Universidad Zaragoza

Proyecto Fin de Carrera

Ingeniería en Informática

Recogida y visualización de datos para caracterización energética de dispositivos Android

Autor

Samuel Machín López

Director

Luis Manuel Ramos Martínez

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

2016

Agradecimientos

A todos los profesores que en mayor o menor medida han contribuido a que haya llegado hasta aquí y que me han dejado no sólo conocimientos técnicos.

A mis compañeros de carrera, por todas esas horas de clases y laboratorios, de largas noches acabando trabajos y también de celebraciones.

A Luisma, director de este proyecto, por apoyarme y aconsejarme cuando lo he necesitado y por todas las horas que ha invertido en mí.

A mi familia, por todo su cariño y su interés.

A mis amigos, por estar ahí en las buenas y en las malas.

A mis padres y mi hermana por el apoyo incondicional y por hacer de mí todo lo que soy.

A Julia, por estar siempre a mi lado, no perder nunca la confianza en mí y hacerme ser mejor persona.

RESUMEN

En este proyecto final de carrera se han investigado maneras de caracterizar el consumo energético de dispositivos móviles Android, centrándonos en el consumo por estados de uso de los usuarios en lugar de por componentes físicos.

Para ello, se ha desarrollado una aplicación para dispositivos Android que recopila una traza cada vez que suceden ciertos eventos en el sistema, recogiendo información sobre el estado del dispositivo en ese momento. Esa traza se almacena en el dispositivo en una base de datos para su posterior tratamiento.

Con la información recopilada, se realiza un modelo de regresión con el fin de obtener un modelo energético y estimar las previsiones de consumo. También se ha analizado los datos obtenidos siguiendo procedimientos y técnicas más sencillas con el fin de comparar métodos y resultados.

Previamente se ha hecho un estudio de aplicaciones disponibles en el mercado y trabajos de investigación relacionados, con el fin de tratar de mejorar lo que se ha hecho hasta ahora en este campo.

Como conclusión, se indican posibles continuaciones de este trabajo y vías futuras de investigación en este campo.

Índice

1. Introducción

En las siguientes subsecciones del presente apartado se detallará la razón de la realización de este Proyecto Fin de Carrera, así como varias nociones del entorno para facilitar la comprensión del plan desarrollado.

1.1. Motivación y objetivo general

En esta sección se detallan cuáles han sido los motivos para la realización de este proyecto final de carrera y los objetivos que se han intentado abarcar.

1.1.1. Relevancia de los *smartphones* y del sistema Android

Desde su irrupción en el mercado, allá por 2007, los teléfonos inteligentes o *smartphones* no han parado de crecer. Según un informe de la Fundación Telefónica del año 2014, España es el líder de la Unión Europea en penetración de *smartphones*, con un 81% de teléfonos inteligentes sobre el total de móviles, un fuerte crecimiento respecto al 63% que existía solamente dos años antes.

Dentro de las diferentes plataformas móviles existentes, el predominio de Android en España es claro. Como podemos ver en la Figura 1, a principios de 2015, pese a una ligera caída con respecto al año anterior, la cuota de mercado del sistema de Google era del 86,7%, muy por encima de sus dos más directos competidores: iOS, con un 10,4% y Windows Phone, con un 2,5%. Esa diferencia ha seguido aumentando durante los nueve primeros meses de 2015.

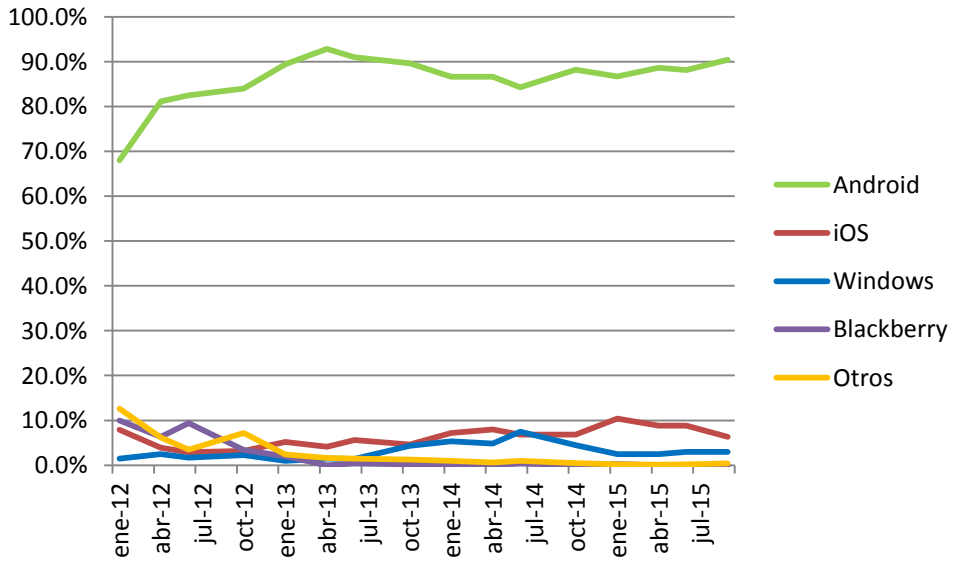

Figura 1: Evolución de la cuota de mercado en España según sistema operativo (Fuente: http://es.kantar.com)

1.1.2. Importancia de la batería en dispositivos móviles

Pese a la importante y creciente presencia de dispositivos móviles en nuestro día a día, todavía existe un quebradero de cabeza común para los usuarios de dichos dispositivos: la autonomía.

Estos teléfonos inteligentes de los que hablamos cada día lo son más. Sin embargo, el rápido aumento de sus capacidades de procesamiento, su conexión a internet o sus resoluciones de pantalla, así como las múltiples funciones que desempeñan (agenda, gps, juegos, alarma, reproductor multimedia, etc.), no ha ido en concordancia con un desarrollo de tecnologías de baterías que permita que el teléfono aguante alejado del enchufe más de un día como norma general.

Existen multitud de aplicaciones para monitorizar y gestionar el uso de la batería en Android. Sin embargo, no son siempre efectivas. Algunas de ellas provocan incluso más gasto de energía que ahorro. Otras veces el usuario no es consciente de cuál es la aplicación que provoca un consumo excesivo, en ocasiones debido a un "energy bug".

En este proyecto se pretende recoger información sobre los estados y eventos que se producen en el teléfono a través de una aplicación diseñada para ese fin, para así entender dónde y cómo se consume la energía de los dispositivos. La información obtenida será utilizada para rastrear el consumo energético del dispositivo, siendo posible, por ejemplo, averiguar el consumo de energía de cada aplicación utilizada o la eficacia de las aplicaciones de control energético.

1.2. Contexto del trabajo

El proyecto ha sido realizado dentro del grupo de Arquitectura de Computadores de la Universidad de Zaragoza (gaZ), un grupo de investigación formado por profesores del Departamento de Informática e Ingeniería de Sistemas (DIIS) de la Universidad de Zaragoza, integrados en el Instituto de Investigación en Ingeniería de Aragón (I3A).

