarga de realizar una solicitud al servidor de YouTube para recoger información sobre el vídeo a partir de su identificador único. Los datos recopilados son los mismos que se incluyen en la clase '*DescriptorVideo*':

  - Título
  - Fecha de publicación
  - Imagen descriptiva

- Descripción resumida
- Canal de YouTube

Las interfaces correspondientes a esta actividad son las mismas que las definidas para la clase '*Reproductor*'.

- **MonitorizacionRed.java**

  Esta clase implementa una hebra y se ejecuta en las clases '*Reproductor*' y '*ReproductorFromURL*' antes de comenzar la carga y reproducción del vídeo. Su cometido consiste en realizar en segundo plano una recolección de datos referentes a la QoS a nivel de red. Para ello se irá rellenando un fichero de texto denominado '*MonitorizacionRed.txt*' con todos los datos recopilados.

  Esta hebra al inicio comienza activando un objeto de tipo '*LocationManager*' [53] que hace uso de un método de tipo *listener* que permitirá detectar cuándo se produce un cambio en la ubicación del dispositivo móvil. Esto permitirá que continuamente se vayan realizando medidas del RSSI en la red móvil y en la red Wi-Fi (si está activada). Cada una de estas medidas se agregará junto con la localización del dispositivo en ese momento.

  Seguidamente, a partir de un objeto '*ConnectivityManager*' [54] se comprobará si el móvil está conectado a la red Wi-Fi y/o a la red móvil. En caso de que esté conectado a la red Wi-Fi, se utilizará un objeto de tipo '*WifiInfo*' [55] para recopilar toda la información posible sobre los parámetros de esta red. Si se está conectado a la red móvil para acceso a Internet se utilizará un objeto '*TelephonyManager*' [56], que permitirá recopilar datos sobre esta red. Asimismo se recopilarán una serie de datos referentes a la celda móvil a la que el terminal móvil está asociado en ese momento.

  Por último, en caso de que el dispositivo esté conectado a Internet a través de la red móvil, se llevará a cabo un sondeo periódico de la tecnología móvil (GSM, UMTS, LTE...) utilizada hasta la finalización de la reproducción del vídeo. Esto permitirá conocer en qué momento se está utilizando cada tecnología de modo que se puedan posteriormente analizar el resto de datos asociándolos a las diferentes tecnologías. Por ejemplo, se podrá descubrir si se producen más interrupciones en el vídeo debidas a congestión en la red cuando se utilizan tecnologías que ofrecen un menor ancho de banda.

- **MonitorizacionReproductor.java**

  Esta hebra se ejecuta junto con la anterior en las clases '*Reproductor*' y '*ReproductorFromURL*' antes de comenzar la carga y reproducción del vídeo. Se encargará de recopilar en segundo plano una serie de datos referentes a la QoS a nivel de aplicación y de almacenarlos en un fichero denominado '*MonitorizacionReproductor.txt*'. Básicamente se capturan todos los eventos posibles del reproductor para después en el posprocesado de los datos obtener estadísticas que resultarán de interés.

  Para capturar los eventos, esta clase hará uso de la librería '*YouTube Android Player API*' [45] descrita en el apartado 4.2.3 de esta memoria. Esta librería ofrece las interfaces '*PlayBackEventListener*' [57] y '*PlayerStateChangeListener*' [58] que incluyen una serie de métodos útiles. Cada uno

estos métodos se ejecutará al producirse un evento concreto y permitirá añadir en el fichero de texto '*MonitorizacionReproductor.txt*' información sobre el evento ocurrido junto con una marca temporal.

- **InformacionVideo.java**

  Esta clase implementa una tercera hebra que se ejecutará junto con las dos anteriores al inicio de la reproducción y cuyo objetivo será añadir información sobre el vídeo que se va a reproducir. La información recopilada se añadirá al fichero '*DescripcionVideo.txt*' junto con la anteriormente recopilada en este mismo fichero que contenía los atributos de la clase '*DescriptorVideo*'. Se trata de datos relativos a estadísticas más específicas de los usuarios de YouTube y del vídeo que se está reproduciendo. Los datos que se recogen en esta hebra son:

  - Definición del vídeo: hd (*High Definition*), sd (*Standard Definition*)...
  - Duración del vídeo.
  - Descripción ampliada del vídeo: Se trata de una descripción más extensa que la recopilada anteriormente en la clase '*DescriptorVideo*'.
  - Número total de reproducciones del vídeo.
  - Número de 'Me Gusta' de los usuarios de YouTube.
  - Número de 'No Me Gusta' de los usuarios de YouTube.
  - Tópicos del vídeo: Se obtendrán todos los tópicos relacionados con el vídeo diferenciando aquellos que son principales y aquellos que son secundarios. Para ello se hará uso de la base de datos *FreeBase* [59], que consiste en un directorio en el que se definen tópicos de forma ordenada y con identificadores únicos. La motivación del uso de esta base de datos consiste en reunir todos los tópicos en una base de datos específica y que es utilizada en gran cantidad de servicios distintos en Internet. Por ello se tratará de tópicos definidos con los que se podrá interactuar en multitud de aplicaciones distintas.

- **DeverloperKey.java**

  Se trata de una clase auxiliar que contiene dos variables de tipo estático correspondientes a dos claves de desarrollador de *Google Developers* para el uso de las APIs de YouTube [60]. Estas claves serán necesarias cada vez que se desee realizar una petición de un vídeo o de estadísticas al servidor de YouTube.

  Estas claves son utilizadas en las clases '*ResultadosBusqueda*', '*Reproductor*', '*ReproductorFromURL*', '*InformacionVideo*' y '*YouTubeFailureRecoveryActivity*'.

- **Encuesta.java**

  Esta clase define el comportamiento referente a la actividad en la que se ofrece un cuestionario al usuario para que lo rellene y envíe todos los datos al servidor. Para ello se utilizan un conjunto de elementos visuales adaptados a cada uno de los datos a rellenar que conforman la interfaz representada en la Figura 4.2.9. La composición visual de esta actividad se describe en el fichero '*/res/layout/actividad_encuesta.xml*'.

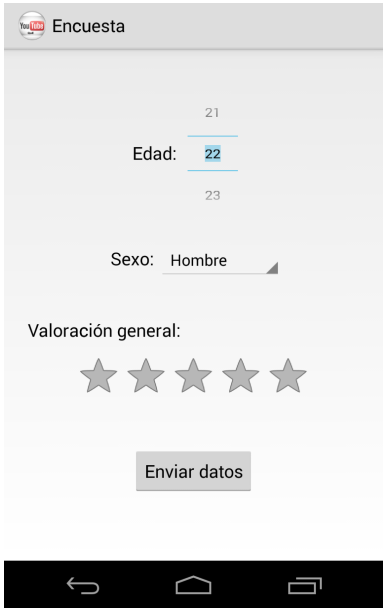

Figura 4.2.9: Interfaz de la encuesta al usuario de la aplicación

Los datos recopilados en este formulario son:

- Edad
- Sexo
- Valoración general del usuario

Los datos de edad y sexo serán almacenados en un fichero de preferencias de modo que en la siguiente ocasión en la que se vaya a realizar una encuesta, se muestren por defecto los últimos valores que se enviaron, todo ello para agilizar el proceso de rellenar la encuesta. En cuanto a la valoración general, se ha incluido una restricción por la cual no se permite el envío de los datos sin que el usuario previamente haya modificado este campo. Con ello se evita que se envíe la encuesta sin haberla rellenado por completo.

En la interfaz de esta actividad se incluye un botón en el que se puede leer 'Enviar datos', que permitirá que el usuario efectúe el envío hacia el servidor de todos los datos recogidos a lo largo del experimento. Al pulsar este botón se ejecutará el método *Enviar(View view)* en el que se almacenarán todos los datos de la encuesta y se procederá a ejecutar la hebra 'EnvioServidor' para llevar a cabo el envío de los cuatro ficheros generados hacia el servidor. En el caso de que se encuentre activo el modo '*debug* en la actividad de la encuesta', en lugar de ejecutar esta última hebra se llamará a la actividad cuya lógica se define en la clase '*ResultadosEncuestaDebug*'.

- **EnvioServidor.java**

  Se trata de una clase que implementa una hebra que realizará en segundo plano la conexión al servidor SFTP y el envío de los cuatro ficheros generados a lo largo del experimento.

  Antes de llevar a cabo el envío, el título de cada uno de los ficheros se modifica incluyéndole al inicio una cadena de caracteres conformada a partir de la fecha y hora de envío y un identificador único del dispositivo móvil. Con ello se podrán clasificar en el servidor los ficheros correspondientes a cada experimento concreto. El identificador del dispositivo móvil se obtiene a partir del método *getdeviceID()* de la clase '*TelephonyManager*' [56] de Android. No obstante, al identificador se le aplica posteriormente una función *hash* con el fin de ocultar la identidad del usuario final.

  Para el proceso de conexión, envío y desconexión del servidor SFTP se hace uso de la API 'JSch (*Java Secure Channel*)' [61] que permite abstraerse considerablemente de estos procesos.

- **ResultadosEncuestaDebug.java**

  Esta clase se utiliza solamente cuando el modo '*debug* en la actividad de la encuesta' se encuentra activo. Se trata de una clase en la que se describe el comportamiento de la actividad en la que se muestran todos los datos que serían enviados al servidor al finalizar la encuesta. El hecho de que se acceda a esta interfaz implica que los datos no se han enviado al servidor y en lugar de ello el usuario podrá observar en la pantalla toda la información que ha sido recopilada.

  La interfaz referenciada en esta clase es la que se muestra en la Figura 4.2.4 y su aspecto visual se describe en el fichero '*/res/layout/actividad_resultados_encuesta_debug.xml*'.

- **YouTubeFailureRecoveryActivity.java**

  Se trata de una clase auxiliar que tiene una serie de métodos que gestionarán posibles fallos en el reproductor de YouTube que se ha embebido en las actividades de las clases '*Reproductor*' y '*ReproductorFromURL*'. En caso de que se produzca un fallo, esta clase hará que se muestre por pantalla el tipo de error que ha ocurrido.

Cabe destacar que existen algunas clases internas dentro de algunas de estas clases descritas que complementan su funcionamiento. Sin embargo, se trata de clases auxiliares cuya aportación no es demasiado relevante, por lo que no se describen en la presente memoria.

Descripción del fichero 'AndroidManifest.xml'

Para finalizar con la descripción de la implementación de la aplicación Android, se describirá en términos generales el contenido del fichero 'AndroidManifest.xml'.

En este fichero se incluyen los siguientes aspectos entre otros:

- Versión mínima de Android soportada:

  La versión mínima de Android que soporta esta aplicación es la 3.0 (*Honeycomb*). Esta restricción se debe a que se necesita la API nivel 11 de Android [62] para implementar algunas de las funciones clave de esta aplicación.

  En la Figura 4.2.10 se presenta la línea del fichero XML en la que se aplica esta restricción.

```
android:minSdkVersion="11"
```

Figura 4.2.10: Versión mínima en el fichero 'AndroidManifest.xml'

- Permisos necesarios para el funcionamiento de la aplicación:

  - Permiso para acceder a Internet.
  - Permiso para acceder a información sobre el estado de las redes.
  - Permiso para acceder a información sobre las redes Wi-Fi.
  - Permiso para obtener la localización precisa mediante GPS, celdas móviles o Wi-Fi.
  - Permiso para obtener la localización aproximada derivada de elementos de la red como celdas móviles y Wi-Fi.
  - Permiso de sólo lectura para acceder a información sobre el estado del teléfono.

  Estos permisos se definen en el mismo orden descrito en las líneas incluidas en la Figura 4.2.11.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Figura 4.2.11: Permisos de la aplicación en el fichero 'AndroidManifest.xml'

  En [63] se enumeran todos los permisos existentes en Android según la documentación oficial. En el momento de instalación de la aplicación se pedirá consentimiento al usuario para aceptar todos los permisos de los que dispondrá la aplicación.

- Nombre de la aplicación. En este caso se ha decidido llamarla 'YouTube-QoE'.

Adicionalmente a estos datos, se definen las diferentes actividades o pantallas de las que dispone la aplicación y sobre cada una de éstas se incluyen algunos aspectos de formato. Cabe destacar entre ellos el fragmento utilizado en '*ReproductorFromURL*' para conseguir que esta actividad pueda ser directamente ejecutada a partir de un enlace *web*. En la Figura 4.2.12 puede observarse la etiqueta de tipo <*intent-filter*> que ha sido utilizada para ello. En este código se

pueden observar los tres dominios distintos que utiliza YouTube típicamente para referenciar vídeos y que podrán ser ejecutados con la aplicación desarrollada en este proyecto.

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />

    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />

    <data
        android:host="www.youtube.com"
        android:scheme="https" />
    <data
        android:host="youtu.be"
        android:scheme="http" />
    <data
        android:host="m.youtube.com"
        android:scheme="http" />
</intent-filter>
```

Figura 4.2.12: Captura de enlaces *web* en el fichero 'AndroidManifest.xml'

4.2.3. Principales librerías utilizadas

En el presente apartado se citan las librerías de programación que se han utilizado al margen de las que venían incluidas en el SDK de Android. Estas librerías han resultado necesarias para el desarrollo de algunas acciones que se mencionarán en adelante.

- ***YouTube Android Player API***

  '*YouTube Android Player API*' [45] es una API ofrecida en *Google Developers* para la inserción de reproductores embebidos de YouTube en cualquier aplicación. Con esta API se puede gestionar la carga y reproducción de cualquier vídeo y otros aspectos sobre la ejecución de estos. En la aplicación del presente proyecto se ha utilizado en las clases '*Reproductor*' y '*ReproductorFromURL*' para la reproducción del vídeo y en la clase '*MonitorizacionReproductor*' para la captura de todos los eventos del reproductor. Para esto último se dispone de las interfaces 'PlayBackEventListener' [57] y '*PlayerStateChangeListener*' [58]. Con ello se tratarán de recoger estadísticas sobre la QoS a nivel de aplicación.

  Los posibles eventos que permite registrar esta API son:

  - AD_STARTED: Este evento se produce en el caso de que antes de reproducir el vídeo se muestre un anuncio por pantalla. El método que captura este evento se denomina *onAdStarted()*.
  - ERROR(NETWORK_ERROR) y ERROR(INTERNAL_ERROR): Se trata de dos eventos que ocurren cuando se produce un error en la reproducción del vídeo, ya sea por un problema del dispositivo móvil o por falta de conectividad a Internet. Estos eventos se capturan desde el método *onError()*.
  - LOADING: Este evento se produce al inicio de la carga del vídeo solicitado. Se captura desde el evento *onLoading()*.

- LOADED: Este evento ocurre al finalizarse la carga del vídeo. Típicamente debería ocurrir después del evento 'LOADING'.
- VIDEO_STARTED: Se ejecuta después del evento 'LOADED' y significa que a partir de este momento se comienzan a recibir datos del vídeo para comenzar el *streaming*. Se implementa en el método *onVideoStarted()*.
- BUFFERING y NOT BUFFERING: Estos eventos hacen referencia al proceso de recepción de datos en el *buffer*. Respectivamente ocurrirán al iniciar y finalizar la recepción de muestras, ya sea cuando se produce una interrupción debido a que se ha vaciado el *buffer* o en la carga inicial del vídeo. Durante el intervalo de tiempo que se extiende entre estos dos eventos la reproducción del vídeo queda interrumpida. Estos eventos se capturan mediante el método *onBuffering()*.
- PAUSED: Este evento ocurre cuando el usuario explícitamente realiza una pausa del vídeo. Se implementa en el método *onPaused()*.
- PLAYING: Este evento ocurre cuando se inicia la reproducción del vídeo o en cualquier momento que se reanude. Este evento se captura desde el método *onPlaying()*.
- SEEKTO: Ocurre cuando el usuario explícitamente avanza o retrocede a lo largo del vídeo para iniciar la reproducción desde un punto concreto. Se implementa en el método *onSeekTo()*.
- VIDEO_ENDED: Evento producido cuando se alcanza el fin del vídeo y por tanto finaliza la reproducción. Se implementa en el método *onVideoEnded()*.
- STOPPED: Evento que referencia el cierre del reproductor embebido de YouTube. A partir de este momento este objeto ya no puede utilizarse, será necesario crear otro nuevo si se desea. Se implementa en el método *onStopped()*.

Todos los métodos mencionados se implementan en la clase '*MonitorizacionReproductor*' y los nombres de los eventos serán los mismos que se utilizarán en el fichero '*MonitorizacionReproductor.txt*' para indicar que se ha producido el evento. Con estos datos posteriormente se podrán obtener otras medidas interesantes como el número de interrupciones, la duración de éstas o el tiempo inicial de *buffering* del vídeo.

- **YouTube Data API (v3)**

  '*YouTube Data API*' [46] es una API distribuida por *Google Developers* para una amplia gama de lenguajes de programación distintos. En el ámbito del presente proyecto se ha utilizado la distribución de Java. Esta API se ha utilizado para la búsqueda de vídeos a partir de una palabra o frase introducida por el usuario y para la recolección de datos referentes a éstos vídeos, tales como el nombre o la descripción.

- ***YouTube Analytics API***

  '*YouTube Analytics API*' [47] es una API desarrollada por *Google Developers*. Esta API permite a la aplicación recoger información sobre estadísticas de reproducción de vídeos, métricas de popularidad e información

demográfica de los vídeos. En la aplicación desarrollada en el presente proyecto se ha utilizado para consultar datos como el número de reproducciones totales de un vídeo, la duración, el número de 'Me Gusta' y 'No Me Gusta' de los usuarios de YouTube, etcétera.

- ***Freebase API***

  '*Freebase API*' [64] es una librería que puede ser descargada desde *Google Developers* que ofrece funcionalidades para realizar consultas sobre la base de datos de *Freebase* [59]. En lo que al presente proyecto se refiere, esta librería se ha utilizado para la consulta de los tópicos relacionados con el vídeo que el usuario reproduce.

- ***JSch (Java Secure Channel)***

  '*JSch*' [61] es una implementación sobre Java del protocolo SSH (*Secure SHell*). Con esta aplicación se podrán realizar conexiones a servidores SSH y todas las funcionalidades que permiten este tipo de servicios. En el presente proyecto se ha utilizado para la conexión al servidor SFTP (*Secure File Transfer Protocol*), que utiliza un canal SSH para el envío seguro de los ficheros.

Como nota adicional, cabe destacar que todas las APIs de YouTube descritas requieren la utilización de una clave de desarrollador que podrá obtenerse de un modo gratuito registrándose y realizando una solicitud de éstas en [60].

4.2.4. Consideraciones finales del desarrollo de la aplicación

A modo de conclusión tras haber realizado por completo la implementación de la aplicación, se presentan en este apartado una serie de consideraciones. Se trata de aspectos que han surgido durante el proceso de desarrollo y que al inicio no estaban contemplados. Sin embargo, se consideran de gran relevancia puesto que afectan en el acabado del producto final diseñado.

- En la fase de desarrollo se contempló la posibilidad de realizar capturas del tráfico sobre la Interfaz activa del dispositivo móvil con acceso a Internet. De este modo, se tendría una fuente de datos adicional que permitiría analizar el tráfico enviado y recibido durante el *streaming* de vídeo. Esto permitiría analizar alguna información como el patrón de envío de datos por parte del servidor de YouTube, los *códecs* utilizados o el descubrimiento mediante ingeniería inversa del protocolo que utiliza YouTube para comunicarse con el cliente. Además, esta información podría relacionarse con el resto de información recopilada dando lugar a la posible obtención de estadísticas de interés.

  Para ello se trató de realizar una implementación en Android que permitiera la captura de tráfico utilizando un fichero binario de una distribución de *TCPDump* [65]. Este fichero binario se creó a partir de un proceso de compilación cruzada (*cross-compiling*) para que pudiera funcionar en dispositivos Android. Sin embargo, una vez programada esta función se pudo

comprobar que los resultados no eran del todo deseables. En algunas ocasiones funcionaba de un modo normal, pero en otras se producían errores que se achacan posiblemente a que Android no gestionaba correctamente la ejecución del fichero binario. Además era necesario que el dispositivo contase con permisos de superusuario, lo que implicaba realizar un proceso de *rooteado* sobre el terminal móvil. Por ello finalmente se decidió eliminar esta función de la aplicación.

No obstante, con el fin de ofrecer la posibilidad de obtener estas capturas de tráfico, se pone a disposición del usuario la documentación necesaria en el Anexo C de esta memoria. En este anexo se indica detalladamente cómo el usuario puede obtener capturas de tráfico con el móvil conectado a un ordenador mediante un cable USB. Se trata de un proceso que utiliza el fichero binario que se había generado para la implementación sobre la aplicación y cuyo funcionamiento se ha validado durante la fase de pruebas de este proyecto. Al final del apartado 4.4 se muestran resultados derivados de la obtención de una traza de ejemplo.

- En la hebra de 'MonitorizacionRed', como se ha descrito con anterioridad, se recogían parámetros referentes a la celda móvil a la que se encontraba conectado el dispositivo. Sin embargo, esto no podrá realizarse desde cualquiera de las versiones que soporta la aplicación desarrollada. Esta funcionalidad se aplica sólo en dispositivos móviles que tengan una versión de Android igual o superior a la 4.2 (*Jelly Bean MR1*). Esto se debe a que para ello son necesarias funciones de la API nivel 17 [66] del SDK de Android.

  Para salvar esta problemática y permitir que la aplicación pueda utilizarse desde la versión 3.0 de Android, se ha programado la aplicación de modo que para versiones anteriores a la 4.2 se puedan recoger todos los datos de esta hebra exceptuando el conjunto de datos que hacen referencia a información sobre la celda móvil.

  Para ello, en la clase '*MonitorizacionRed*' se ha creado un método denominado *medicionSeñal(TelephonyManager tm)* que realizará estas funciones y que sólo se ejecutará en caso de que el dispositivo tenga una versión de Android compatible. Esto se implementa en las líneas de código mostradas en la Figura 4.2.13.

