ICT-UNPA-201-2019
ISSN: 1852-4516
Aprobado por Resolución N° 0889/19-R-UNPA
http://doi.org/10.22305/ict-unpa.v11i2.783

Q2M, una librería para computar métricas de calidad en aplicaciones móviles

Q2M, a library to compute quality metrics in mobile applications

Ariel Machini[1], Juan Enriquez[2], Sandra Casas[3]
[1]*arielmachini@protonmail.com*, [2]*jenriquez@unpa.edu.ar*, [3]*scasas@unpa.edu.ar*
GISP, Instituto de Tecnología Aplicada, Unidad Académica de Río Gallegos
Universidad Nacional de la Patagonia Austral, Río Gallegos, Argentina

Recibido: 11/02/2019. Aceptado: 05/07/2019

RESUMEN

Las tecnologías móviles son cada vez más relevantes en los diferentes ámbitos de la vida cotidiana. A diario, personas de todas las edades descargan y utilizan aplicaciones móviles con diferentes propósitos (compras, comunicación, entretenimiento, etcétera). La calidad de la experiencia que brindan estas aplicaciones a sus usuarios se ve influenciada por diversos factores, por lo que resulta significativo estudiar los elementos que influyen en dicha calidad. Actualmente, existe una amplia variedad de herramientas que proponen métricas de calidad o que permiten la medición de diversas métricas de calidad en dispositivos móviles. Sin embargo, los conceptos de Calidad de Servicio (QoS) y Calidad de Experiencia (QoE) usualmente son tomados por separado, lo cual hace que el análisis completo de la calidad sea una tarea más difícil. Para dar solución a esta problemática, este trabajo presenta la librería Q2M, que contempla una amplia colección de métricas de QoS y de QoE. Esta se puede incorporar en cualquier aplicación desarrollada para la plataforma Android. Para probar el correcto funcionamiento de la librería desarrollada, esta se implantó en diferentes aplicaciones existentes y se realizaron múltiples pruebas con usuarios diferentes.

Palabras clave: Métricas; QoS; QoE; Aplicaciones móviles; Android.

ABSTRACT

Mobile technologies are increasingly relevant in the different scopes of daily life. Everyday, people of all ages download and use mobile applications with different purposes (shopping, communication, entertainment, et cetera). The quality of the experience that these applications provide to its users is influenced by several factors, so it's relevant to study the elements that affect that quality. Currently, there is a wide variety of tools that propose quality metrics or that allow the measurement of various quality metrics in mobile devices. However, Quality of Service (QoS) and Quality of Experience (QoE) concepts are usually taken separately; which makes the whole quality analysis a more difficult task. To provide this problem with a solution, this paper presents Q2M, a library that includes a wide assortment of QoS and QoE metrics. This library could be incorporated in any application developed for the Android platform. To test the correct functioning of the developed library, it was implemented in different existing applications and multiple tests with different users were carried out.

Key words: Metrics; QoS; QoE; Mobile applications; Android.

1. INTRODUCCIÓN

En el transcurso de la última década la masificación de la computación móvil fue acrecentando a tal ritmo que, hoy en día, «*hay más dispositivos móviles que personas en el mundo*» (Boren, 2014, párr. 1): es por esta razón que es evidente la importancia que esta tecnología tiene para la sociedad.

La calidad es un factor de gran importancia para el usuario a la hora de utilizar una aplicación móvil; Sin embargo, es común que dicha calidad se degrade debido a distintos factores contextuales tales como un bajo porcentaje de batería en el dispositivo que se está utilizando, la limitación de recursos (de memoria y CPU), canales físicos poco fiables y con ancho de banda limitado, movilidad de nodos, enrutamiento, condiciones físicas, entre otras. (Luo y Hyu, 2011; Devashish, 2013; Hefeeda y Hsu, 2010)

El concepto de calidad se puede separar en dos áreas más específicas, aunque estrechamente relacionadas: Calidad de Servicio (*QoS*) y Calidad de Experiencia (*QoE*). La QoS y la QoE proporcionan diferentes indicadores que permiten evaluar distintos aspectos de la calidad.

En el escenario actual, sumamente heterogéneo, variado y volátil, una de las áreas de mayor interés para la comunidad científica, los desarrolladores de aplicaciones móviles y las prestadoras de servicios en comunicaciones móviles inalámbricas es la calidad: La gestión de la calidad, el monitoreo, medición y evaluación de indicadores de calidad y la mejora de la QoS y de la QoE de los servicios móviles es un problema actual. Pese a que la literatura ha hecho distinciones entre QoS y QoE, comienza a existir un consenso respecto de que existe una relación causal entre QoS y QoE: si la QoS se ve afectada de manera negativa, entonces la QoE que el usuario de la aplicación perciba será deficiente.

Esta causalidad entre ambos factores existe debido a que la QoS ofrece la infraestructura subyacente de base para la medición de la QoE, y es por esto que también la QoS define los requerimientos no funcionales de un sistema, afectando de esta manera la calidad percibida de los resultados.

Existen algunos esfuerzos realizados en torno a esta problemática, que mediante diversas herramientas permiten recolectar información referente a la QoS o a la QoE, sin embargo, consideran a ambos conceptos por separado y/o no toman en cuenta ciertas métricas que brindan información importante para analizar la calidad en forma integral.

Este trabajo presenta la librería Q2M[1], que permite a desarrolladores de aplicaciones móviles para la plataforma Android obtener el valor de las métricas de QoS y QoE que consideren relevantes para su aplicación.