El grupo de arquitectura de computadores de la universidad de Zaragoza (gaZ) está involucrado en la transferencia de tecnología e investigación aplicada. Los destinatarios de esta transferencia tecnológica son compañías y grupos de investigación que requieren optimización no convencional de software. Ejemplos de esto son sistemas empotrados de tiempo real, paralelismo de difícil extracción, o reducción del consumo energético del software. El grupo trabaja con nuevas plataformas como procesadores multicore, GPGPUS, ó FPGAs.

Su investigación aplicada se centra en la jerarquía de memoria y en los futuros procesadores multicore. Cubren varios segmentos de mercado, desde procesadores para tiempo real duro hasta sistemas masivamente paralelos o sistemas empotrados de bajo coste. Sus propuestas de diseño son testadas ejecutando aplicaciones reales en simulaciones del sistema completo.

También están interesados en el estudio del consumo energético de distintos sistemas, realizando modelos energéticos y proponiendo mecanismos que aumenten su eficiencia energética.

1.3. Objetivos específicos

El principal objetivo de este trabajo es el de recabar información sobre los estados y eventos que se producen en los dispositivos Android, que son los mayores causantes del gasto energético del mismo, así como visualizar y analizar dicha información. Todo con el propósito de conseguir una mayor comprensión del consumo de este tipo de dispositivos para poder actuar en consecuencia y ver el efecto de los eventos en el consumo energético.

La recogida de información se realiza mediante una aplicación para dispositivos Android que crea una traza cuando el usuario realiza determinados cambios o ajustes en su terminal, como activar/desactivar la red wifi, apagar/encender la pantalla, activar/desactivar el gps, abrir/cerrar aplicaciones, etc.

Los datos obtenidos se almacenan en una base de datos para su posterior manipulación y análisis de diferentes escenarios escogiendo distintas combinaciones de variables que se recogen en la traza antes mencionada.

Previamente se realizará un estudio de diversas aplicaciones disponibles en el mercado para dispositivos Android que gestionan y/o monitorizan el uso de la batería, para comprobar su utilidad real y sus características y así ayudarnos a definir nuestra aplicación de recogida de información. Con el mismo fin, resumiremos algunos trabajos de investigación relacionados con nuestro tema de estudio que pongan en perspectiva lo que se ha hecho al respecto en este campo, lo que podemos usar de ellos y lo que falta o es mejorable.

1.4. Planificación del PFC

En la Figura 2 se muestra un diagrama de Gantt con la planificación seguida durante la realización de este proyecto, detallando las diferentes tareas seguidas y el tiempo requerido para completarlas.

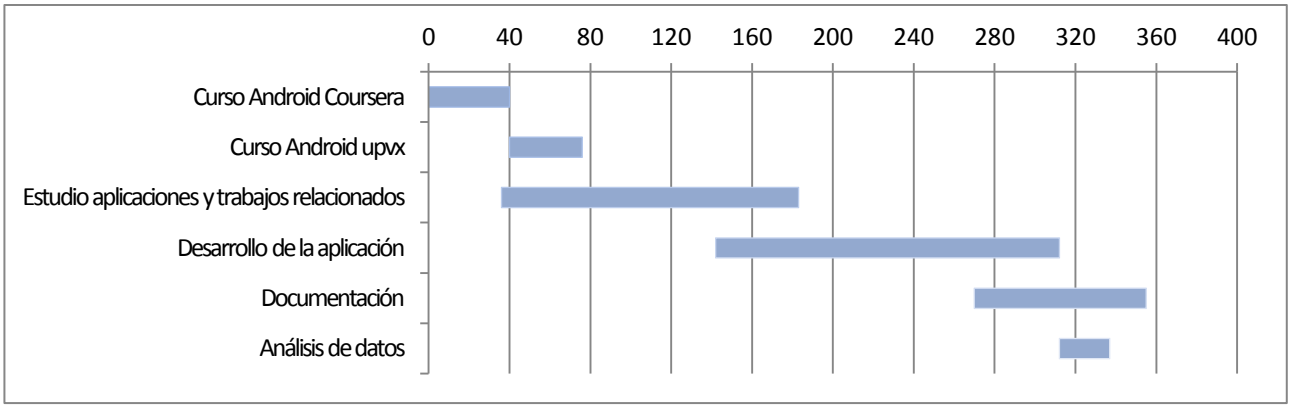

Figura 2: Diagrama de Gantt con la duración en horas de las fases del proyecto

En la Figura 3 se detalla el desglose del esfuerzo para cada tarea del proyecto.

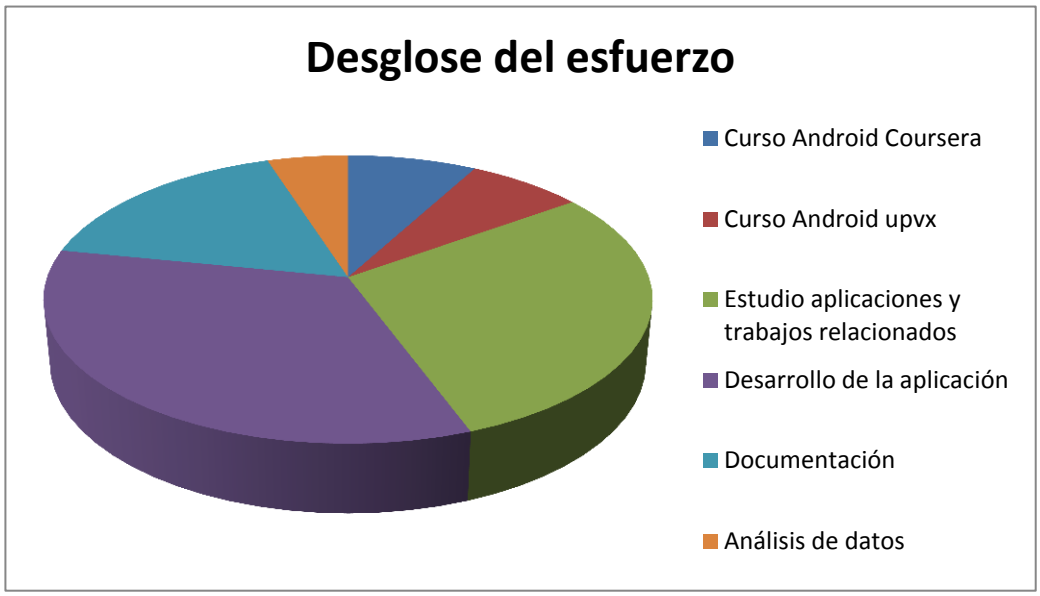

Figura 3: Desglose del esfuerzo por tareas el proyecto

Las horas empleadas para cada fase del proyecto han sido las siguientes:

Para la realización de los cursos de Android de Coursera y de la UPVX se han empleado 40 y 36 horas respectivamente.

Para el estudio de aplicaciones y trabajos relacionados (lo cual incluye reuniones con el director del proyecto) y búsqueda de información se han invertido 147 horas.

Para el desarrollo e implementación de la aplicación móvil se han utilizado 170 horas.

A la hora de escribir la documentación se han empleado 85 horas.

El análisis de datos ha costado realizarlo 25 horas.

Lo que suma un total de 503 horas.

2. Fundamentos

En los siguientes puntos se detallan las bases sobre las que se ha implementado este proyecto.