```
if (Build.VERSION.SDK_INT > Build.VERSION_CODES.JELLY_BEAN_MR1)
    medicionSeñal(tm);
```

Figura 4.2.13: Obtención de parámetros de la celda móvil para versiones compatibles

4.3. Diseño de herramientas en el servidor

En el presente apartado se recoge una documentación de todas las herramientas y servicios desplegados en el lado del servidor.

La función del servidor consistirá en recibir los datos desde cualquier dispositivo móvil. En cada experimento se recibirán cuatro ficheros de texto que deberán procesarse para finalmente añadir la información de forma ordenada en una base de datos. Esta base de datos permitirá realizar consultas selectivas y facilitará el procesado de los datos para analizar toda la información que ha sido recopilada en su conjunto.

Para ello será necesaria la puesta en marcha de un servidor SFTP (*Secure File Transfer Protocol*) que se encargará de la recepción de los ficheros de texto, un servidor de base de datos y el diseño de un programa que lleve a cabo de forma automática la lectura de ficheros y actualice la información de la base de datos. Cada uno de estos tres elementos motivará la realización de un subapartado de los que se desarrollan a continuación.

El funcionamiento de todos estos servicios en su conjunto queda descrito en la Figura 4.3.1.

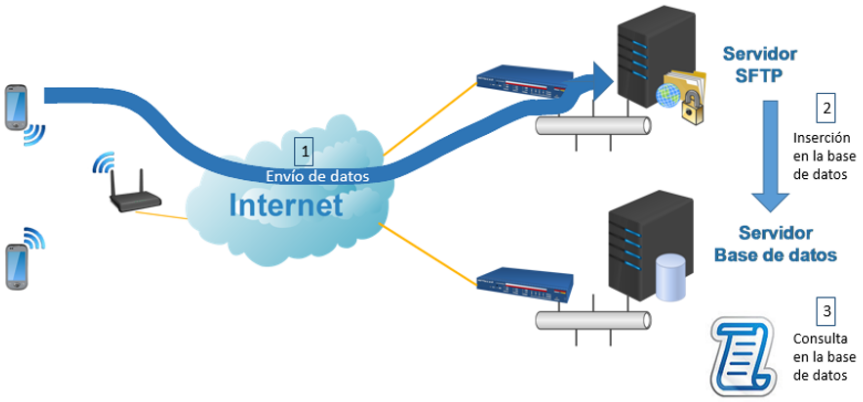

Figura 4.3.1: Descripción del funcionamiento del sistema diseñado

Este apartado se ha realizado con el objetivo de documentar la implementación realizada en el servidor. Si se desea consultar acerca de la instalación y puesta en marcha de todos los elementos del servidor, se deberá consultar el Anexo A de esta memoria, concretamente el apartado A.1 en que se detalla el despliegue de todos los servicios y herramientas necesarios en el servidor.

4.3.1. Servidor SFTP

El protocolo SFTP (*Secure File Transfer Protocol,* también denominado *SSH Transfer Protocol*) es un protocolo del nivel de aplicación del modelo OSI. Se trata de un protocolo que ofrece los mecanismos necesarios para la transferencia y manipulación de archivos de forma remota a través de un flujo de datos fiable. Para ello hace uso del protocolo SSH (*Secure SHell*), que permite establecer canales de datos seguros.

En el diseño previsto en el presente proyecto se ha decidido utilizar una modalidad de SFTP enjaulado. Esto quiere decir que el usuario SFTP no podrá acceder a todo el contenido del *host* en el que se aloja el servidor, sino que se

le ha restringido el acceso a un directorio creado expresamente para éste fin. Con ello, todos los dispositivos móviles podrán enviar a este directorio los ficheros de datos que se han generado durante el experimento utilizando un nombre de usuario y sus credenciales a través de un canal seguro. La elección de esta modalidad se ha elegido con el objetivo de aumentar el nivel de seguridad en el servidor, ya que en el caso de que un usuario consiguiera acceder al servidor SFTP con fines maliciosos, su campo de acción se restringiría únicamente a este directorio.

Además, al utilizarse un canal seguro, se preservará la confidencialidad ya que el contenido que viaja a través de Internet estará convenientemente cifrado para evitar que puedan realizarse escuchas en mitad del canal.

En la implementación se ha elegido la distribución *OpenSSH* [67] ampliamente utilizada en la actualidad debido a que ofrece un alto nivel de seguridad, es relativamente sencilla de instalar y se trata de una distribución de uso gratuito.

A este servicio se le ha incorporado otro denominado *Fail2ban* [68] que hará aún más seguro el acceso al servicio. Esta herramienta se encargará de revisar el registro del servidor en busca de acciones potencialmente maliciosas y llevará a cabo una actualización de las reglas del cortafuegos para la prevención de ataques. En el presente proyecto se utilizará para bloquear el acceso a usuarios, identificados mediante su IP, que hayan realizado un número determinado de intentos fallidos de autenticación al conectarse al servidor SFTP.

4.3.2. Servidor de base de datos

El servidor de base de datos se utilizará para el almacenamiento y organización de la información recogida en el servidor. Para ello se ha diseñado una arquitectura basada en el modelo de base de datos relacional. Este tipo de estructura es más compleja de realizar frente al modelo convencional, pero en el marco en el que se engloba este proyecto permitirá organizar mejor la información y dotar a la base de datos de mayor flexibilidad y eficiencia en el almacenamiento de datos.

En el diseño de la base de datos se ha contemplado crear una serie de tablas que permitan almacenar todos los datos recogidos. Para ello todos aquellos datos cuyo número de entradas en la tabla puede ser variable en función del experimento (eventos de interrupción, actualizaciones de la localización del dispositivo...), han motivado la creación de una nueva tabla. De este modo, cada entrada en las tablas contendrá un campo que identificará al experimento al que pertenece con el fin de que se puedan realizar posteriormente consultas selectivas en las que se puedan obtener los datos en conjunto de cada experimento de forma individualizada.

En la Figura 4.3.2 puede observarse un esquema con la distribución de las tablas junto con todos los campos que contiene cada una de éstas. En este esquema se puede comprobar que el diseño consta de seis tablas distintas y todas ellas tienen como parámetro común '*idDatos_video*'. Este será el campo que se utilizará para diferenciar de forma inequívoca las entradas referentes a un experimento concreto. El significado de cada uno de los datos almacenados se describirá en el apartado 4.4 de esta memoria.

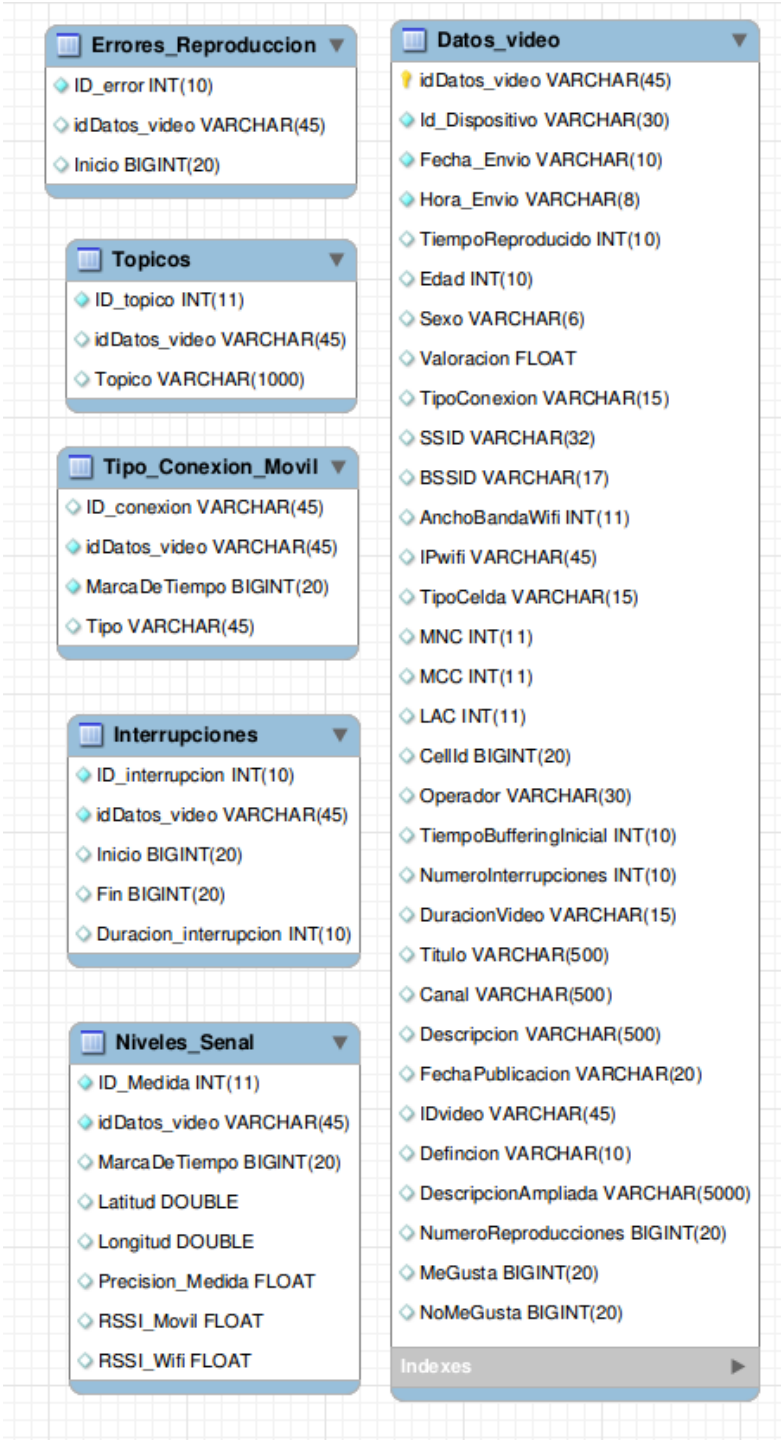

Figura 4.3.2: Diseño de las tablas de la base de datos

Para la implementación de este servidor se ha elegido una distribución de *mySQL* [69]. Se ha elegido en base a que se trata de una distribución de licencia gratuita, ampliamente utilizada y cuyos datos pueden exportarse con facilidad a otros formatos. Además se utilizan una serie de herramientas también gratuitas que facilitarán la abstracción del lenguaje específico de *mySQL* para consultar y gestionar la base de datos. De esta forma no supondrá un gran esfuerzo adaptarse a esta tecnología.

4.3.3. Programa de actualización de la base de datos

Una vez implementados los dos servidores descritos arriba, será necesario desarrollar un programa que actúe de pasarela entre la información almacenada en el servidor SFTP en forma de ficheros de texto y la información que finalmente se almacena en la base de datos.

Para ello, la solución que se ha considerado en el presente proyecto es la de crear un código utilizando el lenguaje de programación Java. Este programa tendrá como cometido la lectura de los ficheros alojados en el directorio del servidor SFTP, el procesado de la información y por último la inserción de ésta en la base de datos.

Además, este programa incorpora la funcionalidad de crear automáticamente la base de datos y sus correspondientes tablas, lo que será necesario en la primera ejecución del programa tras la instalación de todos los servicios. El mecanismo que se lleva a cabo consiste en que en cada ejecución se comprueba si existe la base de datos, y si no es así se ejecuta una rutina que se encargará de generarla de forma automática.

Asimismo, para optimizar el proceso de inserción de los datos y evitar posibles duplicidades en la información, en cada ejecución únicamente se añadirá a la base de datos aquella información que no se encuentre previamente almacenada.

Para realizar las funciones relacionadas con la comunicación con el servidor de base de datos se ha hecho uso de la API '*mySQL connector for Java*' [70]. Esta API permite abstraerse considerablemente de los procesos necesarios de conexión al servidor de base de datos, autenticación, creación de la base de datos y de sus tablas, inserción de datos, desconexión del servidor, etcétera. En definitiva se trata de un API que internamente hace uso del lenguaje SQL (*Structured Query Language*) para comunicarse con el servidor, simplificando así la tarea de programación. De este modo sólo ha sido necesario para el autor de este proyecto aprender algunas nociones básicas sobre este lenguaje, lo que ha supuesto un importante ahorro de tiempo en la realización de este programa.

En los apartados desarrollados seguidamente se recopilan aspectos relacionados con el desarrollo de este programa para informar al lector sobre el funcionamiento interno de éste. No obstante, el lector podrá referirse al Anexo B de esta memoria, concretamente al apartado B.1, donde se desarrolla un manual de usuario para este programa. En este manual se describe cómo configurar el programa y ejecutarlo. Además se recoge una lista de posibles fallos que podrían producirse junto con su diagnóstico más probable.

Procesado de los datos recogidos

Uno de los procesos que cabe destacar en la ejecución de este programa es la lectura de ficheros. Estos ficheros, como se ha comentado en el apartado 4.2.1 de esta memoria, se han conformado de modo que cada línea aporta un nuevo dato. De este modo el programa se encargará de leer todas las líneas secuencialmente para ir realizando un *parseo* y almacenar estos datos en variables. Finalmente cada una de estas variables recogidas se añadirán a un campo de la base de datos.

A partir de los datos almacenados, posteriormente se obtienen otros datos derivados de éstos como por ejemplo el número de interrupciones totales, la duración de cada interrupción o el tiempo de *buffering* inicial. Todo ello motiva la elección de la arquitectura elegida, en la que se ha decidido utilizar un servidor SFTP para almacenar todos los datos recopilados para posteriormente poder realizar un posprocesado antes de insertar los datos en la base de datos. Con ello se le resta carga de cómputo a los dispositivos móviles al finalizar el experimento y antes de enviar los datos finales recogidos.

La recopilación de la información de calidad de servicio a nivel de aplicación tiene un especial tratamiento. Esta información se encuentra en el fichero '*MonitorizacionReproductor.txt*' en el que se recogen todos los eventos posibles que ocurren en el reproductor junto con una marca temporal de éstos. Con ello se podrán buscar secuencias de eventos que darán información sobre si ha ocurrido una interrupción debida a que el *buffer* de datos se ha quedado vacío, interrupción explícita por parte del usuario o el tiempo del *buffering* inicial al cargar el vídeo.

Para capturar los eventos se ha utilizado la API de '*YouTube Android Player API*' [45]. En el apartado 4.2.3 de esta memoria pueden consultarse todos los eventos que esta API permite capturar.

De este modo, con esta información se pueden detectar otros eventos que serán de mayor interés para recoger información sobre la calidad de servicio. Estos eventos se identificarán inquívocamente siempre que se produzcan secuencias concretas de los eventos descritos arriba. A continuación se enumeran todos estos eventos:

- **Interrupción debida a agotamiento del *buffer***

  Se distinguen dos posibilidades:

  - La primera de ellas puede ocurrir en la primera interrupción tras el inicio de la reproducción del vídeo o tras una posible reanudación del vídeo realizada explícitamente por el usuario en el caso de que hubiera sido pausado con anterioridad. Ambos eventos darán lugar a que se registre el evento 'PLAYING'. Con ello la secuencia que detectaría una interrupción sería:

    PLAYING
    BUFFERING
    NOT BUFFERING

- Por otro lado puede ocurrir que se produzcan varia interrupciones consecutivas. De este modo la segunda de ellas y las siguientes se detectarán mediante la siguiente secuencia:

  NOT BUFFERING (indica el final de la interrupción anterior)
  BUFFERING
  NOT BUFFERING

Además se podrá calcular la duración de la interrupción con una precisión de milisegundos. Este valor se obtiene como la diferencia entre las marcas temporales registradas para los últmos eventos de 'BUFFERING' y 'NOT BUFFERING'. En el momento en el que se registra el evento 'NOT BUFFERING' sería cuando se reanuda la reproducción del vídeo.

- **Carga del vídeo e inicio de la reproducción**

  La secuencia que se produce cuando se solicita el vídeo hasta que comienza la reproducción de este es:

  LOADING
  LOADED
  VIDEO_STARTED
  BUFFERING
  NOT BUFFERING

  Con esta secuencia se podrá calcular el tiempo de *buffering* inicial del vídeo con una precisión de milisegundo como la diferencia de tiempo entre los eventos 'BUFFERING' y 'NOT BUFFERING'.

- **Error en la reproducción del vídeo**

  Existen dos posibles errores que pueden producirse durante la reproducción de un vídeo:

  - Error debido a pérdida de conectividad a Internet. Esto se identificaría mediante:

    ERROR(NETWORK_ERROR)

  - Error debido a un fallo en el dispositivo móvil. Identificado por el evento:

    ERROR(INTERNAL_ERROR)

Descripción de las clases del programa

A modo de resumen de la implementación realizada para este programa, se describen a continuación todas las clases de las que se constituye y una descripción de cada una de éstas. Además, con el fin de que el lector pueda conocer el funcionamiento interno del programa, se establecerán las relaciones que hay entre las distintas clases.

- **DescriptoresAplicacion.java**

  Esta clase reune todas las variables necesarias para la configuración del programa. En ellas se indica el nombre que tendrá la base de datos, el directorio SFTP donde se alojan los ficheros de texto con la información, la dirección IP necesaria para acceder al servidor de la base de datos y las credenciales de nombre de usuario y contraseña para acceder a la base de datos. En el código de esta clase se describe el significado de cada una de estas variables. No obstante, el lector puede referirse al Anexo B de esta memoria, concretamente al apartado B.1, donde se describe como llevar a cabo la configuración de este programa.

- **ActividadPrincipal.java**

  Esta actividad es la que alberga el método *main()* que regirá la ejecución del programa. En esta clase se hace uso de objetos de las clases '*BaseDatos*' y '*Muestra*' para llevar a cabo, por orden, la conexión a la base de datos, la inserción de los datos y por último la desconexión.

- **BaseDatos.java**

  En esta clase se incluyen todos los métodos necesarios para la comunicación con el servidor de base de datos. Entre estos métodos se destacan los siguientes:

  - *conectar()*: Para conectarse a la base de datos utilizando la IP del servidor de base de datos y las credenciales del usuario con el que se accederá a éste. Estos datos corresponden a las variables de la clase '*DescriptoresAplicacion*'.
  - *crearBaseDatos()*: Este método se ejecutará sólo si se comprueba que no existe la base de datos. Su cometido será crear la base de datos y todas las tablas utilizadas automáticamente.
  - *leerFicheros()*: Este método llevará a cabo la lectura de los ficheros alojados en el servidor SFTP para parsear la información y almacenarla en variables.
  - *insertarDatos(Muestra muestra, Connection conexion)*: Este método se encarga de insertar en la base de datos sólo aquellos datos que no se encuentran ya incorporados.

- **Muestra.java**

  Esta clase representa la información obtenida de un solo experimento. En las variables que contiene esta clase se almacenará ordenadamente toda la información de forma que cada variable corresponderá a un campo de la

base de datos. Esta clase se utiliza en las clases '*BaseDatos*' y '*Actividad-Principal*' para crear un array de objetos '*Muestra*', de modo que cada posición del array corresponda a un experimento completo.

- **Interrupcion.java**

  Esta clase se utiliza en las clases '*BaseDatos*' y '*Muestra*' y representa un evento de interrupción debido a que se ha vaciado el *buffer*. Básicamente contiene las marcas temporales de inicio y fin de la interrupción y su duración.

- **NivelesSenal.java**

  Esta clase representa una medición de RSSI (*Received Signal Strength Indication*) tanto de la red Wi-Fi como de la red móvil. Es utilizada en las clases '*BaseDatos*' y '*Muestra*' y contiene variables que representan una marca temporal de medida, la medida de estos niveles en decibelios y la localización del dispositivo cuando se ha producido esta medida.

- **TipoConexionMovil.java**

  Esta clase representa un dato sobre el tipo de tecnología de red móvil utilizada. Esta tecnología puede ir variando a lo largo de la reproducción del vídeo. Esto se debe a que el dispositivo móvil puede ir conectándose a diferentes celdas durante la reproducción del vídeo. Por ello se incluye una marca temporal que permita saber en qué momento se utiliza cada tecnología. Esta clase se utiliza en las clases '*BaseDatos*' y '*Muestra*'.

Diagrama de flujo del programa

En la Figura 4.3.3 se muestra un diagrama de flujo de la aplicación. De este modo el lector podrá observar desde un nivel alto de abstracción los procesos que se suceden a lo largo de una ejecución de este programa.

La ejecución consta de las siguientes etapas:

1. Conexión al servidor de base de datos utilizando el usuario creado expresamente para ello.
2. Se comprueba si existe la base de datos y las tablas necesarias. En caso de que no existan se genera por completo de forma automática. Si ya existe se pasa directamente a la etapa siguiente.
3. Se procede a la lectura de todos los ficheros almacenados en el directorio del servidor SFTP y se *parsean* para almacenarlos en variables.
4. Se realiza una inserción de todos los datos que no se encontraban en la base de datos.
5. Se procede a desconectarse del servidor de base de datos una vez finalizada la inserción.

Figura 4.3.3: Diagrama de flujo del programa de actualización de la base de datos

4.4. Evaluación de los resultados

Después de haber realizado la puesta en marcha de todas las herramientas necesarias, se ha llevado a cabo un período de pruebas de una duración aproximada de un mes. Durante este tiempo se ha tratado de distribuir la aplicación y recoger un número significativo de experimentos.