Este trabajo se organiza como se expone a continuación: En la Sección 2 se discute sobre los trabajos relacionados; en la Sección 3 se describen métricas e indicadores QoS y QoE móviles; en la Sección 4 se detalla el proceso de diseño y desarrollo de la librería; en la Sección 5 se exponen diversas pruebas realizadas con la librería y en la Sección 6 se desarrollan las conclusiones y el trabajo a futuro.

2. TRABAJOS RELACIONADOS

Android provee a desarrolladores una amplia variedad de artefactos (tales como rutinas, protocolos, herramientas y librerías con diversas funcionalidades) para la creación de aplicaciones móviles. Sin embargo, en la actualidad no ofrece herramientas para la medición y el análisis de la calidad (tanto QoS como QoE). En función de esta situación, a lo largo de los

[1] "Q2" representa a las dos áreas de la calidad que contempla la librería (QoS y QoE) y "M" representa a los dispositivos móviles.

últimos años se desarrollaron diversas aplicaciones y herramientas con la finalidad de cubrir este aspecto. A continuación se citan algunos de ellos:

Android QoS SDK (https://github.com/RestComm/android-QoS). Provee métricas sobre QoS/QoE (tales como geolocalización, servicio de voz y mensajería) aplicables a diversos aspectos del sistema operativo Android, así como también sobre aplicaciones. Proporciona analíticas con el fin de permitir su incorporación nativa en aplicaciones móviles o SDK.

Fastah network kit API (https://getfastah.com). Facilita una librería para Android que tiene la finalidad de realizar estimaciones sobre la calidad de la red (QoS) de alta precisión. Mide la latencia y la congestión utilizando servidores propios de alto rendimiento.

Android Network measures (https://github.com/APISENSE/android-network-measures). Brinda herramientas para llevar a cabo mediciones de red (QoS) en Android mediante el uso de utilidades como ping, traceroute y descargas/subidas por TCP/UDP.

QoE Probe for Android (https://github.com/farnazfotrousi/QoE-Probe-Android). Es una aplicación móvil Android que se puede integrar con otra aplicación para recolectar información relevante para la QoS y la QoE. La recolección de esta información tiene como objetivo estudiar las relaciones específicas entre las métricas de QoS y el impacto de la QoS en la QoE. Cabe destacar que también existe una variante para iOS de esta aplicación.

Netradar (https://www.netradar.org). Es una aplicación para medir y compartir la calidad de conexiones a Internet móviles. Soporta conexiones de datos móviles y Wi-Fi y los resultados que genera se pueden visualizar en un mapa. Este proyecto ofrece clientes de medición para múltiples sistemas operativos y los resultados de las mediciones se almacenan de forma centralizada. Los aspectos que esta aplicación mide son la velocidad de bajada y de subida de datos, la latencia y la intensidad de la señal.

Speedtest (https://www.speedtest.net). Es una aplicación multiplataforma para medir la velocidad y el rendimiento de una conexión a Internet haciendo uso de diversos servidores distribuidos alrededor del planeta. La versión para Android de la aplicación mide las velocidades de carga, de descarga y la latencia y ofrece gráficos en tiempo real que muestran la estabilidad de la conexión, mapas de cobertura de operadores móviles, así como también genera informes detallados de las mediciones realizadas.

nPerf (https://www.nperf.com). Es una herramienta multiplataforma que permite realizar mediciones de QoS sobre conexiones a Internet móviles (2G, 3G, 4G LTE y Wi-Fi). La versión para Android mide velocidades de carga y de descarga, velocidad de navegación y calidad de streaming. Permite comparar los resultados generados con los de otros usuarios y con los de otros operadores, dispone de un mapa interactivo para visualizar las mediciones de otros usuarios y la cobertura de operadores móviles y un monitor de red en tiempo real que permite ver la tasa de transferencia y el uso del plan de datos.

QoE Doctor (Chen et al., 2014). Es una herramienta que ofrece mediciones precisas, sistemáticas y repetibles sobre la QoE en aplicaciones móviles. Su objetivo es diagnosticar dicha QoE mediante el control automatizado de la interfaz de usuario y el análisis de capas cruzadas.

Se basa en la medición de métricas objetivas como lo son la latencia percibida por el usuario, el consumo de datos móviles y el consumo de energía. También cuenta con un conjunto de métricas específicas para aplicaciones tales como Facebook, YouTube y navegadores.

AppInsight (Ravindranath et al., 2012). Es un sistema que implementa binarios de aplicaciones móviles con la finalidad de identificar automáticamente la ruta crítica en las transacciones de los usuarios a través de los límites de llamadas asíncronas. Ayuda a los desarrolladores a diagnosticar fallos de rendimiento en sus aplicaciones y les indica las optimizaciones necesarias para mejorar la experiencia del usuario.

Timecard (Ravindranath et al., 2013). Es un sistema para estimar la QoE a partir del tiempo de respuesta de las aplicaciones a los usuarios. Se basa en el análisis de métricas tales como el tiempo que transcurre desde que el usuario inicia la solicitud y el tiempo que tardará en llegar la respuesta del servidor al cliente para que sea procesada.

Proteus (Xu et al., 2013). Es una herramienta enfocada en aplicaciones de tiempo real que recopila diversos datos acerca del rendimiento de la red (tales como el throughput y la pérdida de paquetes) de forma pasiva para, posteriormente, predecir el rendimiento futuro de dicha red haciendo uso de árboles de regresión.

Todas estas herramientas apuntan a medir la calidad de la red, pero tienen ciertas limitaciones y, por lo general, están enfocadas a considerar solamente el ancho de banda de la red, por lo que proporcionan valores absolutos sin realizar ningún tipo de análisis desde el punto de vista de los usuarios finales. Es por estas razones que los desarrollos aquí expuestos no contemplan ciertas cuestiones, tales como considerar a la QoE y a la QoS de manera integral; ciertas métricas para las cuales no dan soporte o su implementación como librería para que se puedan integrar fácilmente en otras aplicaciones.