2.1. Tecnologías de baterías

Desde que en 1800 Alessandro Volta inventara la pila que actualmente lleva su nombre, han aparecido diversas tecnologías de fabricación de baterías basadas en diferentes materiales. Todas ellas buscan mejorar la tecnología precedente reduciendo tamaños, eliminando materiales contaminantes, aumentando su autonomía o incluso, en los últimos años, añadiendo características como la flexibilidad.

Las principales tecnologías de baterías hasta la fecha en orden cronológico son las basadas en plomo, las basadas en níquel y las basadas en litio.

En la actualidad, la gran mayoría de dispositivos móviles ('smartphones', tablets, e-books, etc) usan las baterías basadas en litio, concretamente las de iones de litio (Li-ion). Estas baterías utilizan un ánodo de grafito y un cátodo de óxido de cobalto, trifilina ($LiFePO_4$) u óxido de manganeso. Sus principales ventajas son una alta energía específica y densidad de capacidad, una buena eficiencia energética, una baja auto-descarga (menos de la mitad que las basadas en níquel) y la ausencia casi total de efecto memoria, es decir, pueden cargarse sin necesidad de estar descargadas completamente y sin que se reduzca su vida útil. Por otra parte, sus mayores limitaciones son que son más costosas que la mayoría de las baterías basadas en plomo o níquel, que no admiten bien los cambios de temperatura y que sufren mucho con las descargas completas.

Tipo	Energía/ peso	Tensión por elemento (V)	Duración (número de recargas)	Tiempo de carga	Auto-descarga por mes (% del total)
Plomo	30-40 Wh/kg	2 V	1000	8-16h	5 %
Ni-Fe	30-55 Wh/kg	1,2 V	+ de 10 000	4-8h	10 %
Ni-Cd	48-80 Wh/kg	1,25 V	500	10-14h *	30 %
Ni-Mh	60-120 Wh/kg	1,25 V	1000	2h-4h *	20 %
Li-ion	110-160 Wh/kg	3,7 V	4000	2h-4h	25 %
Li-Po	100-130 Wh/kg	3,7 V	5000	1h-1,5h	10 %

Figura 4: Tecnologías de baterías y sus características (Fuente: https://es.wikipedia.org/wiki/Batería_eléctrica)

Con vistas al futuro, los investigadores trabajan en mejorar la densidad de potencia, durabilidad, seguridad, tiempo de carga, flexibilidad, coste y demás características de este tipo de baterías. Por ejemplo, investigadores de IBM India han propuesto una fuente de alimentación experimental utilizando celdas de ion-litio de baterías de portátiles desechadas para usar en regiones sin electrificar de países en desarrollo.

2.2. Android

En este apartado se realiza una descripción del sistema operativo Android y se detalla brevemente las diferentes capas de su arquitectura.

2.2.1. Descripción general

Android es un sistema operativo móvil basado en el kernel de Linux. Fue diseñado inicialmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes (*smartphones*), tablets o tabléfonos (*phablets*), aunque en los últimos años se ha ido adaptando también para relojes inteligentes, televisores, automóviles, videoconsolas o cámaras digitales.

En sus inicios fue desarrollado por Android Inc., empresa que Google respaldó económicamente y que, finalmente, adquirió en 2005. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para progresar en los estándares abiertos de los dispositivos móviles. El primer teléfono móvil con el sistema operativo Android fue el HTC Dream y salió a la venta en octubre de 2008. Desde entonces, su penetración en el mercado no ha parado de crecer hasta el punto de tener más de mil millones de usuarios mensuales activos, tal y como reveló la compañía en 2014.

El código fuente de Android es publicado por Google bajo licencias de código abierto, aunque la mayoría de dispositivos montan una combinación de código abierto y código propietario.

2.2.2. Arquitectura Android

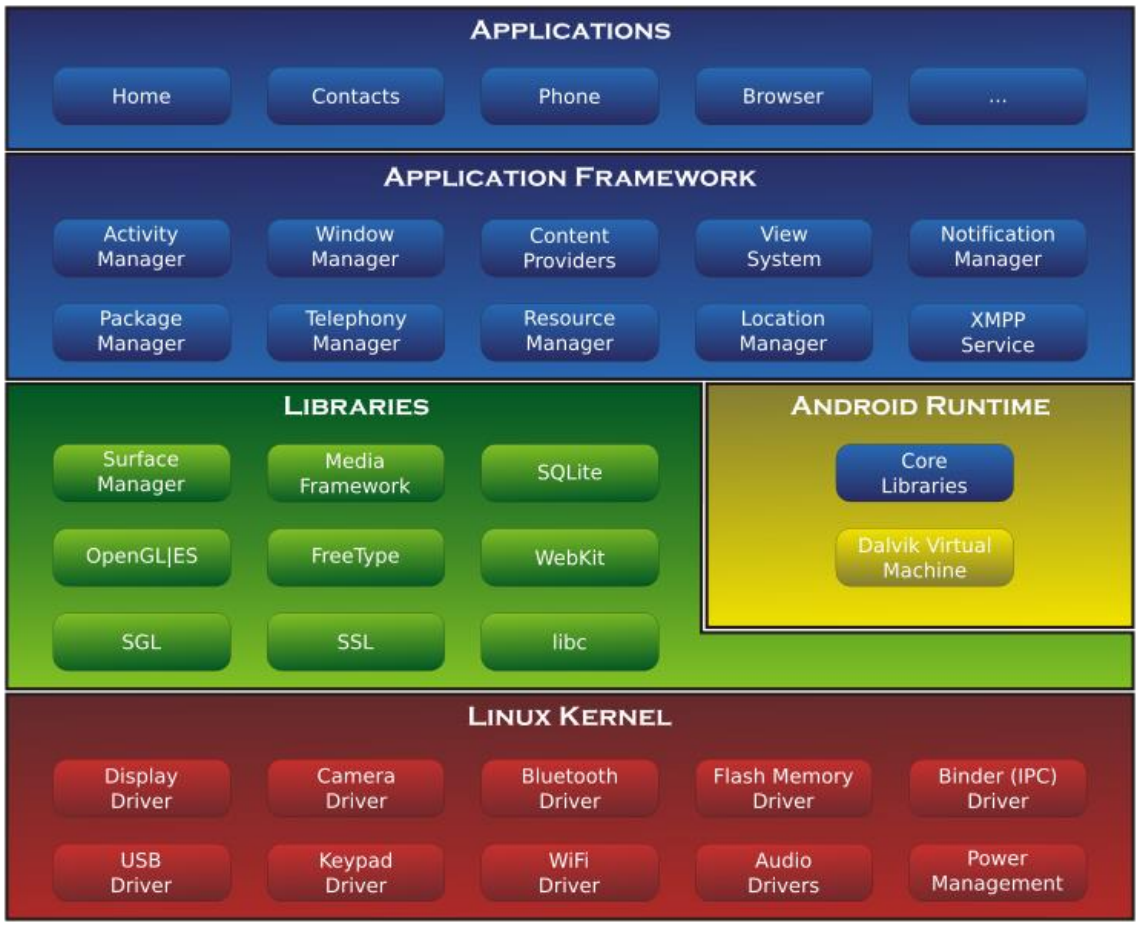

Figura 5: Diagrama de la arquitectura del sistema Android (Fuente: https://en.wikipedia.org/wiki/File:Android-System-Architecture.svg)

En la figura 2 podemos ver las diferentes capas de la arquitectura de Android, las cuales se van a detallar a continuación en orden ascendente, es decir, comenzando por las capas de más bajo nivel hacia niveles superiores.