Finalmente, a día 1 de Septiembre de 2014 se evalúan los resultados. En definitiva, este proceso consiste en tratar de analizar la información que hay en la base de datos. En este apartado se pretenderá describir todos los datos recopilados así como la obtención de estadísticas posteriores. Estas estadísticas no tendrán un gran valor puesto que no se dispone de una muestra representativa. Sin embargo con ellas se pretende demostrar el potencial que puede tener la información contenida en la base de datos.

4.4.1. Descripción de la información recogida en la base de datos

La base de datos se conforma a partir de seis tablas distintas que siguen un modelo relacional. En la Figura 4.3.2 se pueden observar estas tablas con los campos correspondientes a cada una de ellas. Todas ellas contienen un parámetro denominado '*idDatos_video*' que identificará unívocamente a un experimento. De este modo se podrá conocer el experimento al que corresponde cada una de las entradas en la tabla. La división en las seis tablas distintas ha sido motivada debido a la naturaleza de los datos. Existe una tabla principal denominada '*Datos_video*' en la que se incluyen los datos que se recopilan en todos los experimentos. Por otro lado, el resto de tablas incluye un número variable de entradas referentes a cada experimento. Por ejemplo, una de ellas aporta información sobre interrupciones, de modo que cada entrada corresponderá a una interrupción producida en un vídeo. Sin embargo el número de interrupciones para cada vídeo es variable.

A continuación se realiza una descripción de todas estas tablas junto con el significado de cada uno de los campos. De este modo el lector podrá conocer el significado de la información recopilada.

Tabla '*Datos_video*'

Esta tabla contendrá una entrada por experimento recibido en el servidor. En ésta se recogen todos los datos recopilados comunes a todos los experimentos. Los campos de los que se constituye son:

- ***idDatos_video***

  Es un identificador único del experimento. Este identificador es un conjunto de la fecha y hora de envío de los datos al servidor junto con un identificador único del dispositivo móvil que lo ha enviado. Este identificador se obtiene a partir de la API del SDK de Android. Sin embargo, posteriormente se le aplica una función *hash*. Con ello se consigue que no

sea posible identificar el dispositivo aplicando así un carácter anónimo a los datos. Es una variable que permite hasta 45 caracteres.

- ***id_Dispositivo***

  Identificador del dispositivo utilizado en el campo '*idDatos_video*'. Permitirá realizar un filtrado de los datos por dispositivos. Se podrá conocer el número de dispositivos distintos que han participado en el proyecto así como los experimentos asociados a un mismo dispositivo. Es una variable con un máximo de 30 caracteres.

- ***Fecha_Envio***

  Fecha de envío de los datos hacia el servidor. Esta fecha está expresada en el formato '*día/mes/año*'. Es una variable de hasta 10 caracteres.

- ***Hora_Envio***

  Hora de envío de los datos hacia el servidor. La hora está expresada en el formato '*horas:minutos:segundos*'. Es una cadena de hasta 8 caracteres.

- ***TiempoReproducido***

  Tiempo total, expresado en milisegundos, durante el que se ha extendido la reproducción del vídeo. Podría ocurrir que este valor sea superior al de la duración total del vídeo. Esto ocurriría si el usuario por ejemplo decide retroceder durante la reproducción de modo que un mismo fragmento pueda repetirse varias veces. Es un entero de 32 bits sin signo.

- ***Edad***

  Edad del usuario que realiza la encuesta. El rango posible puede ser entre 1 y 110 años. Es un entero de 32 bits sin signo.

- ***Sexo***

  Sexo del usuario que realiza la encuesta. Puede tomar los valores 'Hombre' o 'Mujer'. Es una cadena de hasta 6 caracteres.

- ***Valoración***

  Valoración general del usuario acerca de la calidad de experiencia (QoE) sobre la reproducción del vídeo. Se trata de un valor que puede variar entre 1 y 5 con una precisión de 0.5. Es equiparable al indicador MOS (*Mean Opinion Score*) [10] utilizado a menudo en estudios de calidad de experiencia. Es una variable de coma flotante de precisión simple sin signo.

- ***TipoConexion***

  Esta variable indica el tipo de red al que está conectado el dispositivo móvil. Puede tomar los valores 'WiFi' o 'Movil'. Se trata de una cadena de hasta 15 caracteres.

- ***SSID***

  Indica el SSID (*Service Set IDentifier*) de la red de Wi-Fi en caso de que el dispositivo móvil esté conectado mediante esta tecnología. Si no está conectado a una red Wi-Fi tomará el valor *null*. Se trata de una cadena de hasta 32 caracteres, que es la longitud máxima del SSID por definición.

- ***BSSID***

  Indica el BSSID (*Basic Service Set IDentifier*) de la red Wi-Fi en caso de que el dispositivo móvil esté conectado mediante esta tecnología. Si no está conectado a una red Wi-Fi tomará el valor *null*. Se trata de una cadena de hasta 17 caracteres. Esto se debe a que el BSSID sse define como una dirección MAC que siempre tendrá esta longitud.

- ***AnchoBandaWifi***

  Indica el ancho de banda máximo teórico de la red Wi-Fi en función de su tecnología radio en caso de que el dispositivo móvil esté conectado mediante Wi-FI. Si no está conectado a una red Wi-Fi tomará el valor *null*. Se trata de un entero de 32 bits sin signo.

- ***IPwifi***

  Este campo contiene la función *hash* aplicada a la IP del dispositivo móvil en caso de estar conectado a la red Wi-Fi. De este modo se preservará la privacidad del usuario, puesto que no se puede conocer su IP. Si no está conectado a una red Wi-Fi tomará el valor *null*. Se trata de una cadena de hasta 45 caracteres.

- ***TipoCelda***

  Indica la tecnología de acceso asociada a la celda de la red móvil. Por ejemplo, WCDMA (*Wideband Code Division Multiple Access*) o FDMA (*Frequency Division Multiple Access*). Es una variable de hasta 15 caracteres.

- ***MNC***

  Indica el MNC (*Mobile Network Code*) asociado a la celda de la red móvil. Es un entero de 32 bits.

- ***MCC***

  Indica el MCC (*Mobile Country Code*) asociado a la celda de la red móvil. Es un entero de 32 bits.

- ***LAC***

  Indica el LAC (*Location Area Code*) asociado a la celda de la red móvil. Es un entero de 32 bits.

- ***CellId***

  Indica el identificador de celda (*Cell ID*) asociado a la celda de la red móvil. Es un entero de 64 bits.

- ***Operador***

  Indica el nombre operador móvil al que está conectado el dispositivo. Es una cadena de hasta 30 caracteres.

- ***TiempoBufferingInicial***

  Indica el tiempo, expresado en milisegundos, durante el que se ha extendido la carga inicial del *buffer* antes de comenzar la reproducción del vídeo. Es un entero sin signo de 32 bits.

- ***NumeroInterrupciones***

  Indica el número de interrupciones que se han producido debido a que se ha vaciado el *buffer* de datos. No incluye las interrupciones realizadas de forma explícita por parte del usuario. Se trata de un entero de 32 bits sin signo.

- ***DuracionVideo***

  Indica la duración total del vídeo. No implica que se haya reproducido el vídeo durante este tiempo. El tiempo total durante el que se ha reproducido el vídeo viene indicado en el campo '*TiempoReproducido*' de esta misma tabla. El formato de este campo es '*horas:minutos:segundos*'. Se trata de una cadena de hasta 15 caracteres.

- ***Titulo***

  Indica el título del vídeo que se ha reproducido. Se trata de una cadena de hasta 500 caracteres.

- ***Canal***

  Indica el nombre del canal de YouTube que ha subido el vídeo reproducido. Se trata de una cadena de hasta 500 caracteres.

- ***Descripcion***

  Incluye una descripción resumida del vídeo. Se trata de una cadena de hasta 500 caracteres.

- ***FechaPublicacion***

  Indica la fecha en que fue publicado el vídeo reproducido. Sigue el formato '*día/mes/año*'. Se trata de una cadena de 20 caracteres.

- ***IDvideo***

  Representa un identificador único del vídeo reproducido. Este identificador se utiliza en los servidores de YouTube para identificar de forma unívoca el vídeo.

- ***Definicion***

  Indica la definición del vídeo descargado. Se trata de una cadena de hasta 10 caracteres. Entre los posibles valores se encuentran:

  - sd (*Standard Definition*)
  - hd (*High Definition*)

- ***DescripcionAmpliada***

  Incluye una descripción ampliada del vídeo reproducido. Se trata de una cadena de hasta 5000 caracteres.

- ***NumeroReproducciones***

  Indica el número total de reproducciones que ha tenido el vídeo por parte de todos los usuarios de YouTube. Se trata de un entero sin signo de 64 bits.

- ***MeGusta***

  Número total de 'Me Gusta' que tiene el vídeo reproducido. Se trata de un indicador de satisfacción de los usuarios de YouTube que han visto el vídeo. Se trata de un entero sin signo de 64 bits.

- ***NoMeGusta***

  Número total de 'No Me Gusta' que tiene el vídeo reproducido. Se trata de un indicador de insatisfacción por parte de los usuarios de YouTube que han visto el vídeo. Se trata de un entero sin signo de 64 bits.

Tabla '*Interrupciones*'

Esta tabla recoge todas las interrupciones que se producen en cada experimento. Cada entrada de la tabla contiene información sobre una interrupción. Cabe destacar que todas estas interrupciones se deben a que se han agotado los datos en el *buffer*. No se incluyen interrupciones realizadas de forma explícita por parte del usuario. Contiene los siguientes campos:

- ***ID_interrupcion***

  Indica el número por orden de la interrupción producida en un mismo experimento. La primera interrupción de cada experimento tendrá el valor '1' y las siguientes se irán incrementando de uno en uno. Se trata de un entero sin signo de 32 bits.

- ***idDatos_video***

  Identificador único del experimento. Se trata de una cadena de hasta 45 caracteres.

- ***Inicio***

  Indica una marca temporal tomada cuando inicia la interrupción. Esta referencia temporal se define como la cantidad de milisegundos transcurridos desde las 0:00 horas del 1 de Enero de 1970. Es una referencia temporal típica de sistemas que utilizan el sistema operativo Linux. Se trata de un entero sin signo de 64 bits.

- ***Fin***

  Indica una marca temporal tomada cuando finaliza la interrupción. Esta referencia temporal se define como la cantidad de milisegundos transcurridos desde las 0:00 horas del 1 de Enero de 1970. Se trata de un entero sin signo de 64 bits.

- ***Duracion_interrupcion***

  Indica la duración, expresada en milisegundos, de la interrupción que se ha producido. Se trata de un entero de 32 bits sin signo.

Tabla '*Niveles_Senal*'

Esta tabla representa una medida de los niveles de señal (RSSI - *Received Signal Strength Indication*) correspondientes a las redes Wi-Fi y móvil. Estas medidas están asociadas la localización del dispositivo móvil cuando se ha realizado la medición. El nivel de señal de la red Wi-Fi sólo será posible obtenerlo cuando el dispositivo esté conectado a una red Wi-Fi. Sin embargo, el nivel de señal de la red móvil podrá obtenerse en todo caso puesto que el móvil siempre está conectado a la red del operador. Contiene los siguientes campos:

- ***ID_medida***

  Se trata del número por orden de la medición de señal dentro de un mismo experimento. Este número se irá incrementando en una unidad para las sucesivas medidas del mismo experimento. Se trata de un entero de 32 bits sin signo.

- ***idDatos_video***

  Identificador único del experimento. Se trata de una cadena de hasta 45 caracteres.

- ***MarcaDeTiempo***

  Marca de tiempo tomada al mismo tiempo que se anota la medida de señal. Se trata de una referencia temporal correspondiente al tiempo transcurrido, en milisegundos, desde las 0:00 horas del 1 de Enero de 1970. Se trata de un entero sin signo de 64 bits.

- ***Latitud***

  Latitud, en grados, de la localización del dispositivo móvil en el momento de realizar la medición de señal. Variable de coma flotante de doble precisión.

- ***Longitud***

  Longitud, en grados, de la localización del dispositivo móvil en el momento de realizar la medición de señal. Variable de coma flotante de doble precisión.

- ***Precision_Medida***

  Precisión, en metros, de la medida de la localización del dispositivo móvil en el momento de realizar la medición de señal. Variable de coma flotante de doble precisión.

- ***RSSI_Movil***

  RSSI (*Received Signal Strength Indication*), en decibelios, medido sobre la señal de la red móvil recibida. Se trata de una variable de coma flotante de precisión simple.

- ***RSSI_Wifi***

  RSSI (*Received Signal Strength Indication*), en decibelios, medido sobre la señal de la red Wi-Fi recibida. Se trata de una variable de coma flotante de precisión simple.

Tabla '*Tipo_Conexion_Movil*'

Esta tabla representa la tecnología móvil de la celda a la que está conectado el dispositivo móvil. Puesto que el dispositivo puede estar en movimiento durante la reproducción del vídeo, este puede ir conectándose a diferentes celdas con diferentes tecnologías. Por ello puede haber más de una entrada para cada experimento. Contiene los siguientes campos:

- ***ID_conexion***

  Se trata del número por orden de las entradas en esta tabla de un mismo experimento. Este número será '1' para la primera entrada del experimento y se irá incrementando para las entradas consecutivas. Es un entero sin signo de 32 bits.

- ***idDatos_video***

  Identificador único del experimento. Se trata de una cadena de hasta 45 caracteres.

- ***MarcaDeTiempo***

  Marca de tiempo tomada cuando se detecta la nueva tecnología. Se trata de una referencia temporal correspondiente al tiempo transcurrido, en milisegundos, desde las 0:00 horas del 1 de Enero de 1970. Se trata de un entero sin signo de 64 bits.

- ***Tipo***

  Esta variable indica el tipo de tecnología de la red móvil que se utilizada en el momento registrado por el campo '*MarcaDeTiempo*'. Se trata de una cadena de hasta 45 caracteres. Las tecnologías más típicas que podrán utilizarse son:

  - CDMA
  - EDGE
  - GPRS
  - HSDPA
  - HSPA
  - HSPA+
  - UMTS
  - LTE

Tabla '*Topicos*'

Esta tabla incluye todos los tópicos relacionados con los diferentes vídeos reproducidos. Al incluir en cada entrada el campo '*idDatos_video*', se puede identificar el título del vídeo al que corresponde y todos los datos almacenados del mismo experimento. Contiene los siguientes campos:

- ***ID_topico***

  Se trata del número por orden de las entradas en esta tabla de un mismo experimento. Este número será '1' para la primera entrada del experimento y se irá incrementando para las entradas consecutivas. Es un entero sin signo de 32 bits.

- ***idDatos_video***

  Identificador único del experimento. Se trata de una cadena de hasta 45 caracteres.

- ***Topico***

  Indica un tópico relacionado con el vídeo que se ha reproducido en el experimento identificado por el campo '*idDatos_video*'. Se trata de una cadena de hasta 1000 caracteres.

Tabla '*Errores_Reproduccion*'

Esta tabla recoge todos los posibles errores que se pueden producir a lo largo de la reproducción de un vídeo. Estos errores se pueden deber a pérdida de la conectividad hacia Internet o por fallos producidos en el dispositivo móvil.

- ***ID_error***

  Se trata del número por orden de las entradas en esta tabla de un mismo experimento. Este número será '1' para la primera entrada del experimento y se irá incrementando para las entradas consecutivas. Es un entero sin signo de 32 bits.

- ***idDatos_video***

  Identificador único del experimento. Se trata de una cadena de hasta 45 caracteres.

- ***Inicio***

  Marca de tiempo tomada cuando se produce el error. Se trata de una referencia temporal correspondiente al tiempo transcurrido, en milisegundos, desde las 0:00 horas del 1 de Enero de 1970. Se trata de un entero sin signo de 64 bits.

4.4.2. Análisis de los resultados

Con esta gran cantidad de datos resultará útil saber cómo realizar consultas que permitan obtener unos datos concretos. Para ello, en el Anexo D de esta memoria se presenta una documentación en la que se explican algunos conceptos básicos para la realización de consultas en bases de datos *mySQL*. Asimismo se incluyen algunos ejemplos de consultas que pueden resultar interesantes en el presente proyecto.

A continuación se presentan algunas estadísticas obtenidas con los datos recogidos durante la fase de pruebas. En este momento se dispone en la base de datos

de un total de treinta y un experimentos distintos realizados desde un total de siete dispositivos móviles diferentes.

Una de las estadísticas que se presentan en múltiples estudios de la revisión bibliográfica hecha en este proyecto, es la inferencia de la relación entre el parámetro MOS (*Mean Opinion Score*) [10] y el número de interrupciones experimentadas. Para ello sólo se contarán las interrupciones causadas por un agotamiento de las muestras del *buffer*. En la Figura 4.4.1 puede observarse una gráfica con los resultados obtenidos.

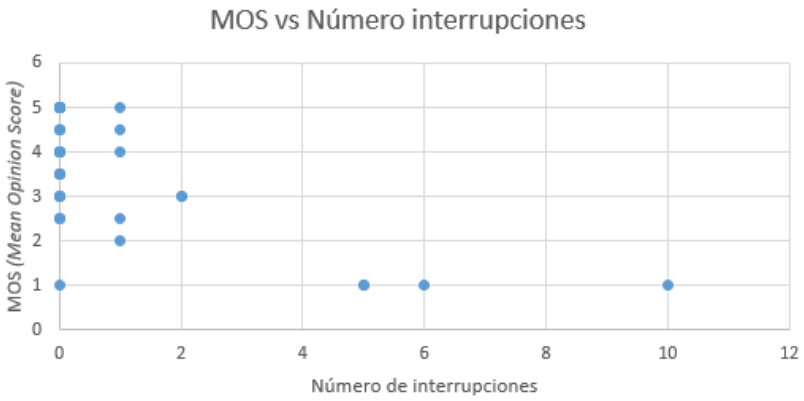

Figura 4.4.1: Gráfica de *Mean Opinion Score* frente al número de interrupciones

Otra estadística que puede resultar de interés podría ser la obtención de una gráfica que relacione el parámetro MOS (*Mean Opinion Score*) [10] frente al tiempo medio por interrupción a lo largo de la reproducción del vídeo. En la Figura 4.4.2 puede observarse una gráfica realizada con los datos de los que se dispone.

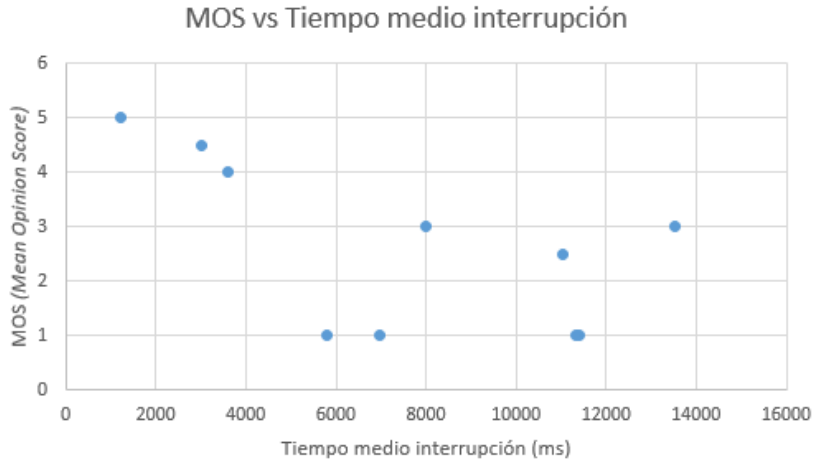

Figura 4.4.2: Gráfica de *Mean Opinion Score* frente al tiempo medio por interrupción

Ambas estadísticas son poco significativas debido a la reducida muestra de la que se dispone en la base de datos. Sin embargo puede observarse una tendencia que sigue lo expuesto en algunos trabajos que se han podido encontrar en la revisión bibliográfica. Un aumento en el número total de interrupciones o en el tiempo medio de duración de éstas afecta negativamente sobre la calidad experimentada por el usuario.

Por otro lado, con los niveles de señal medidos se podría trazar un mapa en el que mediante un código de colores pudieran representarse de un modo visual todas las medidas. También se podría por ejemplo obtener un histograma que represente los niveles de señal de la red móvil o Wi-Fi registrados. En la Figura 4.4.3 se observa la obtención de un histograma que representa los niveles de señal de la red Wi-Fi con los datos de los que se dispone.

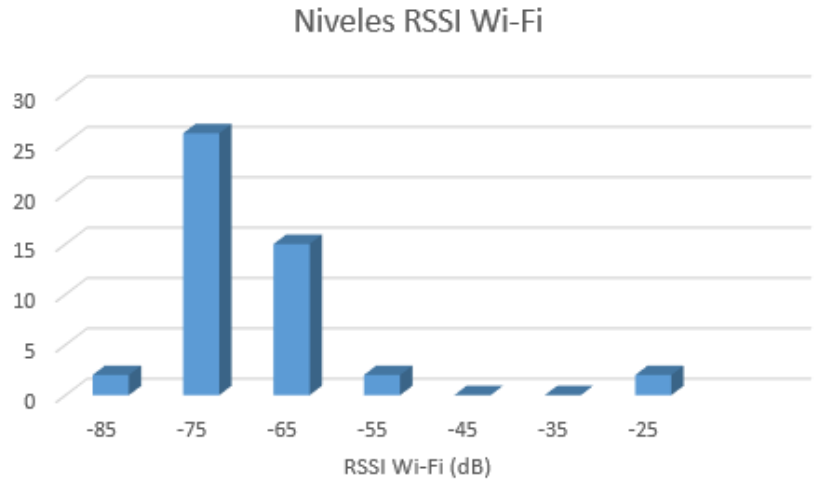

Figura 4.4.3: Histograma de niveles RSSI en redes Wi-Fi