3. MÉTRICAS E INDICADORES DE QOS Y QOE MÓVIL

Una métrica (medida) es un valor numérico o nominal asignado a características o atributos de un ente computado a partir de un conjunto de datos observables y consistentes con la intuición. Otra posible definición de métrica es una correspondencia o mapeo de un dominio empírico (mundo real) a un mundo formal (matemático). Por lo tanto, una medida es un valor numérico o nominal asignado al atributo de un ente por medio de dicha correspondencia o mapeo. (DeMarco, 1986)
Una métrica debe cumplir con ciertas características. Debe tener características matemáticas deseables y, cuando la métrica representa una característica que presenta un incremento frente a rasgos positivos o deseables y viceversa para el caso contrario, el valor de esa métrica debe incrementar o decrementar de la misma manera.
Una forma de clasificar las métricas es dividirlas en estáticas y dinámicas. Las métricas estáticas son utilizadas para medir las características estáticas de una aplicación, como el tamaño del código o la complejidad del mismo. Las métricas dinámicas permiten medir el comportamiento de la aplicación, como por ejemplo la usabilidad y fiabilidad, las cuales se calculan con la aplicación en ejecución. Cabe aclarar que las métricas no representan un fin por sí mismas, estas revelan datos e información sobre la experiencia personal del usuario cuando hace uso de una aplicación. La información obtenida de las métricas ayuda a realizar un mejor análisis y a tomar decisiones más acertadas con respecto al desarrollo o uso de una aplicación. (Tahir y Ahmad, 2010)

Calidad de Servicio (QoS)

La gestión de la QoS se puede considerar como un área especializada de la administración de sistemas distribuidos, y refiere a la supervisión y el control necesarios para garantizar que se alcancen y se mantengan las propiedades de calidad de servicio deseadas, que se aplica tanto a las interacciones continuas de los medios como a las interacciones discretas. (Blair y Stefani, 1997)

La QoS brinda los siguientes beneficios: Le otorga a los administradores control sobre recursos de red y les permite administrar la red desde una perspectiva de negocio en lugar de hacerlo desde una perspectiva técnica; Asegura que aplicaciones sensibles al tiempo y que aplicaciones de misión crítica tengan los recursos que necesitan, permitiendo también el acceso a la red a otras aplicaciones; mejora la experiencia del usuario (UX) y reduce los costos al utilizar los recursos existentes eficientemente, de esta manera retrasando o reduciendo la necesidad de expansiones o mejoras. (Gurwinder, 2011)

Al diseñar aplicaciones para plataformas móviles, el desarrollador debe enfocarse en los requerimientos no funcionales, entre los cuales se halla la QoS.

Por lo general, la QoS es requerida cuando las aplicaciones son sensibles al retardo, y en redes donde la capacidad es un recurso limitado, como en la comunicación mediante una conexión de datos móviles. La QoS es la capacidad para proveer diferentes prioridades a diferentes aplicaciones/flujos de datos/usuarios, así como también la capacidad para garantizar cierto nivel de rendimiento para determinado flujo de datos. (Svend y Jari, 1998)

Calidad de Experiencia (QoE)

La QoE se refiere a la aceptabilidad general de una aplicación o servicio, de acuerdo a la percepción subjetiva del usuario final: Se refiere a cómo el usuario percibe la calidad. De esta manera la QoE incluye todos los aspectos de un sistema de extremo a extremo (cliente, terminal, red, infraestructura de servicios, etcétera), donde la aceptabilidad general puede verse influenciada por las expectativas del usuario, así como también por el contexto. (Kuipers et al., 2010)

Las métricas correspondientes a la QoE se pueden subdividir en dos tipos: objetivas y subjetivas. Las métricas subjetivas se encuentran relacionadas a la opinión del usuario, que evalúa la calidad de la aplicación o del servicio en base a su experiencia con la aplicación (Ickin, 2015), dependiendo su evaluación de factores subjetivos tales como el contexto o las expectativas que tiene el usuario de la aplicación. Por lo general, el cálculo de métricas subjetivas se basa en la realización de cuestionarios o encuestas a los usuarios; Esto representa una dificultad debido a que cuesta tiempo y dinero. (Kim et al., 2008; Chen et al., 2014)

Con respecto a las métricas objetivas, se destaca su carácter sistemático, exacto y repetible, y representan diversas propiedades como el tiempo de presentación de los datos, el consumo de los datos móviles, el consumo de energía, la interfaz de usuario y la presentación del contenido. (Ickin, 2015)

A continuación (

Tabla 1) se detalla brevemente el conjunto de métricas QoE y QoS que este trabajo aborda:

Tabla 1 — *Métricas de QoE y QoS que se abordan en este trabajo.*

Métrica	Descripción	Tipo
Batería cargando	Estado de carga del teléfono (cargando/sin cargar).	QoE
Brillo de la pantalla	Porcentaje de brillo de la pantalla del teléfono del usuario.	QoE
Conexión a una red	Indicador de conexión del teléfono a una red (independientemente si tiene conexión a Internet o no).	QoE
Intensidad de la señal	Intensidad de la señal de la conexión en uso en dBm.	QoE
Latencia percibida por el usuario	Tiempo desde que el usuario dispara cierto evento hasta que recibe una respuesta.	QoE
Luz ambiental	Nivel de luz del entorno en el que se encuentra el usuario en lx (lux).	QoE
Porcentaje de carga de la batería	Porcentaje de carga actual de la batería del teléfono.	QoE
Proximidad	Distancia entre el dispositivo y el objeto más próximo a su sensor de proximidad.	QoE
Tipo de conexión	Tipo de conexión (Wi-Fi, 4G, 3G, GPRS, etcétera) a la que está conectado el dispositivo del usuario mientras utiliza la aplicación.	QoE
Uso de memoria	Cantidad de MB o porcentaje de la memoria RAM del teléfono que está en uso.	QoE
Uso del procesador por parte de la aplicación	Porcentaje de uso del procesador que está utilizando la aplicación.	QoE
Jitter	Valor del jitter (o fluctuación del retardo) en la comunicación a una dirección IP determinada.	QoS
Latencia	Tiempo de respuesta transcurrido en la comunicación a una dirección IP determinada.	QoS
Pérdida de paquetes	Porcentaje de pérdida de paquetes en la comunicación con una dirección IP determinada.	QoS

4. DISEÑO Y DESARROLLO DE LA LIBRERÍA Q2M

La librería desarrollada en este proyecto (Q2M) se compone de dos clases con funcionalidades diferentes: *Metricas* y *ConstructorXML*. Estas se muestran en el diagrama de clases de la Figura 1.

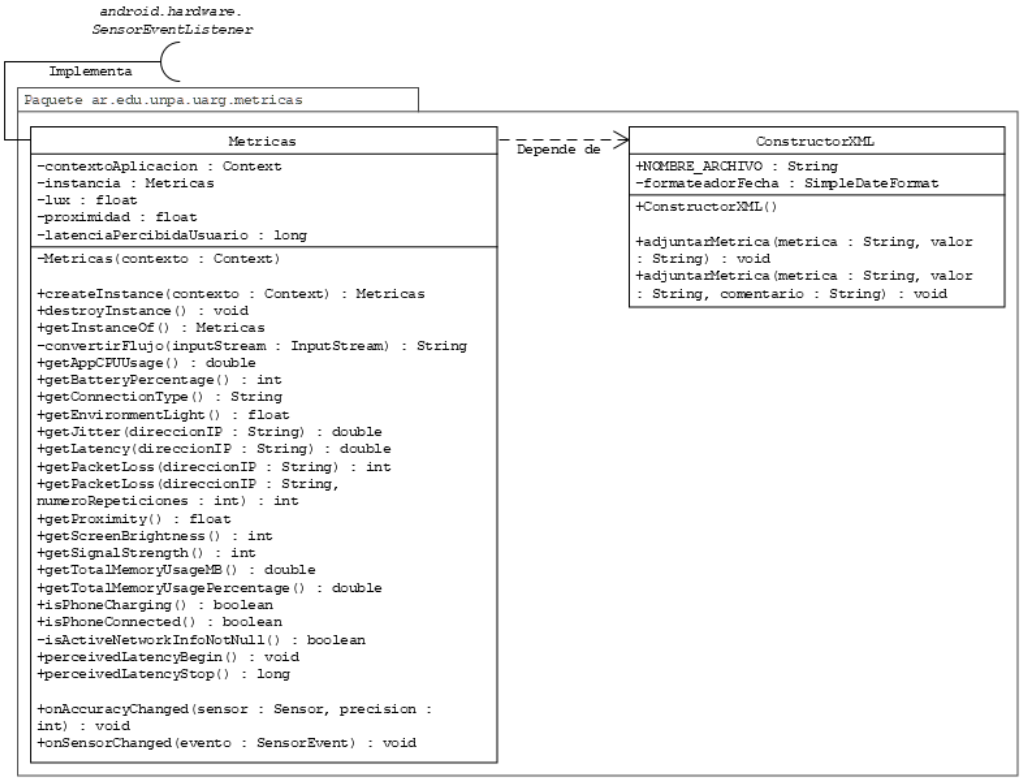

Figura 1 *— Diagrama de clases de la librería del proyecto de investigación*

La clase «*Metricas*» es el componente más importante de la librería. Se encarga de calcular los valores para todas las métricas de QoS y QoE que ofrece la librería, así como también de retornar dichos valores a la clase en donde se invoque y de llevar a cabo las funciones necesarias para registrarlos en un archivo XML.
Esta clase sigue el patrón de diseño *singleton*, por lo cual se maneja con una única *instancia estática* que se crea mediante el método *createInstance*, el cual recibe el contexto de la aplicación Android a la cual se le implanta la librería por parámetros; Esta instancia se puede obtener en cualquier momento (y en cualquier clase) invocando al método *getInstanceOf* y también se puede destruir mediante el método *destroyInstance*.
En la Figura 2 se detalla cómo la librería interactúa con el sistema operativo del teléfono móvil y, a su vez, con una aplicación Android una vez es implantada en esta.