Como ya se ha mencionado, el **kernel** de Android está basado en una de las ramas de soporte a largo plazo (Long-Term Support, LTS) del kernel de Linux, mayormente en sus versiones superiores a la 3.0.1. Este núcleo proporciona servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. Además actúa como capa de abstracción entre el hardware y el resto de la pila del software.

En una capa superior se encuentran las **bibliotecas** y el **Runtime de Android**.

El Runtime es un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik,

la cual ha sido escrita de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos con formato Dalvik Executable (.dex), que está optimizado para que necesite la menor cantidad posible de memoria. La máquina virtual está basada en registros y corre clases compiladas por el compilador de Java transformadas al formato .dex por la herramienta incluida "dx". Con la versión Android 4.4 se introdujo la posibilidad de elegir Android Runtime (ART) como nuevo entorno de runtime, el cual usa compilación previa (ahead-of-time, AOT), en lugar de compilación "justo a tiempo" (just-in-time, JIT) como su predecesora Dalvik, para compilar enteramente el bytecode de la aplicación a código máquina desde el momento en que se instala la aplicación, lo cual acelera su ejecución y reduce el consumo de batería a costa de aumentar ligeramente el tiempo de instalación y el espacio de almacenamiento en memoria. A partir de la versión 5.0 de Android se usa exclusivamente ART.

Android también incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo (Framework) de aplicaciones de Android. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

En el siguiente nivel de la arquitectura se encuentra el marco de trabajo de aplicaciones (**Application Framework**) que es el conjunto de APIs que los desarrolladores emplean para escribir aplicaciones y que utilizan las aplicaciones base de Android. La arquitectura está diseñada para simplificar la reutilización de componentes. Cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework).

Por último, en el nivel superior, se encuentran las **aplicaciones**. Las básicas del sistema operativo incluyen un cliente de correo electrónico, aplicación de SMS, calendario, mapas, navegador, contactos, etc. Todas las aplicaciones están escritas en lenguaje de programación Java.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%

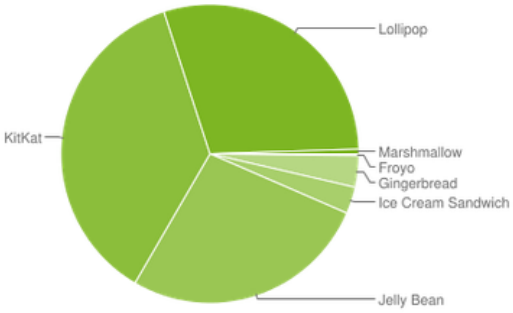

Figura 6: Distribución de las distintas versiones de Android a Diciembre de 2015 (Fuente: http://developer.android.com/intl/es/about/dashboards/index.html)

2.2.3. Conceptos relacionados con la energía

A continuación, se definen algunos términos que guardan relación con el ámbito del consumo energético de los dispositivos.

- **Wakelock:** Es un mecanismo por el cual el sistema permite a las aplicaciones mantener despierto el dispositivo si lo necesita para su ejecución en segundo plano. Existen dos tipos, parciales (mantiene la CPU activa) y totales (mantienen también la pantalla encendida). Si no se gestiona bien, puede provocar que el dispositivo se mantenga activo sin necesidad, incurriendo en un consumo excesivo de la batería.
- ***Energy bug*:** Fallo en la programación de una aplicación por el que sucede un excesivo consumo de la batería. Normalmente es debido al mal manejo de los wakelocks.
- **Gobernador de CPU:** Funcionalidad que los dispositivos Android heredan del kernel de Linux y que se encarga de dictar la frecuencia a la que funciona la CPU del dispositivo según el uso que se haga de él. Hay de muchos tipos, desde los más conservadores (menos rendimiento, más duración de batería), hasta los más exhaustivos (mayor rendimiento, menos autonomía) pasando por otros más adaptativos (modulando de distintas maneras la frecuencia de la CPU dependiendo del uso que se le dé). Algunos de los más significativos son:
    - **ONDEMAND:** Este gobernador suele ser el que viene por defecto en la mayoría de casos. Se caracteriza por **pasar a la máxima frecuencia**

posible cuando usemos el móvil y luego si no hace falta tanto, ir bajando paso a paso hacia frecuencias más bajas. Con este gobernador se consigue una gran fluidez del sistema pero no ahorra mucha batería.

- **CONSERVATIVE:** Siempre pone la **mínima velocidad posible hasta que el procesador se congestiona**, entonces sube de velocidad paso a paso. Conservative proporciona una experiencia menos receptiva que ondemand, pero puede ahorrar batería.
- **PERFORMANCE:** Un gobernador que parte de la idea de que cuanto mayor sea la frecuencia, antes acabará la CPU lo que esté haciendo y podrá volver a *Deep Sleep*. **Siempre pone la frecuencia del procesador a la máxima posible**.
- **INTERACTIVEX:** Muy similar a *OnDemand* pero optimizado. Se basa en otro gobernador llamado *Interactive* que escala también a la máxima velocidad pero que, al utilizar un *timer* interno en vez de la cola de procesos, **permite utilizar más las frecuencias intermedias**. Además, cuando la pantalla está apagada, pone el procesador a la mínima velocidad posible.

2.3. Análisis de regresión

En estadística, el análisis de regresión es un proceso estadístico para la estimación de relaciones entre variables. Incluye muchas técnicas para el modelado y análisis de diversas variables, cuando la atención se centra en la relación entre una variable dependiente y una o más variables independientes. Más específicamente, el análisis de regresión ayuda a entender cómo el valor típico de la variable dependiente cambia cuando cualquiera de las variables independientes es variada, mientras que se mantienen las otras variables independientes fijas. Más comúnmente, el análisis de regresión estima la esperanza condicional de la variable dependiente dadas las variables independientes - es decir, el valor promedio de la variable dependiente cuando se fijan las variables independientes. El objetivo es la estimación de una función de las variables independientes llamada la **función de regresión**.

Se han desarrollado muchas técnicas para llevar a cabo análisis de regresión. Métodos comunes tales como regresión lineal y regresión ordinaria de mínimos cuadrados son paramétricos, en los que la función de regresión se define en términos de un número finito de parámetros desconocidos que se estiman a partir de los datos. Regresión no paramétrica se refiere a las técnicas que permiten a la función de regresión basarse en un conjunto específico de funciones, que pueden ser de dimensión infinita.

Para este proyecto se va a utilizar la **regresión lineal múltiple a través del origen**, que se expresa mediante la siguiente fórmula:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

, donde y_i es la variable dependiente, β_i son los parámetros respectivos a cada variable, x_{ij} es la *i*-ésima observación en la *j*-ésima variable independiente y *p* es el número de parámetros independientes a tener en cuenta en la regresión.

Se dice que es múltiple, ya que tiene más de una variable independiente, y es a través del origen, porque se prescinde del valor residual o intercepto, ya que se van a analizar escenarios con variables excluyentes, es decir, si no se está en un escenario se está en otro (por ejemplo dispositivo con pantalla encendida o apagada; si no se está en ninguno de los dos el dispositivo estará apagado y no consumirá energía).