Estas estadísticas se muestran con el mero objetivo de presentar posibles resultados. No obstante, son sólo algunos ejemplos de la infinidad de estadísticas diferentes que podrían obtenerse con todos los datos que se recopilan en el proyecto.

Por último cabe destacar la posibilidad de la obtención de trazas de tráfico con *TCPDump* [65]. Para ello, en el Anexo C de esta memoria se desarrolla un tutorial en el que se explica cómo puede hacerse esto. Una de las aplicaciones que podría tener la obtención de estas trazas sería por ejemplo analizar el patrón del tráfico de recepción de datos mientras se está efectuando un *streaming* de vídeo. En la Figura 4.4.4 se muestra el ejemplo del patrón de tráfico obtenido en una prueba. Esta prueba ha sido realizada con el dispositivo conectado a una red Wi-Fi. Como se puede observar, en este caso a lo largo de la carga del vídeo se observa una ráfaga inicial de recepción de datos para llenar el *buffer* que se extiende aproximadamente hasta el segundo 15 desde el inicio.

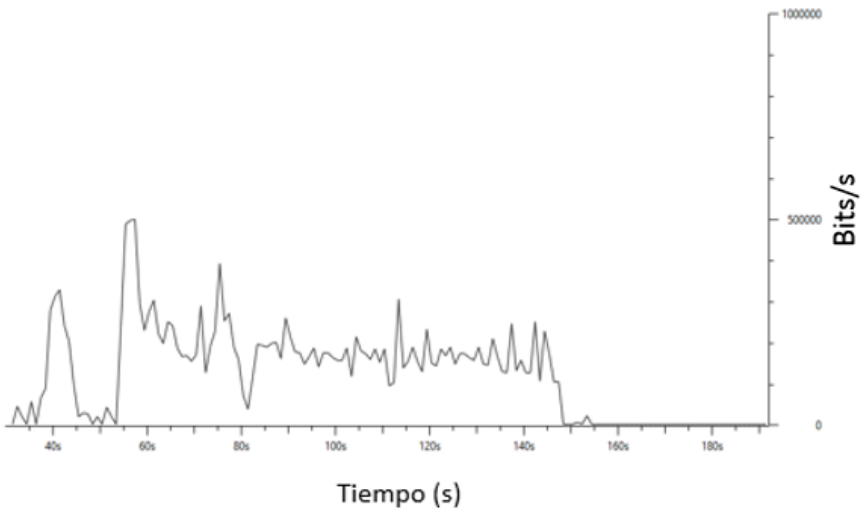

Figura 4.4.4: Traza de tráfico recibido durante la reproducción de un vídeo

Capítulo 5

Conclusiones y vías futuras

El presente capítulo se desarrolla a modo de culminación de la presente memoria. En él se recogerá un apartado de conclusiones después de haber finalizado todo lo referente al Trabajo de Fin de Grado. En este apartado se tratarán de recoger cuestiones relevantes que han surgido a lo largo de los procesos de diseño e implementación y las contribuciones finales del proyecto.

Por otro lado se expone un apartado de vías futuras. Este apartado tendrá como finalidad informar al lector sobre las posibles cuestiones futuras que atañen al proyecto. En él se hará hincapié principalmente sobre futuras mejoras del trabajo que podrían realizarse.

Finalmente, se desarrolla un apartado en el que el autor de este proyecto incluye algunas reflexiones relativas a la realización del proyecto.

5.1. Conclusiones

En el presente Trabajo de Fin de Grado se ha diseñado un sistema compuesto por diversas herramientas que trabajan de forma colaborativa. Con ello se ha conseguido elaborar una plataforma de monitorización de datos sobre *streaming* de vídeo en dispositivos móviles. Estos datos darán información general sobre la Calidad de Servicio (QoS) y la Calidad de Experiencia (QoE) en diversos experimentos en los que se llevan a cabo reproducciones de vídeos del servicio de YouTube. Además se incluyen datos sobre niveles de cobertura de las redes móvil y Wi-Fi, datos demográficos de los usuarios y estadísticas de los vídeos tales como el número de reproducciones, tópicos relacionados, valoración de los usuarios de YouTube, etcétera. Todos estos datos quedarán recogidos en una base de datos centralizada que se utilizará como fuente de información para posibles estudios estadísticos posteriores.

Las principales contribuciones de este proyecto son:

- Se trata de un sistema innovador en tanto que, tras haber realizado la revisión bibliográfica, no se ha podido encontrar una aplicación que realice

las mismas funciones. El trabajo más similar que se ha encontrado ha sido en [31]. Sin embargo según se indica en este artículo, se trata de una aplicación que sólo recopila información sobre la QoE a través del método *OneClick* [32] y relaciona esta información con datos de QoS de nivel de red (velocidad de transmisión, pérdida de paquetes y retardo) medidos en el servidor de *streaming*. Esto indica que el servidor de vídeos debe trabajar de forma colaborativa con el usuario final para combinar la información recogida. En este caso la herramienta tendría sentido para un administrador de un servidor de *streaming* de vídeo que desease recoger información sobre su propio servicio.

En el sistema desarrollado en el presente proyecto, toda la información se recopila en el lado del cliente, desde el dispositivo móvil. Esto ha posibilitado la realización de experimentos sobre el servidor de vídeos YouTube, que se trata actualmente del líder mundial en el ámbito del *streaming* sobre vídeo, lo que magnifica considerablemente el alcance de este proyecto. Además, cabe destacar que en la aplicación desarrollada no sólo se recogen parámetros de QoS a nivel de red, sino que también se han monitorizado eventos en la aplicación que permitirán obtener información de la QoS a nivel de aplicación (interrupciones, tiempo de carga inicial del vídeo, etcétera). Con todo ello se tienen datos de QoE (medida mediante el parámetro MOS), datos de QoS a nivel de red y de aplicación, datos demográficos y datos estadísticos sobre los vídeos reproducidos. Todos ellos se podrán utilizar combinadamente para realizar multitud de estudios diferentes.

- La aplicación desarrollada para móviles Android de cara al usuario funcionará como un reproductor corriente de vídeos de YouTube. Para ello se ha procurado realizar todos los procesos de monitorización de un modo totalmente transparente y se han generado dos vías alternativas para reproducir vídeos. Por un lado se ofrece la posibilidad de buscar vídeos en el servidor de YouTube de modo que se podrá obtener una lista de resultados y elegir uno de ellos. Por otro lado se podrá ejecutar la aplicación directamente desde un enlace *web* que referencie un vídeo concreto de YouTube para llevar a cabo la reproducción de éste. Asimismo, el formulario que se le ofrece al usuario es bastante breve y rápido de completar, pero sin embargo aporta información muy importante. En éste se incluyen datos demográficos (sexo y edad) y la valoración del usuario (*Mean Opinion Score*) tras visualizar el vídeo, lo que dará información sobre la QoE. Con todas estas consideraciones se puede afirmar que el uso de esta aplicación no supondrá un gran esfuerzo adicional para el usuario frente al uso de la aplicación oficial de YouTube para reproducir vídeos.

- Esta aplicación está basada en la filosofía de *crowdsourcing*, lo que ha reportado una gran eficiencia en costes en lo que se refiere a la obtención de datos y la oportunidad de acceder a una muestra muy amplia y heterogénea conformada por todos los usuarios que utilizan móviles con el sistema operativo Android en la actualidad.

- En cuanto a la implementación en el lado del servidor, sólo será necesaria una puesta en marcha inicial de todos los servicios, cuyo proceso viene descrito en la documentación del Anexo A. Una vez hecho esto se podrá utilizar el *software* desarrollado en Java que se encargará de realizar de un

modo totalmente automático la creación de la base de datos y de sus tablas y la inserción de los datos. Este programa actúa de un modo eficiente en tanto que actualiza la base de datos insertando sólo la información que aún no se encuentra alojada, procurando así un ahorro de carga computacional y evitando posibles duplicidades en los datos. La configuración y el uso de este programa puede consultarse en el Anexo B de esta memoria.

- El hecho de utilizar una distribución de *mySQL* para la base de datos ofrecerá una gran flexibilidad en el momento de analizar y procesar los datos. Esta base de datos se ha generado tomando en consideración un modelo relacional que distribuye la información en seis tablas distintas de un modo flexible y eficiente en cuanto al almacenamiento de los datos. Se trata de una base de datos de licencia gratuita y que además permite exportar los datos a otros formatos comúnmente utilizados para el procesado, como por ejemplo el formato CSV.

- Una de las principales contribuciones de este proyecto es la posibilidad de la creación de modelos experimentales que relacionen la QoE y la QoS en *streaming* de vídeo en dispositivos móviles. Con los datos que se recogen en la base de datos se tiene información suficiente para obtener estadísticas que permitan generar modelos de esta naturaleza. Esto resultará un gran logro ya que hasta el momento sólo se han podido encontrar estudios de este tipo sobre YouTube en ordenadores personales. Sin embargo, en la revisión bibliográfica se ha podido comprobar que la generación de tráfico difiere cuando se trata de dispositivos móviles, por lo que estos estudios mencionados no serán extensibles a dispositivos móviles. Además se suma el factor de que típicamente las expectativas del usuario en la reproducción de un vídeo en un ordenador personal son más exigentes que cuando éste reproduce un vídeo en un dispositivo móvil, que suele tener una pantalla más pequeña.

- Por último cabe destacar la posibilidad de la obtención de trazas de tráfico al mismo tiempo que se lleva a cabo el *streaming* de vídeo. Esto permitirá estudiar la generación de tráfico por parte del servidor de YouTube cuando el *streaming* se realiza sobre dispositivos móviles, que como se ha explicado anteriormente difiere del caso de los ordenadores personales. Si bien, se trata de una opción que no está directamente integrada en la aplicación desarrollada ya que esto no ha sido posible tras múltiples intentos.

  Para conseguir esto se presenta una solución alternativa en la será necesario mantener conectado el dispositivo móvil a un ordenador mediante un cable USB y seguir los pasos descritos en el Anexo C de esta memoria.

5.2. Vías futuras

Aunque se pueden dar por conseguidos todos los objetivos propuestos para el proyecto, se recogen a continuación algunos aspectos de este trabajo sobre los que podrían añadirse mejoras o implementaciones adicionales.

Estas mejoras propuestas se resumen en los siguientes puntos:

- Se pudo comprobar que la librería 'YouTube Player API' que ha sido utilizada para registrar eventos y datos relativos al reproductor embebido no incluía algunas funcionalidades que ésta misma ofrecía para otras plataformas como Javascript. Entre estas funcionalidades hubiera resultado de gran interés poder consultar periódicamente la cantidad de datos que se alojan en el *buffer* durante el *streaming* de vídeo y el tamaño máximo de este *buffer*. Con ello podrían realizarse estadísticas sobre la ocupación del *buffer* y la recepción de los datos.

  Una posible mejora consistiría en aplicar esta funcionalidad en caso de que se incluyera en las nuevas distribuciones de esta API para Android. No se trata de algo que pueda ocurrir con seguridad, sin embargo se considera de interés que el lector tenga al menos conocimiento de esta posibilidad en caso de que fuera factible en un futuro.

- En cuanto a la obtención de trazas de tráfico con *TCPDump*, se trata de un objetivo no previsto en un inicio. Sin embargo, durante la fase de diseño se consideró esta opción. De este modo se haría de forma automática la captura de tráfico al mismo tiempo que se llevan a cabo los otros procesos de monitorización.

  Finalmente, tras diversos intentos que han producido un desajuste en la planificación temporal del proyecto, no se ha conseguido integrar esta funcionalidad en la aplicación. La solución que se intentó aplicar necesitaba permisos de superusuario en el dispositivo y con éstos se podría ejecutar automáticamente un fichero binario necesario para iniciar la captura de tráfico. Sin embargo, aunque se ha conseguido en algunas pruebas que funcione con éxito, se trata de una implementación que finalmente se ha desechado debido a que se producían fallos asiduamente.

  No obstante, como se expone en el apartado 5.1, se ha diseñado una solución que no integra esta funcionalidad pero que sí permite la obtención de estas trazas de tráfico. Aunque para ello serán necesarios permisos de superusuario en el dispositivo móvil y conectar éste a un ordenador mediante un cable USB.

  Dicho esto se propone como posible mejora tratar de integrar esta funcionalidad en la aplicación. Aunque a fecha de hoy al menos el autor no ha encontrado soluciones viables, es posible que aparezcan en un futuro algunos mecanismos que lo permitan.

- Por otro lado, se ha conseguido obtener datos sobre el ancho de banda máximo teórico en función del tipo de red al que se está conectado. Sin embargo no se han encontrado métodos que permitan medir la velocidad de transmisión instantánea de un modo fiable. También sería interesante conseguir la medición de parámetros como el *jitter* o el retardo extremo a extremo. Esto también repercutiría en una mejora del proyecto, puesto que se trata de una información que puede resultar de interés para estimar la QoS.

5.3. Valoración personal

Para finalizar la elaboración de esta memoria se exponen a continuación una serie de reflexiones del autor de este proyecto después de haber concluido la elaboración del presente Trabajo de Fin de Grado.

En primer lugar cabe destacar la importancia y el significado de este trabajo. Como su propio nombre indica se trata de un trabajo con el que se concluye un grado formativo cuya duración prevista es de cuatro años. Esto se traduce en que este trabajo se realiza con el fin de poner de manifiesto todas las aptitudes y conocimientos que se han adquirido a lo largo de todo este periodo. En lo que se refiere a este aspecto, ha supuesto un gran reto en el sentido de que ha sido necesario continuamente actuar con capacidad analítica, una de las principales características que deben definir a un ingeniero. Por otro lado se ha tenido que hacer uso de gran cantidad de conceptos y conocimientos aprendidos a lo largo de estos cuatros años.

Si lo mencionado arriba es de gran relevancia, quizás el principal éxito reside en el hecho de que ha sido necesario integrar gran cantidad de conocimientos necesarios para la realización satisfactoria del proyecto. Esto se traduce en la necesidad de aplicar la capacidad de abstracción, un concepto que se encuentra presente en continuamente en la docencia de este grado. Con esto ha sido posible realizar por separado distintas implementaciones para después integrarlas y hacerlas colaborar entre sí dando lugar a un único sistema que se encargará de llevar a cabo todos los cometidos planteados en un inicio. Se podría afirmar que se trata de un trabajo que se ajusta en gran medida a lo que podría presentarse en un escenario profesional real.

Para la elaboración del proyecto ha sido necesaria una formación adicional que se considera de gran utilidad añadida a los conocimientos que ya se tenían. Entre estos aspectos formativos adquiridos, se ha alcanzado un nivel bastante avanzado de programación de aplicaciones para dispositivos Android y una profundización en el lenguaje de programación Java. Asimismo ha sido necesario poner en marcha dos servicios: SFTP (*Secure File Transfer Protocol*) y servidor de base de datos. Para este último servicio también ha sido necesaria la adquisición de conocimientos básicos para la elaboración de un esquema de base de datos adecuado que a partir de sus tablas permita ser lo más eficiente y flexible posible.

En cuanto al programa de actualización de base de datos, se ha necesitado introducirse en el lenguaje SQL para interactuar con el servidor de base de datos. Con ello ha sido posible la creación automática de la base de datos y sus tablas y la posterior inserción de los datos.

También merecen una especial mención todos los conocimientos teóricos adquiridos en la fase preliminar del proyecto. Durante todo este periodo se realizó una revisión del estado del arte y se leyó documentación relacionada con aspectos como el *streaming* de vídeo, la calidad de servicio (QoS) y la Calidad de Experiencia (QoE) o la filosofía de *crowdsourcing*. Esto ha permitido conocer con gran rigurosidad todos los ámbitos que engloba este proyecto.

Por último se destaca la satisfacción personal y profesional del autor, en tanto que ha sido capaz de abordar todo el proyecto en su totalidad. Pese a haberse tratado de un trabajo bastante desafiante para éste, finalmente se han alcanzado

todos los objetivos propuestos inicialmente e incluso se han conseguido otros de forma adicional.

Apéndices

Apéndice A

Manual de instalación

En este anexo se pretende describir todo el proceso necesario para la puesta en marcha del sistema diseñado en el proyecto. Para ello se instalarán todos los servicios necesarios en el ordenador que actuará como servidor y posteriormente se describirá el proceso de instalación de la aplicación para el dispositivo móvil.

A continuación se desarrollan dos apartados principales en los que se describen respectivamente los pasos que se deben realizar en el lado del servidor y en el lado del cliente (aplicación móvil) para implementar el diseño descrito en el presente proyecto.

A.1. Manual de instalación del lado del servidor

En el lado del servidor se instalarán las siguientes herramientas y servicios:

- Servidor SFTP para la recepción de ficheros con la información recopilada en los dispositivos móviles.
- Servidor de base de datos *mySQL* para el almacenamiento de los datos recopilados de un modo ordenado.
- Entorno de desarrollo integrado *Netbeans* para facilitar la ejecución del programa en Java que se encargará de trasladar los datos de los ficheros a la base de datos de *mySQL*.

En primer lugar cabe destacar que los procesos descritos a continuación son los que se han realizado en el diseño del presente proyecto. No obstante, esto no quiere decir que no existan distribuciones alternativas para la instalación de estos servicios cuyo funcionamiento sea compatible con la aplicación desarrollada para Android.

En los siguientes subapartados se describirá de forma detallada la instalación de cada uno de estos servicios. Cabe destacar que la implementación del servidor en su conjunto se ha realizado sobre el sistema operativo Linux Ubuntu 12.04.

Servidor SFTP

La instalación de este servidor se va a realizar en una modalidad enjaulada. Es decir, en el servidor se va a crear un usuario para el acceso mediante SFTP con privilegios restringidos, de modo que sólo podrá acceder a un directorio específicamente creado para éste. Esta modalidad se aplica en aras de la seguridad del servidor, ya que se evitará que cualquier persona pueda acceder a todos los directorios del host que alberga al servidor pudiendo realizar alguna acción de tipo malicioso.

La distribución utilizada será '*opennSSH*' [67], una distribución gratuita con soporte para Linux que ofrece SFTP entre otros servicios.

Para ello se realizará el proceso descrito seguidamente:

- Desde la terminal de Ubuntu se solicitan permisos de usuario y se instala del paquete '*opensh-server*':