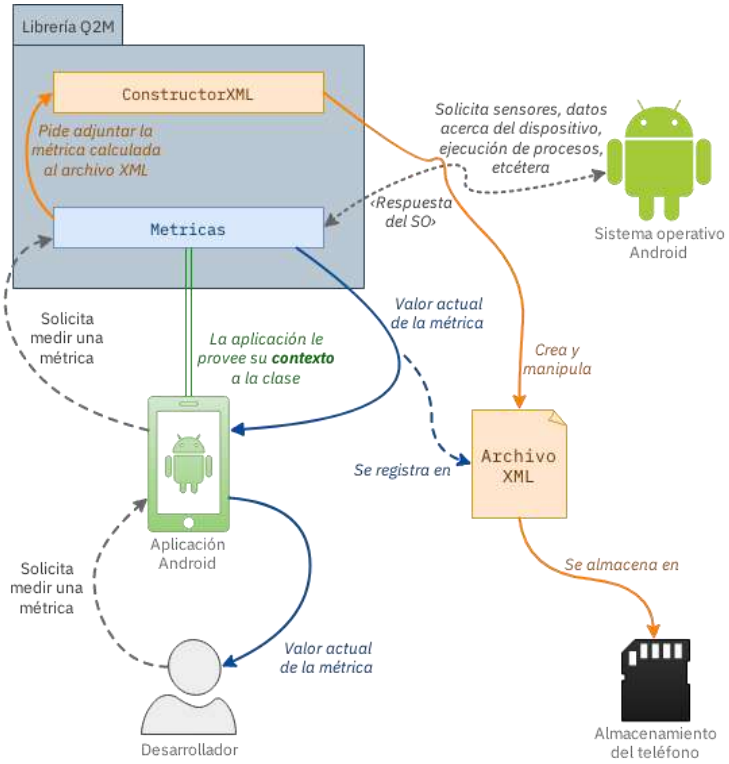

Figura 2 *— Diagrama funcional de la librería ya implantada en una aplicación Android.*

Para obtener el valor de la mayoría de las métricas, la clase «*Metricas*» depende del sistema Android, el cual facilita acceso a sensores, ejecución de comandos específicos y a una diversidad de información sobre el estado y el hardware del teléfono. Es por esto que, al introducir la librería en la aplicación, el desarrollador deberá agregar ciertos permisos al archivo «*AndroidManifest.xml*»:

```
<!-- Permisos requeridos para las métricas "intensidad de la señal" y "tipo de conexión" --
>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

<!-- Permiso requerido para las métricas "latencia", "jitter" y "pérdida de paquetes" -->
<uses-permission android:name="android.permission.INTERNET"/>

<!-- Permisos requeridos para poder manejar el archivo XML en el que se guardan las
métricas -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Fragmento de código 1 *— Permisos adicionales requeridos en el archivo «AndroidManifest.xml» para el correcto funcionamiento de la librería.*

La clase «*ConstructorXML*» es utilizada por la clase «*Metricas*» y tiene la responsabilidad de manipular el archivo XML (denominado «*metricas.xml*» por defecto): Si el archivo no existe, lo crea en el directorio raíz del teléfono del usuario; y si se ejecuta algún método para calcular una métrica, la clase se encarga de adjuntar el valor de salida de dicho método (mediante la función *adjuntarMetrica*) al archivo XML.
A continuación (Fragmento de código 2 y Fragmento de código 3) se muestra el formato con el que se almacenan las métricas:

```
<metrica identificador="Identificador de la métrica" fecha="Fecha de
generación">Valor</metrica>
```

Fragmento de código 2 *— Línea de ejemplo del archivo XML generado.*

Como se muestra en el diagrama de clases (Figura 1), la clase «*ConstructorXML*» también permite al desarrollador agregar un comentario (opcional) en forma de atributo al nodo:

```
<metrica identificador="Identificador de la métrica" fecha="Fecha de generación"
comentario="Comentario opcional del desarrollador">Valor</metrica>
```

Fragmento de código 3 *— Línea de ejemplo con comentario del archivo XML generado.*

En el Fragmento de código 4 se expone un ejemplo general en forma de pseudocódigo que deja ver la relación que existe entre las clases «*Metricas*» y «*ConstructorXML*» y, por ende, por qué «*Metricas*» depende de «*ConstructorXML*» (Figura 1):

```
clase METRICAS
	componente CONSTRUCTOR-XML

	[...]
	función CALCULO-METRICA(parámetros) devuelve un dato
		«Aquí se realizan las acciones necesarias para calcular el valor de
		la métrica, que se almacena en la variable
		VALOR-METRICA»

		CONSTRUCTOR-XML ejecuta ADJUNTAR-METRICA(NOMBRE-METRICA, VALOR-METRICA,
		COMENTARIO-OPCIONAL)

		devolver VALOR-METRICA
	fin de la función
	[...]
fin de la clase
```

Fragmento de código 4 *— Ejemplo del procedimiento que realiza la librería en pseudocódigo.*

Una vez terminada la codificación de las clases se procedió a generar un archivo JAR (*Java ARchive*) con la finalidad de compactar estas clases dentro de una *librería standalone*, logrando de esta manera que se pueda incorporar fácilmente la funcionalidad que ofrecen ambas clases a cualquier aplicación Android que cumpla con los requisitos mínimos para utilizarla. Para hacer esto se agregaron dos tareas dentro del archivo Gradle «*build.gradle*» del módulo *app* en el proyecto de Android Studio en el que se tenían las clases: *exportarJAR* y *eliminarJARAntiguo*. Estas tareas se exponen en el Fragmento de código 5 y el Fragmento de código 6:

```
task exportarJAR(type: Copy) {
	from('build/intermediates/intermediate-jars/release/')
	into('release/')
	include('classes.jar')
	rename('classes.jar', 'Q2M.jar')
}
```

Fragmento de código 5 *— Tarea exportarJAR del archivo build.gradle.*