A la hora de interpretar los resultados, habrá que observar ciertos indicadores que confirmen la validez del modelo de regresión y su aproximación a la realidad.

- El coeficiente $\boldsymbol{R^2}$ y $\boldsymbol{R^2}$ **ajustado** son medidas de la bondad del ajuste, es decir, cuánto se ajustan los datos observados al modelo de regresión propuesto. R^2 tomará valores entre 0 y 1, siendo un valor cercano a 1 un indicador de un buen ajuste y cercano a 0 de un mal ajuste. Además, el R^2 ajustado relaciona el R^2 con el número de variables. Debido a que cada vez que introducimos una nueva variable independiente en el modelo, R^2 no puede hacer otra cosa que aumentar, R^2 ajustada puede interpretarse como una corrección de honestidad. Nos castigará disminuyendo cuando introduzcamos variables innecesarias.
- El **Error Típico** o desviación estándar, devuelve el error típico del valor de *Y* previsto para cada *X* de la regresión. Será interesante obtener el mínimo posible.
- El **nivel de confianza** establece un intervalo de confianza, es decir, el rango de valores posibles entre los cuales se puede encontrar los valores verdaderos de los coeficientes estimados por la regresión. El nivel de confianza típico es el del 95%.
- **Estadístico t** o pruebas de significancia de los coeficientes del modelo de regresión. Sirve para rechazar la hipótesis de que cada uno de los coeficientes sean nulos. Observando su probabilidad asociada (valor p) podremos ver cuánto es de probable que el valor obtenido para t ocurra por azar. Se suele aceptar una probabilidad menor del 5% ($p < 0,05$).
- **Prueba F** o pruebas de significancia conjunta del grupo de variables. Similar a la anterior pero con respecto al conjunto, no a cada variable por separado. El valor crítico de F asociado, nos indica la probabilidad de que ese valor de F ocurra por azar. Como en el caso anterior, se suele aceptar una probabilidad menor del 5%.

3. Trabajo relacionado

La gran proliferación de dispositivos móviles en general y del sistema Android en particular, ha provocado un aumento exponencial con el paso de los años de aplicaciones de todo tipo. Como una de las mayores preocupaciones de los usuarios de estos dispositivos es la duración de la batería, existen en el mercado una gran variedad de aplicaciones relacionadas con la gestión de la batería y el consumo de energía.

En este apartado se van a analizar los tipos de aplicaciones de esta clase que existen en la actualidad, así como las características de algunas de las más populares. También se van a recoger algunos trabajos de investigación realizados hasta la fecha que guarden relación con el tema que aborda este proyecto.

3.1. Aplicaciones de gestión de energía

Dentro de las aplicaciones de gestión de energía se puede hacer una distinción en dos grandes grupos: las aplicaciones pasivas, que simplemente recopilan información, y las aplicaciones activas, que permiten al usuario modificar determinados parámetros o configuraciones del sistema. También se mencionará las opciones de las que dispone el propio sistema Android por defecto en ambas categorías.

3.1.1. Aplicaciones informativas

Dentro de esta categoría se encuentran las aplicaciones que recaban y muestran información como el estado actual de la batería, los consumos de las distintas aplicaciones instaladas y de sistema, gráficas de dichos consumos y/o de diversos parámetros de la batería (nivel de carga, temperatura, voltaje, etc.), predicciones de tiempos de descarga de la batería para distintos usos o la opción de guardar archivos externos de históricos o logs. Algunas también cuentan con la opción de añadir widgets en el escritorio o en el área de notificaciones con dicha información, pudiendo llegar a ser esto su principal o única característica. A continuación se presentan algunas aplicaciones de esta clase y sus características.

- **Battery snap**
    - Gráfico dinámico con la carga de la batería.
    - Información actual del estado de la batería.
    - Datos históricos.
    - Gráficos: 1 día, 2 días y 1 semana.
    - Acceso directo a la información de batería del sistema.
    - Exporta a la raíz de la tarjeta sd un archivo .txt con el historial.

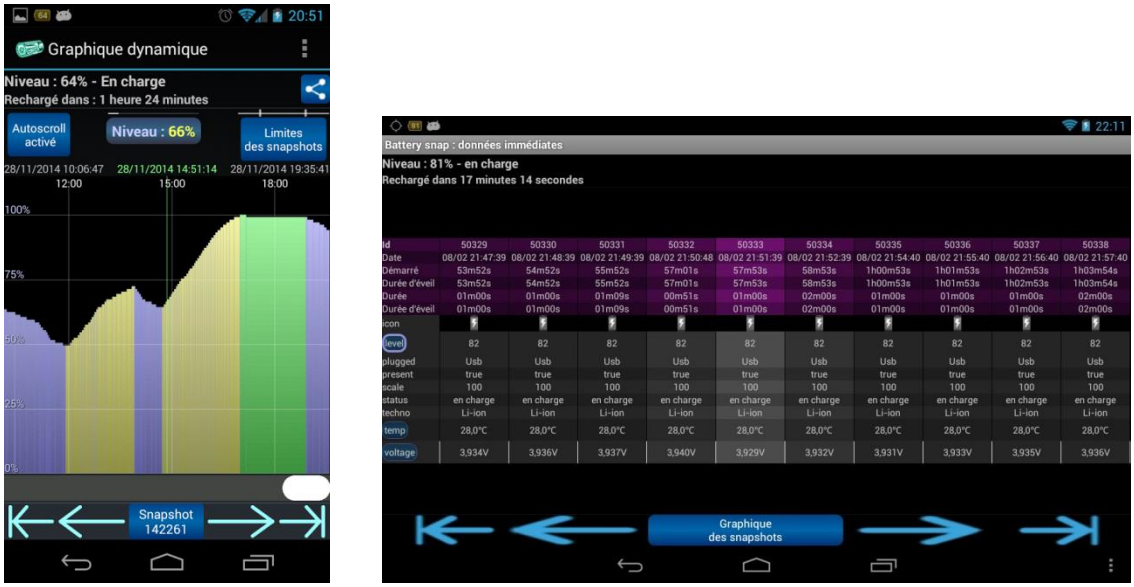

Figura 7: Capturas de pantalla de la aplicación Battery Snap. Gráfica de carga/descarga (izqda.) e histórico de datos recogidos (drcha.)

- **Battery monitor widget**
  - Información del estado de la batería.
  - Gráficos históricos.
  - Alarmas basadas en datos de la batería.
  - Reseteo de las estadísticas de la batería (root).
  - Estimación del tiempo de carga/descarga.
  - Cálculo de la capacidad real de la batería.