```
> sudo su
> apt-get install openssh-server
(Se confirma la instalación y se espera a que finalice)
```

- A continuación se crea el usuario SFTP incluyendo la contraseña y se le asocia un directorio '*/home*':

```
> groupadd hostusers
> useradd -g hostusers -d /home -s /sbin/nologin [nombre_usuario]
> passwd [nombre_usuario]
(Se escribe el password)
```

Donde aparece la etiqueta *[nombre_usuario]*, se escribirá el nombre que se desea que tenga el usuario SFTP. Habrá que tener en cuenta que este nombre se debe mantener a lo largo del todo el proceso en adelante para la correcta instalación del servicio, por lo que en la aplicación de Android también habrá que utilizar las mismas credenciales.

- Se abre el fichero de configuración por defecto para realizar unas modificaciones:

```
> gedit /etc/ssh/sshd_config
```

En este fichero se comentará la línea '*Subsystem sftp /usr/lib/openssh/sftp-server*' añadiéndole el carácter '#' delante. La línea debe quedar finalmente así:

```
#Subsystem sftp /usr/lib/openssh/sftp-server
```

Y al final del fichero se debe añadir la siguiente línea:

```
Subsystem sftp internal-sftp
```

Es muy importante que se añada después de la última línea del fichero para que la configuración sea correcta.

Debajo de esta última línea incluida, se incluirán a su vez las siguientes líneas:

```
Match Group hostusers
ChrootDirectory /hosting/ %u
ForceCommand internal-sftp
```

Con ello se indica al servidor SSH cuál será el directorio enjaulado en el que se ofrece el servicio SFTP y se añadirá el usuario creado anteriormente.

Una vez realizadas estas modificaciones se podrá guardar y cerrar el fichero.

- Desde la terminal de nuevo se crearán los directorios que se han indicado en la configuración del servidor y se otorgarán al usuario SFTP los permisos necesarios para utilizar el directorio en el que se almacenan los ficheros:

```
> mkdir /hosting
> mkdir /hosting/[nombre_usuario]
> mkdir /hosting/[nombre_usuario]/home
> chown [nombre_usuario]:hostusers /hosting/[nombre_ususario]/home
```

- Para finalizar se reinicia el servicio para que se ejecute con la nueva configuración:

```
> service ssh restart
```

Una vez finalizados estos pasos, el servidor SFTP estará correctamente instalado y preparado para su utilización.

Instalación del servidor de base de datos *mySQL*

Junto con la instalación del propio servidor se base de datos, se va a incluir también un servidor HTTP que permitirá posteriormente gestionar la base de datos desde una interfaz *web* de forma intuitiva.

Como distribución de servidor HTTP se ha utilizado *Apache* [71], y para la base de datos *mySQL* [69], ambos gratuitos. Además se va a utilizar la herramienta *phpMyAdmin* [72], que servirá para el acceso a la base de datos a través de la interfaz *web* mencionada.

En adelante se listan los pasos a seguir para la correcta instalación de estos servicios:

- Se solicitan permisos de superusuario y se instala el paquete 'apache2' y el módulo 'php5':

```
> sudo su
> apt-get install apache2 (Se confirma la instalación y se espera a que finalice)
> apt-get install php5 libapache2-mod-php5
(Se confirma la instalación y se espera a que finalice)
```

Se podrá comprobar si el servidor HTTP se ha instalado de forma satisfactoria accediendo a la siguiente dirección desde un navegador *web*:

http://localhost

- A continuación se reinicia el servidor:

```
> /etc/init.d/apache2 restart
```

El directorio por defecto de este servidor HTTP será '*/var/www*'.

- Se otorga permiso al usuario de Ubuntu para manipular el contenido del directorio y se le dan permisos de lectura y ejecución a todos los usuarios y de escritura sólo al propietario:

```
> chown -R [usuario]:www-data /var/www
> chmod -R 755 /var/www
```

Donde *[usuario]* se sustituye por el nombre del usuario del sistema operativo que actualmente se está utilizando.

- Se procede a instalar el servidor y el cliente de myQSL necesarios para la creación de la base de datos:

```
> apt-get install mysql-server mysql-client
```

Antes de finalizar la instalación se demandará una contraseña para el usuario 'root', que tendrá todos los permisos sobre la base de datos. Tras rellenar estos datos, se finaliza la instalación y ya se tiene instalado correctamente el servidor *mySQL*.

- A continuación, se deben instalar una serie de módulos adicionales para el correcto funcionamiento de los servicios instalados:

```
> apt-get install php5-mysql php5-curl php5-gd php5-idn php-pear php5-imagick php5-imap php5-mcrypt php5-memcache php5-ming php5-ps php5-pspell php5-recode php5-snmp php5-sqlite php5-tidy php5-xmlrpc php5-xsl
(Se confirma la instalación y se espera a que finalice)
```

- Se reinicia de nuevo el servidor 'Apache':

```
> /etc/init.d/apache2 restart
```

- Se instala el paquete 'phpMyAdmin':

```
> apt-get install phpmyadmin
```

Durante la instalación se podrá ver un diálogo en el que se deberá indicar el servidor *web* 'Apache2' para su configuración automática y posteriormente se indicará 'No' cuando se pregunte acerca de la configuración de la base de datos 'dbconfig-common'.

- Para que se pueda acceder a la interfaz que ofrece *phpMyAdmin*, será necesario configurar convenientemente los ficheros de configuración del servidor Apache. Para ello se accede al siguiente fichero:

```
> gedit /etc/apache2/httpd.conf
```

Inicialmente estará vacío. Únicamente habrá que escribir la siguiente línea y guardar el fichero:

```
Include /etc/phpmyadmin/apache.conf
```

- Hecho esto, se procederá a reiniciar por última vez el servidor apache:

```
> /etc/init.d/apache2 restart
```

Para comprobar el correcto funcionamiento de la interfaz *web* para la base de datos, se podrá acceder al siguiente enlace:

http://localhost/phpmyadmin

Y, si todo funciona conforme a lo previsto, deberá visualizarse la interfaz expuesta en la Figura A.1.1.

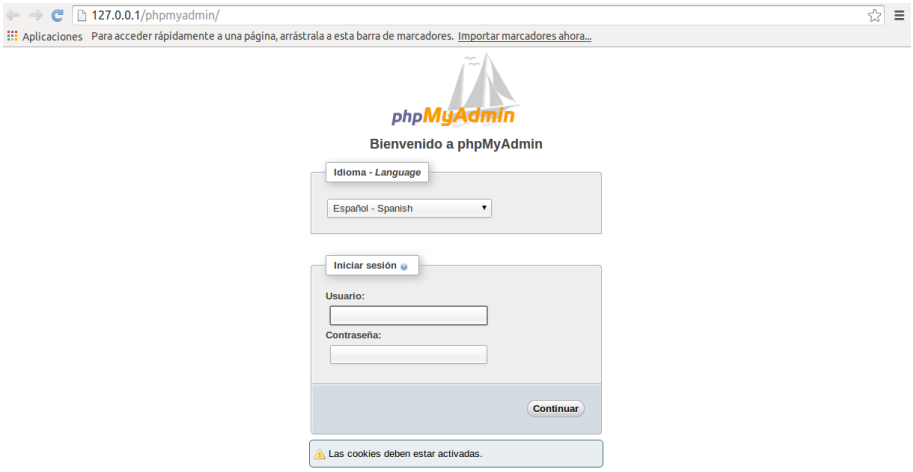

Figura A.1.1: Interfaz *web* de *phpMyAdmin*

Desde esta interfaz se podrá acceder a la base de datos introduciendo el usuario 'root' y la contraseña que antes se le ha asociado. La utilización de esta interfaz es bastante intuitiva, por lo que facilitará la gestión de la base de datos al mismo tiempo que el usuario se abstrae en gran medida del lenguaje *mySQL* que se está utilizando internamente.

Con ello ya se tiene toda la infraestructura necesaria para el funcionamiento correcto del servidor de base de datos y se podrá proceder a la utilización del programa desarrollado en Java para la creación y actualización automática de la base de datos.

No obstante, sería interesante realizar una configuración apropiada del acceso a la base de datos. Con el fin de no poner en peligro la seguridad del servidor ante distintos tipos de ataques desde el exterior, se ha decidido utilizar una estrategia conservadora. En la configuración diseñada habrá un único usuario 'root' con todos los permisos al que sólo se podrá acceder desde la propia máquina del servidor, evitando así accesos remotos a la base de datos que puedan tener intenciones maliciosas.

Para hacer realidad esta configuración bastará con configurar el acceso a la interfaz de *phpMyAdmin*, puesto que el propio servidor *mySQL* por defecto tiene la configuración que se desea para el usuario 'root'. Para configurar *phpMyAdmin* habrá que efectuar los pasos descritos seguidamente.

- Desde la terminal con permiso de superusuario, se accede al siguiente fichero de configuración:

```
> gedit /etc/phpmyadmin/apache.conf
```

En este fichero se añadirán las siguientes líneas inmediatamente después de la línea '<*Directory /usr/share/phpmyadmin*>':

```
Order
Deny,Allow Deny from All
Allow from 127.0.0.1
```

En estas líneas se restringe el acceso únicamente a la propia máquina del servidor identificada por la dirección IP 127.0.0.1.

Para que este último paso tenga efecto será necesario reiniciar por última vez el servidor http. Para ello se ejecutará el comando:

```
> /etc/init.d/apache2 restart
```

Con ello se consigue la configuración deseada, de modo que si alguien intentase conectarse de forma remota a la interfaz de *phpMyAdmin*, se mostraría el mensaje ilustrado en la Figura A.1.2, impidiendo el acceso del usuario.

Figura A.1.2: Acceso remoto restringido a *phpMyAdmin*

Instalación de *Netbeans*

Para la ejecución del programa en Java se ha elegido *Netbeans* [73], entre otros motivos porque se trata de una herramienta gratuita con un buen soporte. Sin embargo se podría utilizar cualquier otro entorno integrado de programación que acepte la compilación de programas en Java.

La instalación de *Netbeans* desde una distribución de Linux Ubuntu se puede realizar de forma sencilla desde la terminal mediante el siguiente comando:

```
> sudo apt-get install netbeans
```

Al finalizar la instalación de este programa se podrá ejecutar y utilizar con normalidad, ya que incorpora por defecto todos los módulos necesarios. Entre estos módulos se incluye *Java Development Kit* (JDK), necesario para compilar y ejecutar proyectos en Java.

Direccionamiento del servidor

Se debe garantizar el acceso al servidor desde Internet con la finalidad de que los dispositivos móviles puedan conectarse a éste para enviar mediante SFTP la información recopilada.

Habrá que tener en cuenta que, independientemente de si se utiliza una IP pública, un dominio que pueda resolverse mediante DNS o incluso un dominio basado en *Dynamic* DNS (para la gestión de IPs dinámicas), habrá que configurar correctamente la aplicación en Android indicando el dominio o la IP para que la conexión se realice satisfactoriamente. Esta configuración se especifica en el apartado A.2 de este anexo.

Si se desea implementar algún tipo de cortafuegos que filtre el acceso al servidor, se podrá tener en cuenta que únicamente es necesario el acceso al puerto 22 del servidor, que es el que utiliza SFTP por defecto. De esta forma, una configuración admisible sería aceptar sólo aquel tráfico entrante que se dirija a dicho puerto.

A.2. Manual de instalación de la aplicación Android

Antes de llevar a cabo la instalación de la aplicación de Android habrá que tener en cuenta las siguientes consideraciones:

- La aplicación sólo será compatible con dispositivos que tengan instalada la versión 3.0 de Android (Android Honeycomb) o superior. Esto se debe a que algunas funcionalidades de las que no se podía prescindir se incluyen por primera vez en la API 11 del SDK (*Software Development Kit*) de Android, que se aplica a la versión de Android mencionada.

- Habrá que configurar correctamente el acceso al servidor SFTP para que el envío de los datos recopilados se realice satisfactoriamente. Esto se debe a que habrá que indicar la IP de la máquina donde se aloja el servidor SFTP junto con el nombre de usuario y contraseña para acceder a éste. Esta configuración sólo será necesaria una vez tras la instalación de todos los servicios necesarios en el lado del servidor. Con ello la aplicación móvil queda convenientemente configurada mientras no se modifique la dirección IP del servidor SFTP o las credenciales de acceso. Una vez hecho esto se podrá compilar el proyecto y distribuir la aplicación.

Para la modificación del código fuente, se utilizará un entorno de programación apto para aplicaciones de Android. En el caso del presente proyecto, se ha determinado utilizar *Eclipse* [74] en base a que se considera la solución con mejor soporte en la actualidad además de tratarse de una herramienta gratuita. Este programa podrá instalarse de un modo sencillo siguiendo las directivas proporcionadas en el tutorial descrito en [75].

Desde el entorno de programación se debe acceder al fichero de código 'DescriptoresAplicación.java', donde se visualizará el extracto de código de la Figura A.2.1.

Las dos primeras variables se utilizarán respectivamente para activar (true) o desactivar (false) los modos '*debug* en la actividad del reproductor' y '*debug* en la actividad de la encuesta'. Encima de cada una de estas variables se puede observar una pequeña descripción de éstos. No obstante se recomienda leer detenidamente la descripción de estos modos proporcionada en el apartado 4.2.1 de esta memoria.