```
task eliminarJARAntiguo(type: Delete) {
       delete 'release/Q2M.jar'
}
```

Fragmento de código 6 *— Tarea eliminarJARAntiguo del archivo build.gradle.*

La librería Q2M se distribuye como software libre y su código fuente está disponible públicamente en un repositorio en GitHub[2].

5. EXPERIENCIAS

Durante el desarrollo del proyecto de investigación se llevaron a cabo diversas pruebas con el objetivo de verificar el correcto funcionamiento de la librería.

Primer escenario: easyFTP

En este escenario se implantó la librería en la aplicación easyFTP[3], desarrollada por Adeel Ahmad (*linkindrew* en GitHub). A grandes rasgos, la aplicación permite conectarse a un servidor FTP para realizar simples operaciones de carga o descarga de archivos. EasyFTP hace uso de la clase FTPClient de Apache Commons[4]. Este escenario de prueba se llevó a cabo con la finalidad de verificar el correcto funcionamiento de la librería Q2M.
Para llevar a cabo las pruebas se utilizó una computadora portátil con GNU/Linux (detalles en la Tabla 2) a la cual se le instaló el paquete *proftpd* (Figura 3).

Tabla 2 *— Características de la portátil utilizada para alojar el servidor FTP.*

Lenovo Legion Y720	
Sistema operativo	Ubuntu 18.04 LTS 64-bit
Memoria RAM	16 GB (2400MHz PC4-19200 SDRAM SODIMM)
Procesador	Intel® Core™ i7-7700HQ (6M Cache, 2.8 GHz)

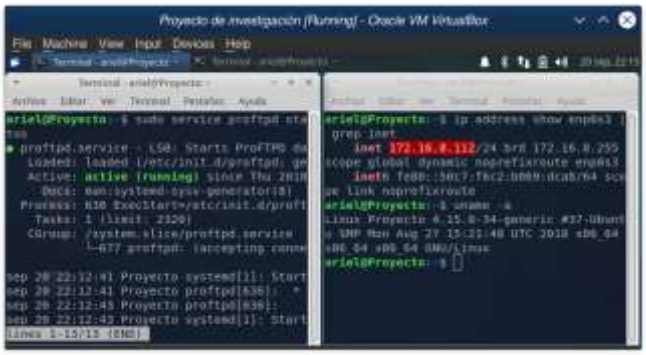

Figura 3 *— Servidor proftpd activo en Ubuntu 18.04 LTS.*

La librería se introdujo en la función *doInBackground*(C) de la clase *uploadTask*(B), ya que se hicieron pruebas de carga de archivos (Fragmento de código 7):

[2] https://github.com/gispunpauarg/Q2M
[3] https://github.com/linkindrew/easyFTP
[4] https://commons.apache.org/proper/commons-net/apidocs/org/apache/commons/net/ftp/FTPClient.html

```
[...]
public class demo extends ActionBarActivity {
	[...]
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		[...]
		Metricas.createInstance(demo.this); (A)
	}
	[...]
	class uploadTask extends AsyncTask<String, Void, String> { (B)
		[...]
		@Override
		protected String doInBackground(String... params) { (C)
			Metricas instancia = Metricas.getInstanceOf(); (D)
			try {
				[...]
				instancia.perceivedLatencyBegin(); (E)
				if (instancia.isPhoneConnected()) {
					instancia.getConnectionType();
					instancia.getJitter(params[0]);
					instancia.getLatency(params[0]);
					instancia.getPacketLoss(params[0], 10);
					instancia.getSignalStrength();
				} else {
					Log.e("Error", "No se puede realizar la medición de las
					métricas porque el dispositivo no está conectado a una
					red.");
				}
				ftp.connect(params[0],params[1],params[2]);
				[...]
				ftp.uploadFile(is,"test.png");
				return new String("Upload Successful");
			} catch (Exception e) {
				[...]
			} finally {
				instancia.perceivedLatencyStop(); (F)
			}
		}
		[...]
	}
}
```

Fragmento de código 7 *— Parte del código fuente de easyFTP donde se utiliza la librería.*

La librería fue implantada dentro de la actividad[5] principal de la aplicación; dentro de la clase *demo*. Se creó una instancia de la misma invocando al método estático *Metricas.createInstance*(A), al cual se le pasó por parámetros el contexto de la aplicación (*demo.this*), necesario para el funcionamiento de esta. Así se guardó en memoria la instancia estática de la clase *Metricas*, la cual es recuperada(D) dentro del método *doInBackground* la clase anónima *uploadTask* para, posteriormente, ejecutar una serie de funciones con el fin de recolectar métricas(E) (*batería cargando*, *tipo de conexión*, *jitter*, *latencia*, *pérdida de paquetes*, *intensidad de la señal* y *latencia percibida por el usuario*; Detalles sobre estas métricas pueden verse en la

[5] Una actividad o «*activity*» es una cosa única y enfocada que el usuario puede hacer. Casi todas las actividades interactúan con el usuario, por lo cual la clase *Activity* se toma el trabajo de crear una ventana con el fin de presentar la interfaz gráfica de usuario. (“Activity”, s.f.)

Tabla **1**). Finalmente, se ejecutó el método *perceivedLatencyStop* dentro del bloque *finally*(F) para asegurar que la métrica sea registrada independientemente de si el proceso de subida falla o no.
En la Figura 4 se muestra el resultado de la ejecución de la tarea de carga de un archivo al servidor FTP, En la columna izquierda se presentan los datos recolectados por Q2M, y en cada caso se expone el identificador de la métrica, la fecha y hora en la que se calculó y el valor obtenido.

Dirección IP del dispositivo cliente[6]: 172.16.8.90

Dirección IP del servidor FTP: 172.16.8.112

```
<metrica identificador="PhoneConnectedToANetwork" fecha="2018-
09-20 22:23:28">true</metrica>

<metrica identificador="ConnectionType" fecha="2018-09-20
22:23:28">Wi-Fi</metrica>