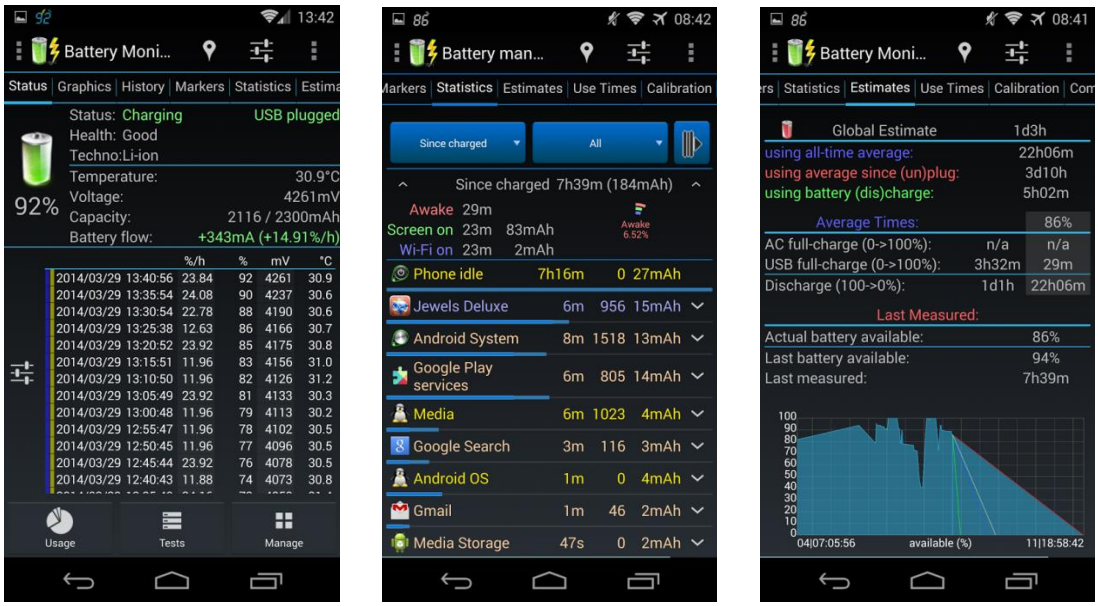

Figura 8: Capturas de pantalla de la aplicación Battery Monitor Widget. Pantalla de estado (izqda.), estadísticas (centro), estimaciones (drcha.)

- **Información de la batería (sistema Android)**

El propio sistema Android ofrece la siguiente información acerca de la batería:

  - Gráfica de descarga.
  - Información de tiempo y porcentaje de carga consumido por las aplicaciones y servicios.
  - Estimación del tiempo de carga restante.

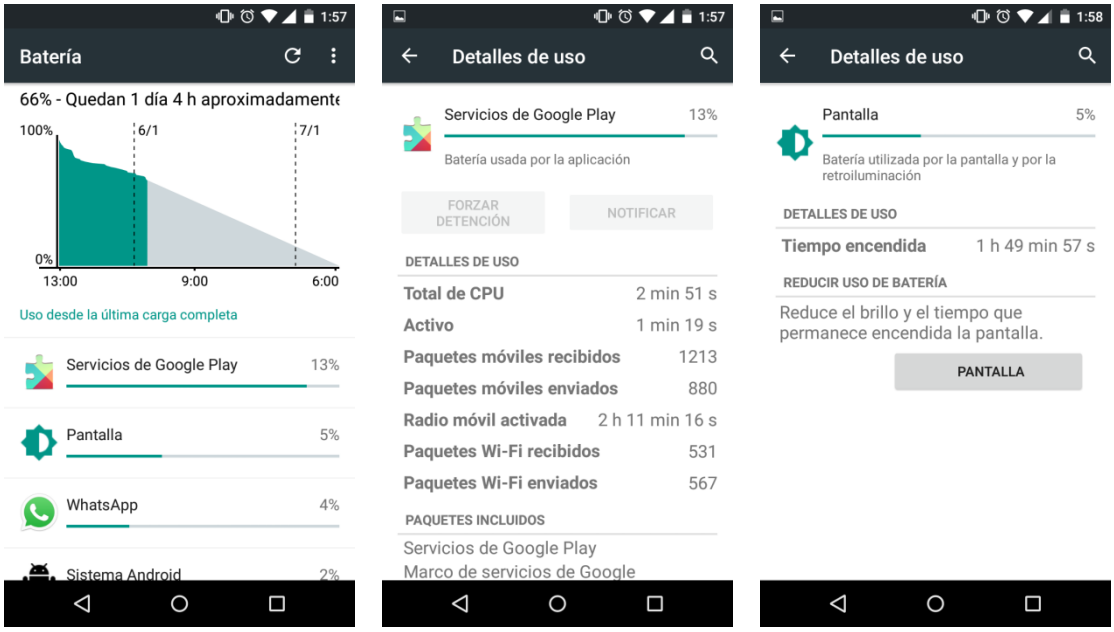

Figura 9: Captura de la información de la batería que proporciona el sistema Android (v5.1). Gráfica y estimación de descarga y uso de aplicaciones (izqda.), detalles de uso de un servicio (centro) y detalles de uso de la pantalla (drcha.)

3.1.2. Aplicaciones activas

Pertenecen a este tipo de aplicaciones las que permiten, además de visualizar información y estados de la batería o el sistema, intervenir al usuario de distintas maneras para intentar optimizar el consumo de energía del dispositivo, como hibernar determinadas aplicaciones en el caso en que veamos que están realizando un consumo excesivo o un comportamiento indeseado, asociar acciones a estados (por ejemplo, apagar el wifi si se detecta que la señal es débil, apagar los datos en un determinado horario o maximizar la duración de la batería apagando lo que no se considere imprescindible si el nivel de carga baja de un determinado nivel, entre otros). Ejemplos de estas aplicaciones son:

- **Greenify**

- Información de aplicaciones instaladas (ejecutándose en segundo plano, funcionamiento programado, usadas recientemente, etc.)
- Posibilidad de ponerlas a hibernar para que no actúen si el usuario considera que están afectando al consumo de energía o al rendimiento general del dispositivo.

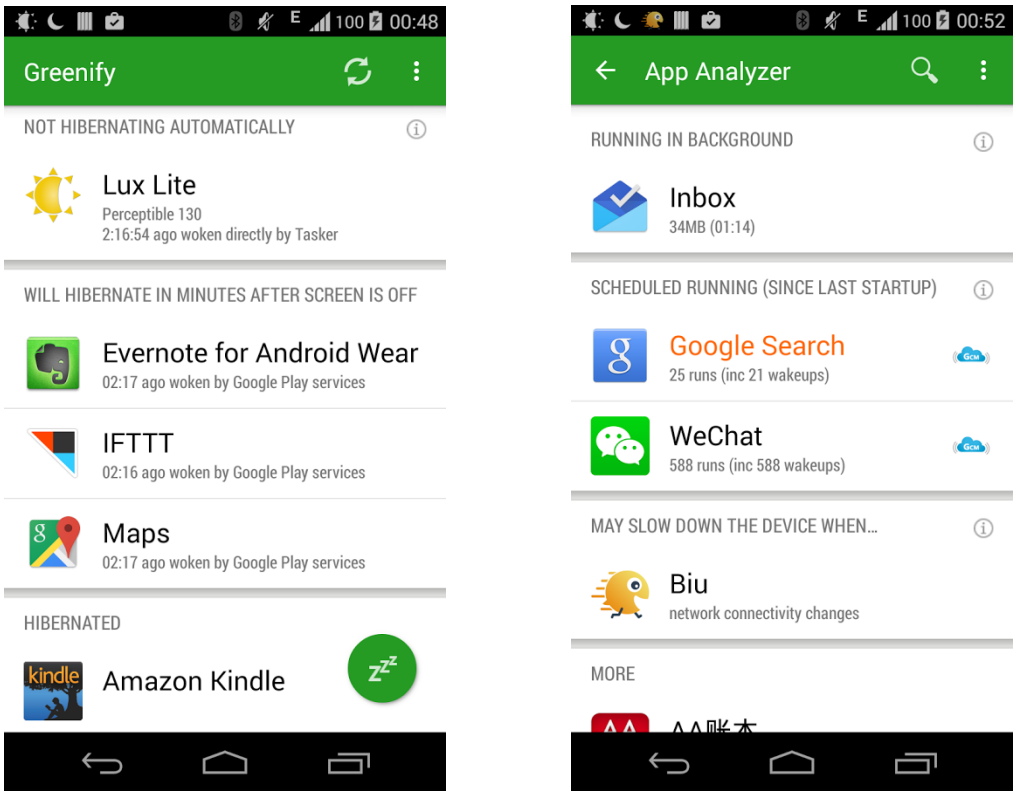

Figura 10: Capturas de pantalla de la aplicación Greenify. Pantalla de inicio (izqda.) y analizador de aplicaciones (drcha.)