```
// Activar (true) o desactivar (false) el modo debug en el que se pueden
// observar los parametros monitorizados de red y del reproductor al mismo
// tiempo que se reproduce un vídeo
final static boolean DEBUG_REPRODUCTOR = false;

// Activar (true) o desactivar (false) el modo debug en el menú de encuesta
// en el que se pueden observar toda la información que sería enviada al servidor.
// IMPORTANTE: Al activar este modo no se envía la información al servidor.
final static boolean DEBUG_ENCUESTA = false;

// Tiempo minimo (en milisegundos) que debe de reproducirse el vídeo
// para que al pulsar el boton de salir se ofrezca la posibilidad
// de realizar la encuesta
final static long TIEMPO_DE_REPRODUCCION = 30000;

// Parametros para el envío de los archivos por SFTP
final static String USERNAME_SFTP = "youtubeqoe";
final static String PASSWORD_SFTP = "[redacted]";
final static String HOST_SFTP="youtubeqoe.noip.me";
final static String PATH_USUSARIO_SFTP = "/home";
```

Figura A.2.1: Variables de configuración de la aplicación

Al final del código de la Figura A.2.1 se pueden observar cuatro variables correspondientes a la configuración de la conexión con el servidor SFTP. Estas cuatro variables son las siguientes:

- *USERNAME_SFTP*: Nombre del usuario SFTP que se creó en el momento de configurar el servidor.

- *PASSWORD_SFTP*: Contraseña asociada al usuario SFTP que se creó en el momento de configurar el servidor.

- *HOST_SFTP*: Este campo debe rellenarse con un nombre de dominio o con una dirección IP que permita al dispositivo móvil acceder al servidor.

- *PATH_USUARIO_SFTP*: Esta variable corresponde a la ruta de ficheros donde se desea alojar la información recopilada en el dispositivo móvil. Si la configuración del servidor se ha realizado tal y como se describe en el apartado A.1 de esta memoria, debe dejarse el directorio por defecto ('*/home*'). Habrá que tener en cuenta que si se indica otro directorio, esto conllevará también una modificación en la configuración del programa de Java que realiza la actualización automática de la base de datos, ya que habrá que indicar la nueva ruta en la que se alojan los ficheros.

Una vez personalizada la aplicación que se desea instalar, se podrá compilar el proyecto dando lugar a la generación automática de un archivo con extensión '*apk*'. Este archivo es un ejecutable para el sistema operativo Android que permitirá instalar la aplicación en el dispositivo. Si se utiliza el entorno *Eclipse*, el archivo se ubicará en el directorio '*/bin*' dentro del directorio del proyecto y se denominará '*YouTubeQoE.apk*'.

Una vez identificado dicho archivo se procederá a la instalación de la aplicación en el terminal móvil. Para ello existen varios métodos alternativos, aunque aquí se presenta solamente el que se considera más sencillo:

- En primer lugar será necesario poder acceder a este archivo desde el dispositivo móvil. Para ello se puede almacenar en la memoria interna o externa del dispo-

sitivo, enviarlo mediante correo electrónico, bluetooth, etcétera. En definitiva se trata de que desde el dispositivo se pueda ejecutar el archivo.

- Una vez que el archivo es accesible de algún modo desde el terminal, se ejecutará pulsando sobre éste y debe aparecer en pantalla una interfaz similar a la mostrada en la Figura A.2.2. La interfaz puede variar algo en función de la versión de Android del dispositivo que se esté utilizando.

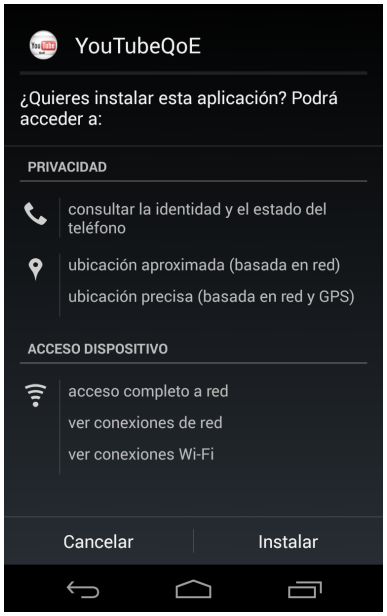

Figura A.2.2: Instalación de la aplicación 'YouTubeQoE'

En esta pantalla se informa de los permisos que el usuario deberá conceder a la aplicación. Para realizar esta concesión de permisos habrá que pulsar sobre el botón 'Instalar' y la aplicación se instalará automáticamente.

Eventualmente puede ocurrir que en el momento de ejecutar el archivo '*apk*', se bloquee la instalación y se muestre un aviso en el que se indica el riesgo de instalar aplicaciones de orígenes desconocidos. Este mensaje debería ser parecido al que se muestra en la Figura A.2.3.

Para solucionar este problema se debe pulsar sobre el botón de 'ajustes' y automáticamente se guiará hacia una pantalla de configuración que debe ser similar a la de la Figura A.2.4.

En este menú de seguridad se debe activar la opción que aparece rodeada con un recuadro en rojo, con la que se permite la instalación de aplicaciones de orígenes desconocidos.

Una vez realizado esto ya se podrá instalar la aplicación de un modo normal tal y como se ha descrito anteriormente y el usuario podrá comenzar a utilizarla.

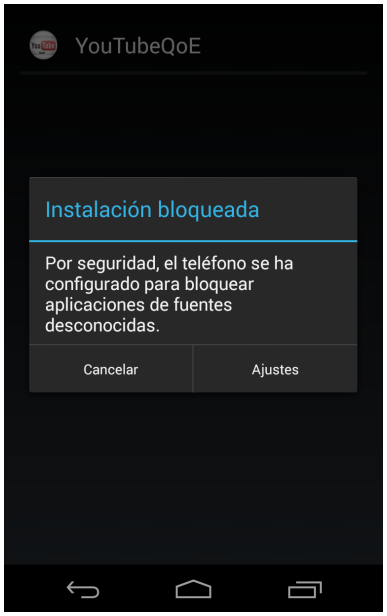

Figura A.2.3: Bloqueo de instalación de aplicación de origen desconocido

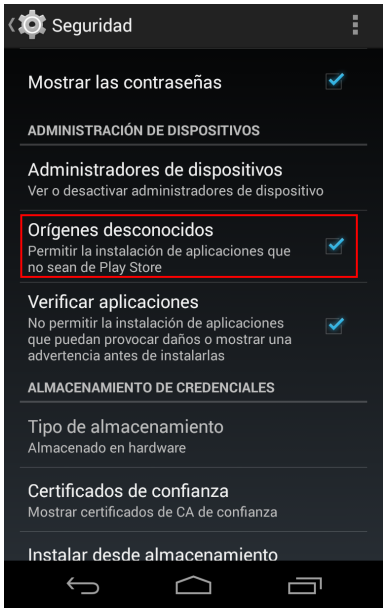

Figura A.2.4: Activación de aplicaciones de origen desconocido

Apéndice B

Manual de usuario

La finalidad de este anexo es recoger todos los aspectos relativos a la utilización de los recursos que se utilizan en el presente proyecto. Para ello, primero deberá realizarse la instalación satisfactoriamente siguiendo las instrucciones expuestas en el anexo A.

Este apartado se va a dividir en dos bloques principales. En el primero de ellos se pretende describir el uso del programa de actualización automática de la base de datos a partir de los ficheros que contienen la información. En el segundo bloque se describirá en términos generales cómo el usuario puede interactuar con la aplicación desarrollada para dispositivos móviles Android.

En ambos apartados se tratará de recoger toda la posible casuística que permitirá realizar fácilmente un diagnóstico ante un posible fallo en el sistema.

B.1. Manual de uso del programa de actualización de base de datos

Este programa ha sido diseñado para ser ejecutado en la misma máquina en la que se aloja la información recibida de los dispositivos móviles. Se trata de una rutina, que podría ser ejecutada periódicamente mediante un script o cuando el gestor del servidor lo desee, que desempeñará las siguientes tareas:

- Se conecta al servidor de base de datos alojado en la propia máquina o en otra máquina utilizando unas credenciales que deben indicarse previamente. El usuario que se utilice para el acceso a la base de datos deberá al menos tener permisos para añadir y consultar datos, y para consultar y crear bases de datos y sus correspondientes tablas dentro del mismo servidor.

- Se comprobará que exista la base de datos y si ésta tiene las tablas necesarias. En caso de que no exista, se creará automáticamente junto con sus tablas.

- Una vez realizada la comprobación de la base de datos, se procederá a la lectura de los ficheros con la información que aún no ha sido insertada en la base de datos y se añadirán todos los campos nuevos para actualizar la base de

datos. Esta base de datos sigue una estructura de base de datos relacional que permite almacenar los datos de una forma ordenada y eficiente.

- Tras realizarse todas las tareas descritas, se desconectará automáticamente de la base de datos.

La utilización de este programa es bastante automática en tanto que sólo se necesita la creación del servidor de base de datos *mySQL* y de un usuario con el que se pueda acceder a éste. Una vez se tiene esto, el resto del trabajo se realizará de forma transparente para el gestor del servidor.

Únicamente habrá que ejecutar el programa de Java a través de entorno de programación integrado o a partir de los ficheros compilados a través de la terminal.

Antes de llevar a cabo la ejecución deberá revisarse la configuración de dicho programa accediendo al fichero '*DescriptoresAplicacion.java*'.

```
//Nombre de la base de datos en la que se desea almacenar la informació
final static String NOMBRE_BD = "youtube";

//Usuario que se va a utilizar para el acceso a la base de datos
//Por defecto se utiliza el usuario "root" que dispone de todos
//los permisos
final static String USUARIO = "root";

//Contraseña asociada al usuario indicado para el acceso a la
//base de datos
final static String PASSWORD = "[redacted]";

//Directorio en el que se encuentran los ficheros con la informacion
//que se desea almacenar en la base de datos
final static String RUTA_FICHEROS = "/hosting/youtubeqoe/home";

//Host en el que se aloja el servidor de base de datos. Por defecto
//el host es "localhost" siempre que el servidor de base de datos se
//aloje en la misma máquina donde se encuentran los fichero de datos
final static String HOST = "localhost";
```

Figura B.1.1: Variables de configuración del programa de creación y actualización de la base de datos

Como se puede observar en la Figura B.1.1, correspondiente al código de '*DescriptoresAplicacion.java*', estas variables incorporan una pequeña descripción encima indicando su significado y recomendaciones de valores que deben dejarse por defecto.

No obstante se describen a continuación dichas variables con mayor detalle:

- *NOMBRE_BD*: Nombre de la base de datos que será creada automáticamente y sobre la que se realizarán posteriormente actualizaciones para insertar la información nueva que se recopila en el servidor.

- *USUARIO*: Nombre de usuario con el que se conectará el programa a la base de datos para realizar las labores oportunas. Se recomienda utilizar el usuario '*root*' creado por defecto, ya que se trata de un usuario que dispone de todos los

permisos existentes y por lo tanto se evitarán problemas debidos a la restricción de permisos. Podría también utilizarse otro usuario que disponga de permisos para consultar las bases de datos existentes en el servidor, para la creación de nuevas bases de datos y tablas para éstas y para la consulta e inserción de datos.

- *PASSWORD*: Contraseña asociada al usuario que se ha indicado en la variable '*USUARIO*'.

- *RUTA_FICHEROS*: Indica la ruta hacia el directorio en el que se encuentran todos los ficheros con la información recopilada en el servidor SFTP. Si se han seguido literalmente los pasos indicados en el Anexo A para la instalación del servidor SFTP, se debe dejar la ruta establecida por defecto.

- *HOST*: Nombre o dirección IP de la máquina en la que se ubica el servidor de base de datos. Típicamente el servidor de base de datos se encontrará en la misma máquina en la que se encuentra la información recopilada del servidor SFTP, por lo que por defecto esta variable será '*localhost*'. Sin embargo, si el servidor de base de datos se ubica en otra máquina distinta, se podrá acceder a ella indicándolo mediante esta variable.

Una vez configurado el programa convenientemente, se podrá proceder a la ejecución de éste. Si la ejecución se realiza de un modo correcto, se podrán ver por pantalla los mensajes descritos abajo:

- En primer lugar, se indican en una lista todos los ficheros que se encuentran en el directorio del servidor SFTP. Habitualmente se podrán observar los cuatro ficheros que se envían conjuntamente en cada reproducción de vídeo con un mismo identificador. Un ejemplo sería el de la Figura B.1.2.

```
Ficheros encontrados:
12072014_173644Id=1189164501-DescripcionVideo.txt
12072014_173644Id=1189164501-MonitorizacionReproductor.txt
12072014_173644Id=1189164501-MonitorizacionRed.txt
12072014_210739Id=1189164501-Encuesta.txt
                              ⋮
```

Figura B.1.2: Ficheros encontrados en el directorio de SFTP

- A continuación se conecta al servidor de base de datos, se comienza la inserción de los datos que aún no se habían subido, y por último se desconecta de la base de datos. En cada una de estas etapas se podrá observar un mensaje por pantalla que indica el estado de ejecución. En la Figura B.1.3 se puede ver un ejemplo de una ejecución en la que se ha realizado satisfactoriamente el proceso de actualización de la base de datos.