<metrica identificador="Jitter" fecha="2018-09-20
22:23:35">137.4</metrica>

<metrica identificador="Latency" fecha="2018-09-20
22:23:35">414.0</metrica>

<metrica identificador="PacketLoss%" fecha="2018-09-20
22:23:45">0</metrica>

<metrica identificador="SignalStrength" fecha="2018-09-20
22:23:45">-47</metrica>

<metrica identificador="UserPerceivedLatency" fecha="2018-09-20
22:23:51">23</metrica>
```

Figura 4 *— Contenido del archivo XML generado tras la ejecución de la función de subida (columna izquierda) y subida exitosa en easyFTP (columna derecha).*

Segundo escenario: Pruebamétricas

Para este escenario se desarrolló una aplicación simple (nombrada "Pruebamétricas") haciendo uso del *IDE* Android Studio, a la cual posteriormente se le implantó la librería. La motivación para crear esta aplicación y para llevar a cabo este escenario de prueba fue la necesidad de disponer de una forma eficaz y eficiente de medir todas las métricas que ofrece la librería, así como también proporcionar una interfaz de usuario simple y fácil de usar para efectuar diferentes pruebas con distintos usuarios, sin importar su nivel de conocimiento tecnológico e informático. En la Figura 5 se muestra la interfaz de la aplicación:

[6] El dispositivo cliente es el que está ejecutando easyFTP.

Figura 5 *— Interfaz de usuario de Pruebamétricas.*

Las pruebas se realizaron con un total de tres usuarios con teléfonos de distintas características. En la Tabla 3 se detalla el perfil básico de cada uno de ellos junto con información técnica sobre sus dispositivos, y en la Tabla 4 se muestran los resultados obtenidos de todas las pruebas que se llevaron a cabo.

Tabla 3 *— Información sobre los usuarios que participaron de las pruebas y sobre sus dispositivos.*

	Usuario #1	**Usuario #2**	**Usuario #3**
Edad	21	20	43
Ocupación	Estudiante universitario	Estudiante universitario	Informático
Tipo de usuario[7]	Usuario esporádico, con conocimientos	Usuario novato	Usuario frecuente, con conocimientos
Detalles del dispositivo			
Marca	XIAOMI	MOTOROLA	MOTOROLA
Modelo	Redmi Note 5A Prime	Moto E5 Plus	Google Nexus 6
Nivel de la API[8]	25	26	23
Versión del sistema operativo Android	7.1.2 (Nougat)	8.0.0 (Oreo)	6.0.0 (Marshmallow)

Tabla 4 *— Resultados de la ejecución de las dos pruebas.*

	Usuario #1		**Usuario #2**		**Usuario #3**	
	Dirección IP utilizada: 64.207.187.111					
	Prueba #1	**Prueba #2**	**Prueba #1**	**Prueba #2**	**Prueba #1**	**Prueba #2**
Brillo de la pantalla	19%	*N/A*	52%	29%	87%	40%

[7] ***Usuario novato:*** «*escaso conocimiento semántico*» (el conocimiento semántico es la comprensión que tiene el usuario de las funciones que se realizan, el significado de los datos de entrada y de salida, entre otras cosas).
Usuario esporádico, con conocimientos: «*conocimiento semántico razonable de la aplicación*».
Usuario frecuente, con conocimientos: «*buenos conocimientos semánticos*». (Pressman, 1998)

[8] A*pplication* P*rogramming* I*nterface*: Consiste en un conjunto de definiciones de subrutinas, protocolos de comunicación y herramientas para la construcción de software. ("Application programming interface", s.f.)

Intensidad de la señal	-41 dBm	-88 dBm	-45 dBm	-103 dBm	-53 dBm	-113 dBm
Latencia	217 ms	341 ms	408 ms	211 ms	399 ms	391 ms
Latencia percibida por el usuario	7 segundos	13 segundos	8 segundos	8 segundos	9 segundos	8 segundos
Luz ambiental	125 lux	16 lux	2 lux	650 lux	0 lux	0 lux
Porcentaje de carga de la batería	7%	30%	67%	66%	78%	33%
Proximidad	5 cm	5 cm	0 cm	5 cm	1 cm	1 cm
Batería cargando	false	true	false	true	false	true
Teléfono conectado a una red	true	true	true	true	true	true
Tipo de conexión	Wi-Fi	4G	Wi-Fi	4G	Wi-Fi	3G
Uso de memoria	1556 MB (54.71%)	1440 MB (50.63%)	1100 MB (59.08%)	955 MB (51.29%)	361 MB (35.88%)	347 MB (34.49%)
Uso del procesador por parte de la aplicación	*N/A*		36%	32%	*N/A*	
Jitter	9.3	18.0	43.3	45.7	137.3	12.0
Pérdida de paquetes	0%	20%	0%	0%	0%	0%

Las pruebas exhibidas en la Tabla 4 se hicieron utilizando la misma dirección IP para que los valores de las métricas de QoS (y en consecuencia, las métricas de QoE afectadas) fueran comparables.

De los resultados se observa que cuando los valores de las métricas de QoS (jitter, latencia o pérdida de paquetes) indicaron un desempeño subóptimo, las métricas de QoE se vieron afectadas negativamente. Un ejemplo de esta situación se ve en la latencia percibida por el usuario, que en este caso causó que el usuario tenga que esperar más tiempo para ver los resultados de la ejecución de su prueba.

La pérdida de paquetes (prueba #2 del usuario #1) es el aspecto que más afectó a la latencia percibida por el usuario cuando toma valores no óptimos; seguido por el jitter (prueba #1 del usuario #3); y luego la latencia, que causó efectos indeseables sobre dicha métrica de QoE pero fueron casi imperceptibles.