- **CPU Tuner** (root)
  - Información actual (porcentaje de batería, temperatura, estado de la carga, ajustar frecuencia máxima y mínima del procesador, posibilidad de activar/desactivar datos, wifi, bluetooth, etc.)
  - Estadísticas (tiempo en las frecuencias disponibles en la cpu y transiciones totales entre dichas frecuencias)
  - Disparador (realiza acciones dependiendo del nivel de la batería)

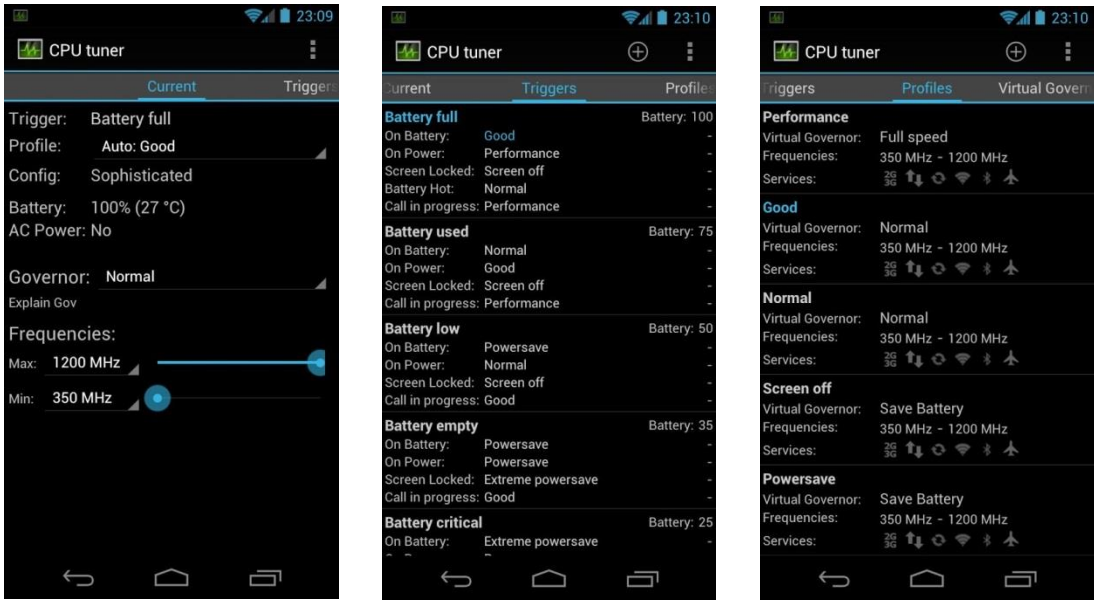

Figura 11: Capturas de pantalla de la aplicación CPU Tuner. Modo actual (izqda.), disparadores (centro) y perfiles (drcha.)

Cabe mencionar también el esfuerzo de Android en sus últimas versiones por mejorar de manera efectiva la autonomía de la batería. A continuación se mencionan dos "aplicaciones" del sistema introducidas en las versiones 5 y 6 de Android respectivamente.

- **Modo ahorro de energía** (sistema Android, versión 5.X)
    - Reducción del rendimiento en aras de la autonomía.
    - Limitación de la vibración.
    - Limitación de la mayor parte de la transmisión de datos en segundo plano.

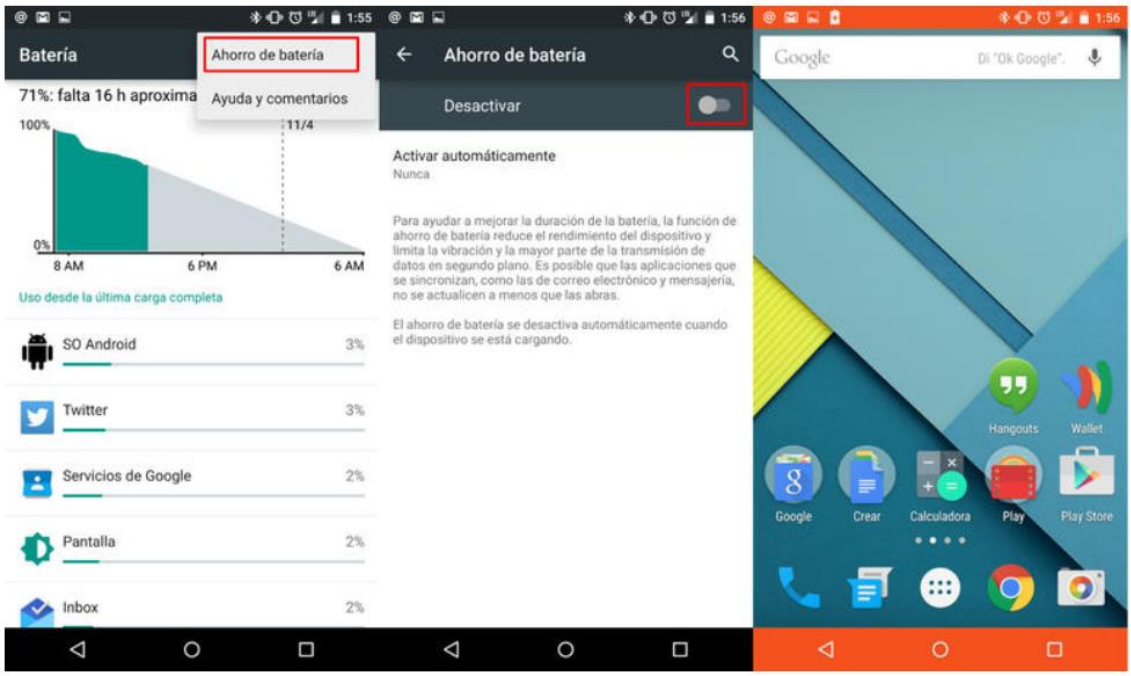

Figura 12: Captura de la activación del modo ahorro de batería. La barra superior e inferior se vuelven naranjas.

- **DOZE** (Sistema Android, versión 6)

Una vez activado, el dispositivo detecta si el móvil ha permanecido en una superficie sin utilizar durante más de 30 minutos y activa las siguientes características:

  - La red se desactiva, a menos que la aplicación tenga alta prioridad y Google Cloud requiera actualización de datos.
  - Los “wake locks” son ignorados y sin prioridad alguna.
  - Las alarmas programadas se desactivan, a excepción de las más importantes y configuradas así por la propia aplicación y posteriormente el usuario.
  - La red WiFi se mantiene parcialmente inactiva, pues la búsqueda de estas no se encuentra en funcionamiento.
  - La sincronización de las aplicaciones no recibe prioridad, estas se desactivan y no obtienen autorización alguna.