```
Conectado a la base de datos "youtube"
Comenzando la inserción de datos...
¡Inserción de datos finalizada!
Desconectado de la base de datos "youtube"
```

Figura B.1.3: Ejemplo de conexión, inserción de datos y desconexión de la base de datos

Estos ejemplos expuestos muestran lo que ocurriría en una ejecución en la que todo funcione conforme a lo previsto. No obstante podrían ocurrir algunos errores a lo largo de la ejecución. Con el objetivo de que el gestor del servidor pueda realizar fácilmente un diagnóstico de estos, se presentan los fallos más comunes que se pueden producir.

Errores en tiempo de ejecución

A continuación se presentan los mensajes de error más recurrentes que pueden suceder en tiempo de ejecución en el programa.

- 'ERROR al conectarse a la base de datos'

  Este problema ocurre cuando no hay conectividad con el servidor de base de datos. Para solucionarlo se debe comprobar que el servidor se encuentra activo y que en la clase '*DescriptoresAplicacion*' se indica correctamente el *host* en el que se aloja el servidor. Igualmente, si el servidor se aloja en otra máquina, se debe comprobar que no existe ningún cortafuegos o sistema de filtrado que rechace la conexión.

- 'ERROR en la creación de la base de datos'

  Este problema sólo puede ocurrir en el caso de que la base de datos definida en '*DescriptoresAplicacion*' no exista y por lo tanto se intente crear de forma automática. El error en la creación de la base de datos típicamente puede deberse a que el usuario que se está utilizando no dispone de permisos necesarios, por ello se recomienda utilizar el usuario '*root*'.

  Si el problema persiste, se podría optar por crear esta base de datos manualmente mediante otras herramientas que lo permitan.

- 'ERROR: No existe el directorio de ficheros indicado'

  Este error indica que no existe el directorio donde se ubican los ficheros con la información que se desea insertar en la base de datos. Para solucionar este problema se debe revisar la ruta indicada en el fichero de configuración de la clase '*DescriptoresAplicacion*'.

- 'ERROR al insertar datos en la base de datos'

  Este error se produce cuando no ha sido posible realizar una inserción de datos en la base de datos correspondiente. Típicamente esto se deberá a que el usuario utilizado no cuenta con los permisos necesarios. Este error no se debe a un problema de conectividad, ya que llegado este punto de ejecución, el programa ya se ha conectado satisfactoriamente al servidor de base de datos.

- 'ERROR al cerrar la conexión con la base de datos'

  Este error indica que la conexión con la base de datos ya se había cerrado antes de lo previsto. Probablemente se deba a que se ha perdido la conectividad con el servidor. En caso de que esto ocurra, se recomienda comprobar la conectividad con el servidor de base de datos y ejecutar el programa para realizar de nuevo la inserción de datos.

Como se describe en el apartado 4.2.1 de esta memoria, en cada envío realizado desde un dispositivo móvil, se envían cuatro ficheros con información de distinta naturaleza. De este modo, en el lado del servidor se leerán secuencialmente los cuatros ficheros que tienen un mismo identificador para aunar la información e insertarla de forma ordenada en la base de datos. En este proceso también podrían ocurrir errores debidos a un mal *parseado* de dichos ficheros. Estos errores serían los siguientes:

- 'ERROR: Lectura de encuesta errónea
  Fichero: [nombre_fichero]'
- 'ERROR: Lectura de monitorizacionRed errónea
  Fichero: [nombre_fichero]'
- 'ERROR: Lectura de monitorizacionReproductor errónea
  Fichero: [nombre_fichero]'
- 'ERROR: Lectura de DescripcionVideo errónea
  Fichero: [nombre_fichero]'

Se trata de errores que no deberían ocurrir salvo extrañas ocasiones, debido a que la información enviada se ha normalizado para que esto no ocurra. Sin embargo, podría ocurrir que estos ficheros hayan sido manipulados manualmente o que en el envío hayan sido alterados. En caso de que esto ocurra, en el mensaje de error se indica el nombre del fichero en el que se ha producido el fallo. Por lo que podrá solucionarse simplemente con eliminar dicho fichero renunciando así a la información que éste contenía. Con ello sólo se perdería la información concreta de este fichero, mientras que la información del resto de ficheros del mismo experimento sí que sería incorporada, ya que el programa se ha diseñado con la robustez suficiente como para salvar esta problemática.

B.2. Manual de uso de la aplicación Android

El cometido de este apartado es explicar cómo puede interactuar el usuario con la aplicación de Android a través de la interfaz. Para ello se hará un recorrido a lo largo de todas las posibilidades que tiene ésta.

La aplicación consta de cuatro ventanas o actividades (como se denominan en la documentación oficial de Android) a través de las cuáles el usuario puede navegar.

La primera actividad muestra una interfaz como la de la Figura B.2.1.

En esta interfaz se podrá escribir en el campo marcado como '1' en la Figura B.2.1, una palabra o frase sobre la que se desee realizar una búsqueda de vídeos. Una vez escrito esto, se puede indicar el número máximo de resultados que se desean buscar mediante el elemento marcado como '2' en la Figura B.2.1. Este límite se utiliza con el fin de economizar en el consumo de datos del dispositivo móvil, ya que cuanto mayor sea el número de vídeos a buscar, mayor será el

Figura B.2.1: Actividad principal de la aplicación

volumen de datos que envíe el servidor de YouTube a la aplicación. El límite se encuentra en 20 vídeos por defecto y puede oscilar desde 1 a 50 vídeos en total, por lo que podrá ser personalizable a disposición del usuario.

También se podrá pulsar sobre el icono identificado como '3' en la Figura B.2.1, para que aparezca un menú desplegable en el que a su vez se puede observar un submenú denominado 'Acerca de. . . '. Si se pulsa sobre éste último, se desplegará un texto como el de la Figura B.2.2 en el que se informa al usuario sobre la finalidad de esta aplicación y los datos que se van a recopilar con el objetivo de motivar su participación en el proyecto y de que éste tenga conocimiento de que su privacidad queda protegida con el envío de datos anónimo.

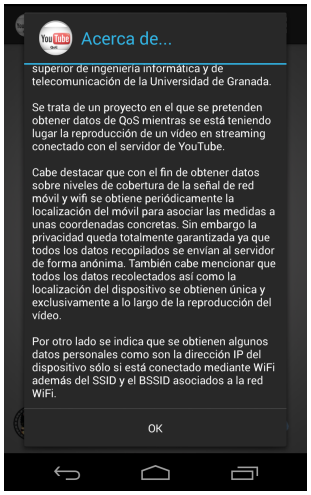

Figura B.2.2: Menú 'Acerca de. . . ' de la aplicación

Una vez elegido el término de búsqueda y el número máximo de vídeos, podrá iniciarse la búsqueda pulsando sobre el botón 'Buscar' y aparecerá una nueva actividad con un diálogo como el de la Figura B.2.3, en el que se informa de que la búsqueda de vídeos se encuentra en proceso.

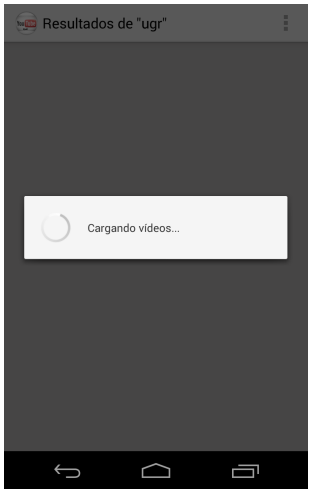

Figura B.2.3: Diálogo de proceso de cargando vídeos

Una vez finalizada la búsqueda de vídeos, se podrá observar una lista con todos los resultados encontrados de modo que el usuario podrá elegir cuál de ellos desea reproducir. En la Figura B.2.4 se puede observar un ejemplo de una lista de resultados de búsqueda.

Figura B.2.4: Lista de resultados de la búsqueda de vídeos

Al pulsar sobre uno de los vídeos de la lista de resultados, se abre una nueva actividad en la que se reproduce el vídeo seleccionado. En esta actividad la interfaz se ajusta automáticamente en función de si el dispositivo se encuentra

en posición vertical u horizontal. En la Figura B.2.5 se presenta un ejemplo de esta actividad en sus dos disposiciones posibles.

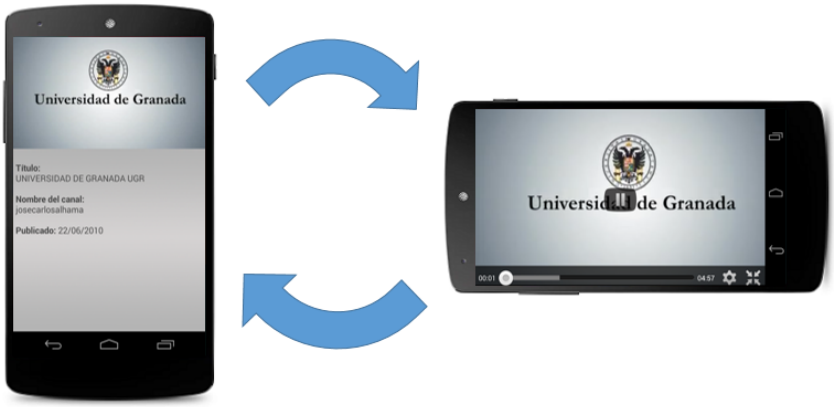

Figura B.2.5: Ejemplo de reproducción de vídeo vertical y horizontal

Una vez que se está reproduciendo el vídeo, existen diferentes alternativas por las que se puede pasar de esta actividad a otra. Se describe abajo la posible casuística: - Si se ha reproducido un tiempo igual o superior al umbral de tiempo preestablecido, al salir el usuario de la actividad pulsando el botón de 'atrás' propio del sistema operativo Android, se desplegará un diálogo en el que se le invita a realizar una encuesta. El tiempo umbral por defecto es de 30 segundos y puede modificarse siguiendo las directivas del Anexo A de esta memoria, concretamente en el apartado A.2. Este caso descrito queda ilustrado en la Figura B.2.6.

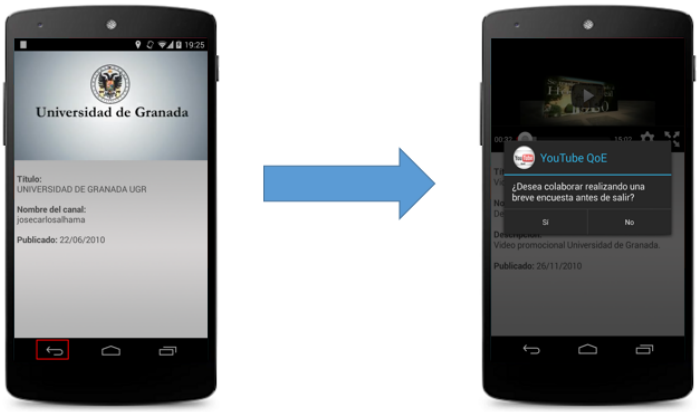

Figura B.2.6: Ejemplo de invitación de encuesta

- Si se ha reproducido un tiempo menor al umbral de tiempo establecido, al pulsar el botón 'atrás', se pasará a la actividad anterior en la que se listan los resultados encontrados con anterioridad. En la Figura B.2.7 puede observarse el comportamiento descrito.

Figura B.2.7: Ejemplo visualización de menos de 30 segundos

- Si en cualquier momento el usuario pulsa el botón '*home*' de Android o se sale de la aplicación mediante otro método que no sea el botón de 'atrás', la actividad se cierra automáticamente. Cuando la aplicación se vuelva a abrir, se podrá observar la actividad con la lista de resultados encontrados anteriormente. Este comportamiento se ha diseñado para cerciorarse de que sólo se invite a realizar una encuesta en el caso de que el usuario visualice la pantalla en todo momento durante la reproducción del vídeo. La Figura B.2.8 ejemplifica el caso de que se pulse el botón de '*home*' y se vuelva a abrir más tarde la aplicación de nuevo.

Figura B.2.8: Ejemplo al pulsar el botón '*home*' de Android

En el caso de que se invite al usuario a realizar la encuesta y este pulse en el botón 'Sí' del diálogo, se abrirá la actividad correspondiente a la encuesta. Esta encuesta tendrá una interfaz como la de la Figura B.2.9. En ella el usuario rellenará los datos que se demandan y podrá pulsar el botón de 'Enviar datos'. Es necesario rellenar todos los datos para que se posibilite el envío. Los datos de edad y sexo se almacenarán de modo que la siguiente vez que aparezca este formulario, se tendrán por defecto los valores que se rellenaron la última vez. Con ello se agiliza el proceso de rellenado del formulario ya que en cada dispositivo móvil típicamente será el mismo usuario el que utilice la aplicación.

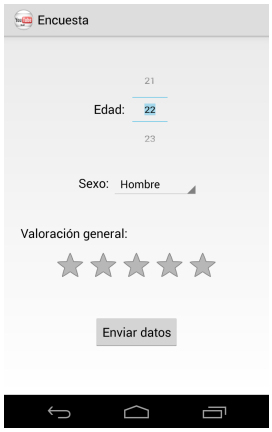

Figura B.2.9: Actividad de encuesta

Una vez que el usuario pulsa el botón para enviar los datos al servidor, se abre un diálogo en el que se muestra que el envío se encuentra en proceso. En caso de que finalmente el envío se realice satisfactoriamente, se volverá a la pantalla principal y se podrá leer por pantalla el mensaje '¡Gracias por realizar la encuesta!', como se puede ver en la Figura B.2.10.

Figura B.2.10: Envío de datos satisfactorio

Si por lo contrario el envío no finaliza satisfactoriamente por problemas de conectividad con el servidor, se mostrará por pantalla el mensaje 'Error: No se ha podido contactar con el servidor', como el indicado en la Figura B.2.11. En este caso sería conveniente que este fallo de conectividad se pusiera en conocimiento del gestor del servidor debido a que podría tratarse de un problema en el servidor SFTP.

Figura B.2.11: Error en el envío de datos

Una función adicional que se ha añadido durante el proceso de desarrollo de la aplicación es la opción de reproducir vídeos a partir del enlace *web* a éstos. Esta funcionalidad se ha implementado con el fin de aumentar potencialmente el uso de la aplicación, ya que estos enlaces se utilizan a menudo en redes sociales y aplicaciones de mensajería instantánea para compartir vídeos con otros usuarios. Con ello, ya no sólo es posible buscar vídeos en el servidor de YouTube, sino también reproducir vídeos de YouTube a partir de su enlace correspondiente. Una vez que se pulse sobre el enlace, se ofrecerá entre varias alternativas la opción de ejecutar la aplicación 'YouTubeQoE' y, al elegir ésta, automáticamente comenzará la reproducción del vídeo junto con los procesos de monitorización. Finalmente, se invitará al usuario a realizar la encuesta en caso de que se haya excedido el umbral de tiempo de reproducción y se haya pulsado el botón 'atrás', tal y como ocurría en el caso descrito anteriormente. Esta funcionalidad queda gráficamente descrita en la Figura B.2.12, donde se puede observar todo el proceso desde que se pulsa sobre el enlace *web* hasta el comienzo de la reproducción del vídeo.

Por último, cabe destacar también el supuesto de que el dispositivo no tenga conectividad de red al inicio, cuando el usuario pulsa el botón 'Buscar' para comenzar la búsqueda de vídeos. En este caso, para evitar que la aplicación quede bloqueada debido a que no hay acceso a Internet, se desplegará un diálogo que indicará al usuario que no existe ninguna conexión activa. En este diálogo se presenta un único botón en el que se puede leer 'Volver' y que cuando es pulsado hará que se vuelva a la actividad anterior donde se presenta el cuadro de búsqueda. Este caso descrito queda reflejado en la Figura B.2.13, donde se puede observar el diálogo mencionado.

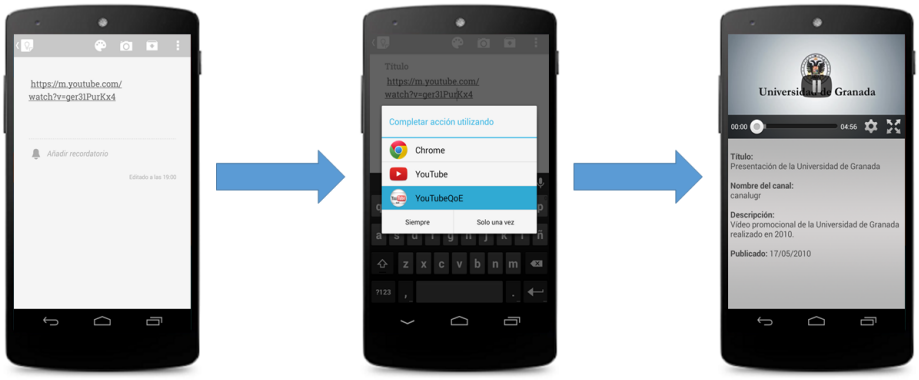

Figura B.2.12: Reproducción de vídeo desde enlace *web*

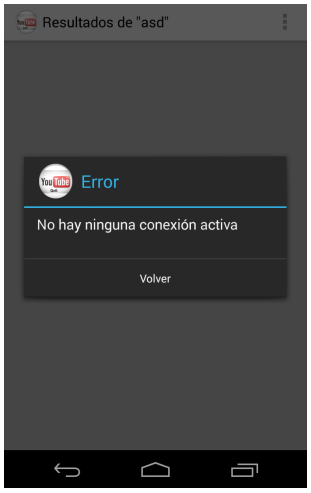

Figura B.2.13: Fallo de conectividad en el dispositivo móvil

Apéndice C

Obtención de trazas utilizando *TCPDump*

Este anexo se ha realizado con la finalidad de que el lector conozca cómo podría obtenerse una traza con todos los datos enviados y recibidos en el dispositivo móvil al mismo tiempo que se está reproduciendo un vídeo. Estas trazas pueden resultar de gran utilidad para analizar cómo se produce el tráfico, pudiendo analizar algunos patrones como ráfagas de envío o niveles de transmisión medios.

Cabe destacar que se trata de una funcionalidad ajena a las herramientas desarrolladas en el proyecto, ya que éstas son totalmente independientes del proceso que se describe a continuación. Resulta de gran relevancia informar además al lector de que para que todo este proceso tenga validez, el dispositivo móvil debe ser *rooteado* previamente de modo que se tengan privilegios de superusuario.

En un primer bloque se describirán todos los pasos previos para la instalación de la herramienta utilizada. Una vez completada la instalación, podrán realizarse capturas con el dispositivo móvil conectado a un ordenador mediante un cable USB, lo que será descrito en el segundo apartado de este anexo.

C.1. Instalación previa de *TCPDump*

Para la obtención de trazas se va a utilizar una distribución de *TCPDump*. Se trata de una herramienta de *sniffing* ampliamente conocida y de código abierto, por lo que en este caso se considera la mejor solución.

Para ello, se parte de un archivo binario que ha sido generado para dispositivos Android y que se adjunta con la documentación de este proyecto. Este binario será el que habrá que ejecutar para iniciar y terminar la captura de datos y generar un archivo con extensión '*.pcap*' que contendrá los resultados.

El primer requisito esencial antes de comenzar, será tener instalados los drivers necesarios para el dispositivo móvil y poder ejecutar desde la terminal la herramienta ADB (*Android Debugging Bridge*) incluida en el SDK de Android. Esta instalación dependerá de la marca del dispositivo móvil y del sistema operativo

del ordenador que se va a utilizar. Por ello, se presupone correctamente instalado y configurado, lo que no será de gran dificultad ya que existen multitud de tutoriales en Internet.

Una vez hecho esto se podrá comprobar que se ha realizado todo correctamente conectando el dispositivo móvil al ordenador mediante un cable USB y escribiendo en la terminal el siguiente comando:

```
> adb devices
```

Si el dispositivo está conectado y el ordenador lo ha reconocido, debe poderse leer un mensaje como el de la Figura C.1.1, donde se observa una lista con el identificador del dispositivo y a la derecha se puede leer '*device*'.

Figura C.1.1: Lista de dispositivos conectados

Una vez completada esta comprobación, se podrán realizar las labores necesarias para insertar el fichero binario en el dispositivo móvil. Este fichero se denomina 'tcpdump' y en un principio se almacenará en el directorio raíz de la memoria externa del dispositivo móvil. Para ello se debe ejecutar el comando:

```
> adb push [ruta_archivo_tcpdump] /sdcard/tcpdump
```

Donde la etiqueta *[ruta_archivo_tcpdump]* debe sustituirse por la ruta absoluta al fichero 'tcpdump' que se ubica en el ordenador. En la Figura C.1.2 se ejemplifica este proceso desde un ordenador con sistema operativo Windows, aunque igualmente podría realizarse desde un ordenador con cualquier otro sistema operativo.

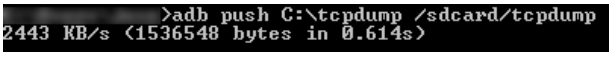

Figura C.1.2: Transferencia de archivo binario al dispositivo móvil

A continuación se abre un *shell* de Android, con el que se podrá gestionar desde la terminal el sistema operativo del terminal móvil. Esto puede realizarse desde el ordenador mediante el comando:

```
> adb shell
```

Se solicitarán permisos de superusuario mediante el comando:

```
> su
```

Para ello es necesario que el dispositivo esté *rooteado* y que posiblemente el usuario dé permiso en el móvil explícitamente para esta aplicación.

El archivo binario se introducirá en un directorio propio de Android, por ejemplo en el directorio '*/data/local*'. Para ello se copia el archivo al nuevo directorio y se borra de la tarjeta de memoria mediante los siguientes comandos:

```
> cp /sdcard/tcpdump /data/local/tcpdum
> rm /sdcard/tcpdump
```

Se le otorgan al fichero todos los permisos necesarios de lectura/escritura:

```
> chmod 777 /data/local/tcpdump
```

Con ello ya se ha completado todo el proceso necesario para poder utilizar *TCPDump* desde el móvil cuando se desee.

C.2. Capturar paquetes con *TCPDump*

Una vez que se ha completado todo el proceso de instalación de *TCPDump*, mediante la herramienta ADB de Android y con el dispositivo móvil conectado a un ordenador, se podrán realizar capturas de tráfico.

En relación al presente proyecto, la utilidad que esto podría tener sería realizar una captura en el momento en el que se está reproduciendo un vídeo para observar cómo se recibe el tráfico desde el servidor de YouTube. Para ello, antes de acceder a la aplicación desarrollada en este proyecto, se recomienda establecer una conexión de tipo *shell* al dispositivo. Esto podrá hacerse con el móvil conectado y escribiendo desde la terminal del ordenador los siguientes comandos:

```
> adb shell
> su
(Se aceptan los permisos de superusuario)
```

Se accede seguidamente al directorio donde se ha ubicado el fichero binario de *TCPDump*. Si se han seguido las instrucciones del apartado C.1, se puede acceder mediante el siguiente comando al directorio '*/data/local*':

```
> cd /data/local
```

Antes de llevar a cabo el proceso de captura, es necesario identificar el nombre de la interfaz de red que está activa en ese momento. Para ello se puede ejecutar el comando:

```
> netcfg
```

Tras ello aparecerá una lista con todas las interfaces del dispositivo móvil. La interfaz que tiene conectividad con Internet será aquella cuya IP es distinta de '0.0.0.0' o de la IP correspondiente a la interfaz de *loopback* (127.0.0.1). En la Figura C.2.1 puede verse un ejemplo en el que la interfaz activa es 'rmnet_usb0', cuya IP asociada es '10.83.222.186'.

Para comenzar finalmente la captura de paquetes se puede utilizar el siguiente comando:

```
> ./tcpdump –s 0 –i [nombre_interfaz] –w [fichero_pcap]
```

Donde *[nombre_interfaz]* debe sustituirse por el nombre de la interfaz de red de la que se desea capturar el tráfico. Por ejemplo, típicamente cuando el dispositivo móvil está conectado a una red Wi-Fi, esta etiqueta deberá sustituirse por 'wlan0'. La etiqueta *[fichero_pcap]* debe sustituirse por la ruta y el nombre del fichero *pcap* donde se almacenarán los resultados. Por ejemplo, para almacenarlo en el directorio raíz de la memoria externa puede escribirse '*/sdcard/captura.pcap*'.

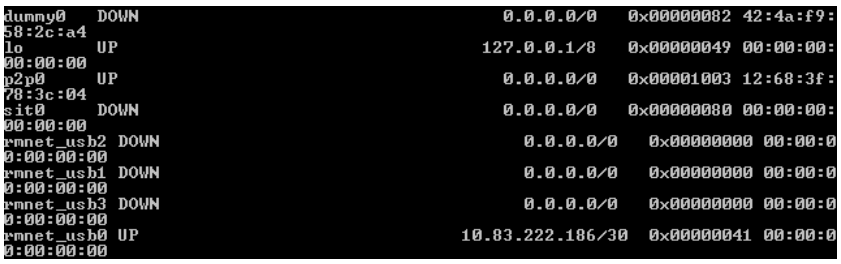

Figura C.2.1: Ejemplo de lista de interfaces del dispositivo

A modo de ejemplo aclaratorio, se presenta en la Figura C.2.2 una captura realizada sobre la interfaz 'wlan0' del dispositivo móvil.

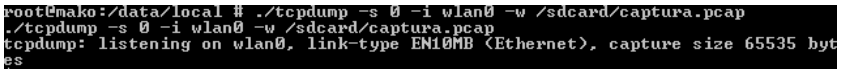

Figura C.2.2: Ejemplo de captura de tráfico

Este comando podría ejecutarse antes de realizar una reproducción de un vídeo y finalizarse al término de éste con sólo pulsar ctrl+C para finalizar el proceso. Con ello ya se tiene generado el archivo *pcap* en el directorio que se ha definido como destino. Este fichero podrá leerse mediante la ampliamente conocida herramienta *Wireshark* [76] de código abierto.

Para ello se podrá volcar dicho archivo a un ordenador en el que se tenga dicha herramienta instalada y podrá observarse la traza completa capturada. Un método rápido para enviar cualquier fichero del dispositivo al ordenador sería utilizando la funcionalidad *pull* de ADB. De este modo podría escribirse el siguiente comando:

> *adb pull [ruta dispositivo] [ruta_PC]*

Donde *[ruta_dispositivo]* debe sustituirse por la ruta absoluta al fichero *pcap* y *[ruta_PC]* por la ruta absoluta del directorio del ordenador donde se desea enviar el fichero.

Como ejemplo se presenta la ilustración de la Figura C.2.3.

Figura C.2.3: Transferencia de archivo *pcap* al ordenador

Cabe destacar que en la llamada que se ha realizado con *TCPDump* para iniciar la captura podrían realizarse algunas modificaciones a voluntad del usuario para modificar algunos parámetros que puedan resultar de interés. Para ello se recomienda revisar la documentación oficial de *TCPDump* [65] en la que se muestran todas las funcionalidades posibles.

Apéndice D

Consultas básicas en *mySQL*

En este anexo se recogen una serie de nociones básicas que informarán al lector sobre la realización de consultas en la base de datos. En estas consultas se permitirá solicitar un conjunto de información concreto y de forma ordenada, lo que facilitará en gran medida el análisis y procesado de datos.

Las bases de datos *mySQL* hacen uso del lenguaje SQL (*Structured Query Language*). Por ello será necesario conocer a grandes rasgos la sintaxis de este lenguaje si se desean realizar consultas.

D.1. Manual de consultas básicas

En primer lugar, resulta necesario explicar brevemente cómo realizar consultas utilizando el lenguaje SQL.

Todas las consultas se realizarán mediante una sentencia que comience con el comando '*SELECT*'. Seguidamente, se listan todos los campos de todas las tablas que se deseen consultar. Para ello se escribirá el nombre de la tabla al que corresponde el campo seguido de un punto y el nombre de éste. Por ejemplo, 'Tabla1.campo1'. Si se desean consultar todos los campos de una tabla se puede utilizar el carácter '*'. En este caso se podría escribir 'Tabla1.*'. Cada uno de los campos listados estará separado por comas.

Una vez indicados todos los campos que se desean consultar, se añadirá el parámetro '*FROM*' seguido de las tablas en las que se va a realizar la consulta. Se deben incluir todas las tablas implicadas en esta consulta, no sólo las que se citen después del comando '*SELECT*' sino también las referenciadas en el resto de la sentencia.

Por último podrá incluirse de forma opcional un comando '*WHERE*'. Este comando permitirá incluir condiciones sobre los datos que se van a consultar. Por ejemplo, en el caso de que se deseara sólo consultar aquellas entradas que tengan un contenido concreto dentro de un campo se podría añadir el comando $<$*WHERE tabla1.nombre='Juan'*$>$. De este modo sólo se mostrarían las entradas que cumplan que el campo 'nombre' de la tabla 'Tabla1' sea 'Juan'.

Cabe destacar que si se realiza una consulta sobre datos de dos tablas mediante un comando del tipo <*SELECT Tabla1.*, Tabla2.* FROM Tabla1, Tabla2*>, se listarán todas las posibles combinaciones que existan entre las entradas de la Tabla1 y de la Tabla2. Si se desea obtener sólo entradas que correspondan a un mismo identificador común habrá que utilizar una sentencia del tipo <*SELECT Tabla1.*, Tabla2.* FROM Tabla1, Tabla2 WHERE Tabla1.id = Tabla2.id*>. De este modo mediante el comando '*WHERE*' se presentará una tabla en la que los datos de cada entrada cumplirán la condición 'Tabla1.id = Tabla2.id'.

D.2. Ejemplos prácticos de consultas en la base de datos

Una vez definidas en términos generales las posibilidades que se pueden tener en una consulta en la base de datos, se desarrollan a continuación una serie de ejemplos prácticos. Estos ejemplos se considera que pueden ser recurrentes por parte del usuario de la base de datos. Además, a partir de estos ejemplos se podrán realizar algunas modificaciones que permitan adaptar la consulta a las necesidades del usuario sin necesidad de tener un amplio conocimiento sobre el lenguaje SQL.

En todos estos ejemplos se va a utilizar la interfaz *web* de *phpMyAdmin* que ha sido instalada en el servidor. No obstante las consultas se realizan utilizando el lenguaje SQL, por lo que las sentencias utilizadas en todos los ejemplos podrán extenderse a cualquier aplicación cliente de bases de datos *mySQL*.

En el caso de *phpMyAdmin* bastará con acceder a la base de datos sobre la que se desea realizar la consulta y hacer *click* en la pestaña SQL para poder escribir el comando que se ejecutará en la búsqueda. En la Figura D.2.1 se observan los pasos para acceder a este menú, donde '*youtube*' es el nombre de la base de datos que se ha elegido en este caso. Sin embargo, este nombre se podrá modificar en la configuración del programa de actualización de la base de datos. Para saber cómo acceder a esta interfaz *web* y la configuración de la base de datos, se puede consultar la documentación de Anexo A y Anexo B de esta memoria.

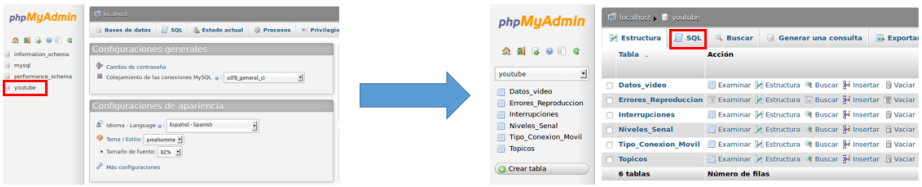

Figura D.2.1: Acceso al menú SQL en *phpMyAdmin*

Consulta sobre los datos de un experimento concreto

Como se indica en el apartado 4.4 de esta memoria, en todas las tablas de la base de datos se utiliza un campo denominado '*idDatos_video*' que permitirá

identificar unívocamente el experimento al que pertenece la entrada de la tabla. Ello permitirá que se puedan consultar datos relativos a un único experimento.

Por ejemplo, para conocer las interrupciones que se han producido en un experimento concreto se podría utilizar el siguiente comando:

```
> SELECT Interrupciones.* FROM Interrupciones WHERE Interrupciones.idDatos_video=[id_experimento]
```

Donde *[id_experimento]* corresponderá al identificador único del experimento rodeado por comillas simples.

Si en lugar de realizar la consulta sobre la tabla '*Interrupciones*' se desea realizar la consulta sobre cualquier otra tabla, sería sencillo modificar este comando. Únicamente habría que sustituir el nombre de la tabla '*Interrupciones*' por el nombre de la nueva tabla.

En la Figura D.2.2 se muestra un ejemplo en el que se listan las interrupciones del experimento cuyo identificador es '25082014_145030Id=-251852373'. En este caso la reproducción del vídeo sólo ha sufrido una interrupción. Para ello se ha utilizado el comando:

```
SELECT Interrupciones.* FROM Interrupciones WHERE Interrupciones.idDatos_video='25082014_145030Id=-251852373'
```

ID_interrupcion	idDatos_video	Inicio	Fin	Duracion_interrupcion
1	25082014_145030Id=-251852373	1408970987895	1408970991512	3617

Figura D.2.2: Consulta sobre las interrupciones de un experimento

Consulta de los experimentos de un mismo dispositivo

En esta consulta se va a proceder a realizar una consulta en la que se puedan observar todos los datos de la tabla '*Datos_video*' que pertenezcan a un mismo dispositivo móvil. Esto se podrá conseguir gracias al campo '*Id_Dispositivo*' que identifica a un dispositivo de forma única.

Con ello podría utilizarse el comando:

```
SELECT Datos_video.* FROM Datos_video WHERE Datos_video.Id_Dispositivo=[id_dispositivo]
```

Donde *[id_dispositivo]* corresponderá al identificador único del dispositivo móvil rodeado por comillas simples.

En la Figura D.2.3 puede observarse la consulta realizada sobre el dispositivo cuyo identificador es '1189164501'. En esta consulta, en lugar de listar todos los campos de la tabla '*Datos_video*', sólo se obtendrán el identificador del experimento, el número total de interrupciones y la valoración del usuario (MOS). Para ello se ha utilizado el siguiente comando:

```
SELECT Datos_video.idDatos_video, Datos_video.numeroInterrupciones, Datos_video.valoracion FROM Datos_video WHERE Datos_video.Id_Dispositivo = '1189164501'
```

idDatos_video	numeroInterrupciones	valoracion
13082014_130855Id=1189164501	0	5
13082014_131346Id=1189164501	0	4
13082014_195455Id=1189164501	0	3.5
19082014_172918Id=1189164501	0	3.5
19082014_174220Id=1189164501	10	1

Figura D.2.3: Consulta sobre experimentos de un dispositivo móvil

Podrían realizarse consultas en las que se imponga como condición el contenido de otros campos tales como la fecha de envío, hora de envío, número de interrupciones, etcétera. Para ello se podría modificar la condición impuesta después del comando '*WHERE*'.

Consulta incluyendo múltiples condiciones

Por último se muestra un ejemplo en el que se realiza una consulta más compleja. En este caso se trata de una consulta en la que se imponen dos condiciones sobre los datos a partir del comando '*WHERE*'.

En esta consulta se obtendrán sólo las interrupciones de los experimentos que superen más de dos interrupciones. Esto se traduce en dos condiciones: que coincidan los identificadores de los experimentos en las tablas '*idDatos_video*' e '*Interrupciones*', y que además el campo '*numeroInterrupciones*' sea mayor que 2. Para realizar esta consulta se ha utilizado el siguiente comando:

```
SELECT Interrupciones.* FROM Datos_video, Interrupciones WHERE Interrupciones.idDatos_video=Datos_video.idDatos_video AND Datos_video.NumeroInterrupciones>2
```

En la Figura D.2.4 se muestra el resultado de este ejemplo. En este caso se ha encontrado un único experimento que tiene un total de 5 interrupciones, el resto de experimentos de la base de datos tienen 2 o menos interrupciones.

ID_interrupcion	idDatos_video	Inicio	Fin	Duracion_interrupcion
1	19082014_174220Id=1189164501	1408462524326	1408462539438	15112
2	19082014_174220Id=1189164501	1408462551432	1408462557389	5957
3	19082014_174220Id=1189164501	1408462563444	1408462569395	5951
4	19082014_174220Id=1189164501	1408462577434	1408462598459	21025
5	19082014_174220Id=1189164501	1408462601432	1408462618461	17029

Figura D.2.4: Consulta de información de interrupciones de experimentos con más de dos interrupciones

Bibliografía

[1] Real academia de ingeniería, "Calidad de sevicio." [cited 2014 May 14]. Available from: `http://diccionario.raing.es/es/lema/calidad-de-servicio-0` .

[2] E. Gallo, M. Siller, and J. Woods, "An ontology for the quality of experience framework," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pp. 1540–1544, Oct 2007.

[3] k. Mintauricks, "Empirical studies of quality of experience (QoE) - a systematic literature survey," Master's thesis, University of Oslo, 2010.

[4] T. M. O'Neill, "Quality of experience and quality of service for ip video conferencing," *White Paper by Polycom*, 2002.

[5] ITU-T Recommendation, "ITU-T rec. p.800.1: New appendix I - definition of quality of experience (QoE)." [cited 2014 May 14]. Available from: `http://diccionario.raing.es/es/lema/calidad-de-servicio-0` .

[6] M. N. Zapater and G. Bressan, "A proposed approach for quality of experience assurance for IPTV," *First International Conference on the Digital Society*, 2007.

[7] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," *RFC 1633*, 1994.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *Internet Standards Track RFC 2475, IETF*, 1998.

[9] E. Rosen, A. Viswanath, and R. Callon, "Multiprotocol label switching architecture (MPLS)," *IETF RFC 3031*, 2001.

[10] ITU-T Recommendation, "ITU-T rec. p.800.1: Mean opinion score (MOS) terminology," 2003.

[11] M. Alreshood and J. Woods, "Survey on QoE/QoS correlation models for multimedia services," *International Journal of Distributed and Parallel Systems (IJDPS) Vol.4, No.3*, 2013.

[12] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network, vol. 24, no. 2, pp. 36–41*, 2010.

[13] T. Hossfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," *IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE), Dana Point, CA, USA*, 2011.

[14] Cisco Corporation, "Cisco visual networking index: forecast and methodology, 2012-2017." [cited 2014 May 14]. Available from: `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf` .

[15] Alexa Corporation, "The top 500 sites on the web." [cited 2014 June 7]. Available from: `http://www.alexa.com/topsites` .

[16] YouTube, "YouTube press room." [cited 2014 June 7]. Available from: `https://www.youtube.com/yt/press/es/statistics.html` .

[17] Cisco Corporation, "Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018." [cited 2014 June 6]. Available from: `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf` .

[18] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos," *Proc. IEEE Int. Workshp Quality of Service (IWQoS), pp. 229–238*, 2008.

[19] A. Finamore, M. Mellia, and M. M. Munafo, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," *Proc. ACM IMC*, 2011.

[20] J. J. Ramos, J. Prados-Garzon, P. Ameigeiras, J. Navarro, and J. M. Soler, "Charasteristics of mobile YouTube traffic," *IEEE Wireless Comunnication*, 2008.

[21] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. Munafo, and S. Rao, "Dissecting video server selection strategies in the YouTube CDN," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 248–257, 2011.

[22] R. Mok, E. Chan, and R. Chang, "Measuring the quality of experience of HTTP video streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 485–492, 2011.

[23] Cisco Corporation, "Cisco visual networking index forecast highlights." [cited 2014 June 8]. Available from: `http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html` .

[24] Statista, "Forecast: number of public hotspots worldwide 2009-2015." [cited 2014 June 8]. Available from: `http://www.statista.com/statistics/218596/global-number-of-public-hotspots-since-2009` .

[25] Statista, "Forecast: number of private hotspots worldwide 2009-2015." [cited 2014 June 8]. Available from: `http://www.statista.com/statistics/218601/global-number-of-private-hotspots-since-2009` .

[26] Statista, "Number of mobile users worldwide between 2010 and 2020." [cited 2014 June 8]. Available from: http://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010 .

[27] Statista, "Number of daily activations of android devices from august 2010 to april 2013." [cited 2014 June 12]. Available from: http://www.statista.com/statistics/278305/daily-activations-of-android-devices .

[28] Statista, "Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 4th quarter 2013." [cited 2014 June 12]. Available from: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems .

[29] J. Howe, "The rise of crowdsourcing," *Wired magazine 14(6)*, 2006.

[30] E. Estellés-Arolas and F. González Ladrón-de Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information Science (JCR: 1,41)*, 2012.

[31] F. Delli Priscoli, V. Suraci, A. Pietrabissa, and M. Iannone, "Modelling quality of experience in future internet networks," in *Future Network Mobile Summit (FutureNetw), 2012*, pp. 1–9, 2012.

[32] K.-T. Chen, C.-C. Tu, and W.-C. Xiao, "Oneclick: A framework for measuring network quality of experience," in *INFOCOM 2009, IEEE*, pp. 702–710, 2009.

[33] University of Pisa, "Portolan network sensing architecture." [cited 2014 April 28]. Available from: http://portolan.iet.unipi.it .

[34] Google Play, "Portolan network tools." [cited 2014 April 28]. Available from: https://play.google.com/store/apps/details?id=it.unipi.iet.portolan.traceroute .

[35] University of Cambridge, "Device analyzer for android." [cited 2014 April 28]. Available from: http://deviceanalyzer.cl.cam.ac.uk .

[36] Google Play, "Device analyzer." [cited 2014 April 28]. Available from: https://play.google.com/store/apps/details?id=uk.ac.cam.deviceanalyzer&hl=es .

[37] University of Berkeley, "Carat: Collaborative energy diagnosis." [cited 2014 April 28]. Available from: http://carat.cs.berkeley.edu .

[38] Google Play, "Carat." [cited 2014 April 28]. Available from: https://play.google.com/store/apps/details?id=edu.berkeley.cs.amplab.carat.android&hl=es .

[39] OpenSignal, "3G and 4G LTE cell coverage map." [cited 2014 April 28]. Available from: http://opensignal.com .

[40] Google Play, "OpenSignal." [cited 2014 April 28]. Available from: https://play.google.com/store/apps/details?id=com.staircase3.opensignal&hl=es .

[41] PressureNET website. [cited 2014 April 28]. Available from: http://pressurenet.cumulonimbus.ca .

[42] Google Play, "PressureNET." [cited 2014 April 28]. Available from: https://play.google.com/store/apps/details?id=ca.cumulonimbus.barometernetwork .

[43] Android Developers website. [cited 2014 May 30]. Available from: http://developer.android.com .

[44] Java platform, "Standard edition 7 API specification." [cited 2014 May 30]. Available from: http://docs.oracle.com/javase/7/docs/api .

[45] Google Developers, "YouTube Android player API." [cited 2014 July 22]. Available from: https://developers.google.com/youtube/android/player .

[46] Google Developers, "YouTube Data API (v3)." [cited 2014 July 23]. Available from: https://developers.google.com/youtube/v3 .

[47] Google Developers, "YouTube Analytics API." [cited 2014 July 23]. Available from: https://developers.google.com/youtube/analytics .

[48] Android Developers, "AsyncTask." [cited 2014 July 20]. Available from: http://developer.android.com/reference/android/os/AsyncTask.html .

[49] Android Developers, "Activity." [cited 2014 July 20]. Available from: http://developer.android.com/reference/android/app/Activity.html .

[50] Android Developers, "EditText." [cited 2014 July 20]. Available from: http://developer.android.com/reference/android/widget/EditText.html .

[51] Android Developers, "NumberPicker." [cited 2014 July 20]. Available from: http://developer.android.com/reference/android/widget/NumberPicker.html .

[52] Android Developers, "Bundle." [cited 2014 July 20]. Available from: http://developer.android.com/reference/android/os/Bundle.html .

[53] Android Developers, "LocationManager." [cited 2014 July 22]. Available from: http://developer.android.com/reference/android/location/LocationManager.html .

[54] Android Developers, "ConnectivityManager." [cited 2014 July 22]. Available from: http://developer.android.com/reference/android/net/ConnectivityManager.html .

[55] Android Developers, "WifiInfo." [cited 2014 July 22]. Available from: http://developer.android.com/reference/android/net/wifi/WifiInfo.html .

[56] Android Developers, "TelephonyManager." [cited 2014 July 22]. Available from: http://developer.android.com/reference/android/telephony/TelephonyManager.html .

[57] Google Developers, "YouTubePlayer.PlayBackEventListener." [cited 2014 July 22]. Available from: https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubePlayer.PlaybackEventListener?hl=es .

[58] Google Developers, "YouTubePlayer.PlayerStateChangeListener." [cited 2014 July 22]. Available from: https://developers.google.com/youtube/android/player/reference/com/google/android/youtube/player/YouTubePlayer.PlayerStateChangeListener?hl=es .

[59] Freebase website. [cited 2014 July 23]. Available from: https://www.freebase.com .

[60] Google Developers, "API console - Google code." [cited 2014 July 23]. Available from: https://code.google.com/apis/console .

[61] JCraft website, "JSch - Java Secure Channel." [cited 2014 July 23]. Available from: http://www.jcraft.com/jsch .

[62] Android Developers, "Android 3.0 APIs." [cited 2014 July 23]. Available from: http://developer.android.com/about/versions/android-3.0.html .

[63] Android Developers, "Manifest.permission." [cited 2014 July 23]. Available from: http://developer.android.com/reference/android/Manifest.permission.html .

[64] Google Developers, "Freebase API." [cited 2014 July 23]. Available from: https://developers.google.com/freebase .

[65] TCPDump website, "TCPDump documentation." [cited 2014 July 20]. Available from: http://www.tcpdump.org/#documentation .

[66] Android Developers, "Android 4.2 APIs." [cited 2014 July 6]. Available from: http://developer.android.com/about/versions/android-4.2.html .

[67] OpenSSH website. [cited 2014 July 6]. Available from: http://www.openssh.com .

[68] Fail2ban website. [cited 2014 July 6]. Available from: http://www.fail2ban.org/wiki/index.php/Main_Page .

[69] mySQL website. [cited 2014 July 9]. Available from: http://www.mysql.com .

[70] mySQL website, "mySQL connector Java." [cited 2014 July 20]. Available from: http://dev.mysql.com/downloads/connector/j .

[71] Apache website, "HTTP server project." [cited 2014 July 9]. Available from: https://httpd.apache.org .

[72] phpMyAdmin website. [cited 2014 July 9]. Available from: http://www.phpmyadmin.net .

[73] Netbeans website, "Netbeans IDE." [cited 2014 July 9]. Available from: https://netbeans.org .

[74] Eclipse website. [cited 2014 July 9]. Available from: https://www.eclipse.org/home/index.php .

[75] Sgoliver.net blog, "Entorno de desarrollo Android." [cited 2014 July 9]. Available from: http://www.sgoliver.net/blog/?p=1267 .

[76] Wireshark website. [cited 2014 July 20]. Available from: http://www.wireshark.org .