Otras observaciones que se pueden hacer sobre las pruebas ejecutadas son que, debido a una limitación actual del sistema operativo Android, la métrica de QoE "brillo de la pantalla" no se puede calcular si el ajuste "*adaptive brightness*" o *brillo adaptivo* (que cambia el brillo de la pantalla automáticamente en función de la luz del entorno) está activado en el dispositivo (prueba #2 del usuario #1); que en todos los casos las pruebas realizadas con una conexión de datos móviles resultó en una mejor intensidad de la señal (en dBm, menor es mejor); que se notó que en todos los dispositivos los valores de luz ambiental (en lux) y de proximidad (en cm) fueron imprecisos y que la métrica "uso del procesador por parte de la aplicación" no se pudo calcular en los dispositivos con una versión de Android anterior a 8.0.0 (Oreo) (prueba #1 y #2 del usuario #1 y #3).

6. CONCLUSIONES

En este trabajo se presentó la librería Q2M, la cual permite realizar mediciones de diversas métricas de QoS y QoE y exportar los valores resultantes para su análisis. Esta puede ser utilizada tanto en aplicaciones móviles como en *frameworks* de desarrollo de aplicaciones móviles. El conjunto de métricas que trata la librería puede ser fácilmente ampliado para cubrir más atributos y factores de QoS y QoE. La información que Q2M calcula y recolecta puede ser procesada por otras herramientas y/o analizada con métodos estadísticos.
El trabajo futuro está orientado a incorporar la funcionalidad necesaria para poder trasmitir la información recolectada a un servidor web que organice y centralice los desempeños de calidad de distintos dispositivos sobre una determinada aplicación móvil con la finalidad de realizar procesamientos más complejos.

REFERENCIAS

BLAIR, G. y STEFANI, J. (1997). *Open Distributed Processing and Multimedia*. Addison-Wesley.

BOREN, Z. (7 de Octubre de 2014). *There are officially more mobile devices than people in the world*. The Independent. Recuperado de https://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html

CHALMERS, D. y SLOMAN, M. (1999). *A survey of quality of service in mobile computing environments*. IEEE Communications surveys, 2(2), 2-10. https://doi.org/10.1109/COMST.1999.5340514

CHEN, Q.A., LUO, H., ROSEN, S., MAO, Z.M., IYER, K., HUI, J., SONTINENI, K., LAU, K. (2014). *QoE doctor: Diagnosing mobile app QoE with automated UI control and cross-layer analysis*. Trabajo presentado en Proceedings of the ACM SIGCOMM Internet Measurement Conference (pp. 151-164) de ACM, Vancouver, Canada. https://doi.org/10.1145/2663716.2663726

DEMARCO, T. (1986). *Controlling Software Projects: Management, Measurement and Estimates*. 1° ed. Englewood Cliffs, Estados Unidos: Prentice Hall.

GURWINDER, P. (2011). *QoS Modeling and Analysis using QML in Mobile Computing Environment* (Tesis doctoral, Jayoti Vidyapeeth Women's University). Recuperado de https://shodhgangotri.inflibnet.ac.in/bitstream/123456789/1218/1/gurwinder%20paul%20kaurcorrected.pdf

ICKIN, S. (2015). *Quality of Experience on Smartphones: Network, Application, and Energy Perspectives* (Tesis doctoral, Blekinge Institute of Technology). Recuperado de http://www.diva-portal.org/smash/get/diva2:833883/FULLTEXT02

KIM, H., LEE, D., LEE, J., LEE, K., LYU, W., CHOI, S.. (2008). *The QoE Evaluation Method through the QoS-QoE Correlation Model*. Trabajo presentado en Fourth International Conference on Networked Computing and Advanced Information Management (pp. 719-725) de IEEE, Gyeongju, Korea. https://doi.org/10.1109/NCM.2008.202

KUIPERS, F., KOOIJ, R., DE VLEESCHAUWER, D. y BRUNNSTRÖM, K. (Junio de 2010). *Techniques for Measuring Quality of Experience*. Trabajo presentado en International Conference on Wired/Wireless Internet Communications (pp. 216-227) de Springer, Heidelberg, Alemania. https://doi.org/10.1007/978-3-642-13315-2_18

PRESSMAN, R. (1998). *Ingeniería del Software: Un enfoque práctico*. 4° ed. Madrid, España: McGraw-Hill.

RAVINDRANATH, L., PADHYE, J., AGARWAL, S., MAHAJAN, R., OBERMILLER, I. y SHAYANDEH, S. (2012). *AppInsight: Mobile App Performance Monitoring in the Wild*. Trabajo presentado en OSDI (Vol. 12, pp. 107-120), Hollywood, Estados Unidos.

RAVINDRANATH, L., PADHYE, J., MAHAJAN, R. y BALAKRISHNAN, H. (2013). *Timecard: Controlling user-perceived delays in server-based mobile applications*. Trabajo presentado en Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (pp. 85-100), ACM, Nueva York, Estados Unidos. https://doi.org/10.1145/2517349.2522717

XU, Q., MEHROTRA, S., MAO, Z. y LI, J. (2013). *PROTEUS: network performance forecast for real-time, interactive mobile applications*. Trabajo presentado en Proceeding of the 11th annual international conference on Mobile systems, applications, and services (pp. 347-360), ACM, Taipei, China. https://doi.org/10.1145/2462456.2464453

Android Developers. *Activity*. Recuperado el 1 de Noviembre de 2018 de https://developer.android.com/reference/android/app/Activity

Application programming interface (s.f.). En *Wikipedia*. Recuperado el 4 de Noviembre de 2018 de https://en.wikipedia.org/wiki/Application_programming_interface