3.2. Trabajos de investigación previos

Antes de comenzar a realizar este proyecto, se ha realizado una búsqueda de trabajos publicados relacionados. A continuación se resumen aquellos que se han encontrado más interesantes para el desarrollo del actual proyecto.

Lide Zhang *et al* [1] describen una técnica de construcción automatizada de un modelo energético, usando los sensores de voltaje construidos en los propios dispositivos y el conocimiento del comportamiento de descarga de la batería para monitorizar el consumo de energía mientras se controla la gestión de la energía y los estados de los componentes a nivel individual. Para ello, primero se utilizan medidas directas y un software de "entrenamiento" en el que se mide el impacto de cada componente de forma individual manteniendo el resto constantes. Por ejemplo, se deja la pantalla, el gps, el wifi, los datos y el audio en su mínimo consumo mientras se hace pasar a la cpu por sus distintas frecuencias, o se dejan todos al mínimo y se hace pasar a la pantalla por distintos intervalos de brillo, todo ello mientras se mide el voltaje de la batería. Con estos datos se usan técnicas de regresión para sacar coeficientes de cada componente, minimizar el error y construir un modelo energético basado en la medición directa.

Para compararlo con el modelo energético automático y universal que quieren obtener, diseñan una aplicación que construya ese modelo energético en tres pasos: obtener la curva de descarga de la batería para cada componente de manera individual, determinar el consumo de potencia para cada estado de los componentes y aplicar técnicas de regresión para deducir el modelo energético. Una vez obtenido dicho modelo lo comparan con el medido para comparar resultados.

En otro trabajo, Alex Shye *et al* [2], de la Universidad de Northwestern (EE.UU.), utilizan una idea parecida, la de estimar el consumo por componentes para obtener mediante regresión un modelo energético. En este caso, recogen logs del uso de varios dispositivos por parte de diferentes personas, a los que aplican su modelo energético con el fin de conocer qué componentes son los que más energía consumen cuando el dispositivo está activo, llegando a la conclusión de que son la pantalla y la cpu. Para minimizar dicho consumo energético reducen el brillo de la pantalla y la frecuencia de la cpu paulatinamente y comparan esta técnica con una reducción del brillo y la frecuencia más brusca preguntando a los usuarios, con el objetivo de intentar reducir el consumo al máximo y que afecte mínimamente al uso del dispositivo.

Siguiendo esta temática, Jemin Lee *et al* [3] del IEEE, presentan un método de generación automática de un modelo energético para *smartphones*. Para ello, sus autores recogen datos de patrones de uso cada segundo mediante una aplicación creada para tal fin que recopila esos datos y los junta en lo que denominan "pedazos" cada vez que el nivel de batería baja un 1%. Después de filtrar esos pedazos para evitar datos inválidos se extrae el modelo energético de cada componente hardware mediante regresión. En este caso, los componentes analizados y utilizados son la CPU, la pantalla y el WIFI.

La segunda herramienta que desarrollan para crear el modelo estimado se comporta como un programa de entrenamiento, generando patrones de uso mediante el control de los componentes hardware utilizados. Para cada porcentaje de batería (que generará un “pedazo”), mantienen constante el uso de la CPU, el brillo de la pantalla a un nivel determinado y mandan paquetes TCP de un tamaño fijo (1 MB) al servidor local con una tasa de paquete determinada. Finalmente, para evaluar los modelos energéticos generados, comparan las estimaciones generadas mediante sus herramientas con los datos medidos con los sensores de corriente internos de los que disponen los teléfonos que han empleado, utilizando seis aplicaciones diferentes para realizar las pruebas: un juego, un reproductor de video, un servicio de VOD (Video On Demand) streaming, un navegador web, un programa de mensajería y un SNS (Social Network Service).

Los tres trabajos mencionados tienen en común la búsqueda de un modelo energético a partir de medidas de uso del teléfono con el fin de estimar el consumo del dispositivo ([1] y [3]) o realizar algunos ajustes en los componentes medidos para tratar de minimizar dicho consumo ([2]). De estos trabajos de investigación hay varios aspectos que resultan interesantes para este proyecto, como la toma de datos de uso mediante una aplicación móvil, los estados de los componentes hardware medidos y, sobre todo, el uso de técnicas de regresión para crear el modelo energético. Sin embargo, hay diferentes elementos de estos trabajos que resultan insuficientes (como tratar únicamente la CPU y la pantalla, o aplicar los modelos a aplicaciones concretas) o están enfocados de manera distinta a lo que se busca con este proyecto, de manera que buscan más el consumo por componente hardware bajo unas circunstancias dadas (“entrenan” sus programas manteniendo niveles constantes de CPU, brillo, etc.) que por escenarios de uso del usuario, lo cual pretende este proyecto.

4. Metodología

En este apartado se va a profundizar acerca de la metodología empleada en este proyecto. Al no ser este un proyecto software al uso sino más bien algo más cercano a la investigación, no dispone de requisitos funcionales y no funcionales que un cliente o empresa requiere de un trabajo. En lugar de eso, existen una serie de aspectos que se consideraron necesarios para conseguir el objetivo deseado en un principio y que han podido sufrir modificaciones por el camino.

En los siguientes sub-apartados se va a explicar qué información se ha querido recopilar, almacenar, tratar y analizar, y qué métodos y herramientas se han utilizado para ello.

4.1. Recolección de información

Para analizar el consumo del dispositivo y la influencia de los distintos componentes del sistema y los escenarios de uso, es necesario recopilar información del dispositivo en tiempo real. A tal efecto, se vio claro desde un principio que era necesaria una aplicación que recogiera dicha información y la almacenara para su posterior análisis. Para ello fue necesario realizar dos cursos de introducción a la programación en Android, ya que en la carrera no había ninguna asignatura que proporcionara estos conocimientos. Los cursos se realizaron online en las plataformas Coursera y la UPVX (Cursos MOOC de la Universidad politécnica de Valencia).

Esta aplicación, llamada InfoBateria (código completo disponible en el Anexo 7.3.2), ejecuta un servicio en segundo plano, guardando una traza con los datos que monitoriza cuando alguno de los componentes *hardware* o *software* deseados cambie su estado (a eso lo llamaremos *trigger* o disparador). La información a recoger y sus estados es la siguiente (entre paréntesis los valores que toman los estados en la traza):

- **Fecha/Hora:** En cada traza recogida aparecerá la fecha y hora del sistema. Además, si el usuario así lo decide, se generará una traza cada cierto tiempo, pudiendo elegir entre 15, 30 o 60 minutos de intervalo.
- **Trigger:** En cada traza aparecerá también recogido el evento que ha producido el que se haya disparado dicha traza como un string (SC para *screen*, WI para el Wifi, FA para la *Foreground App*, etc). Este campo facilitará el tratamiento de la información posteriormente y su filtrado por eventos, así como la contabilización del tiempo en cada estado.
- **Wifi:** Son tres estados los que dispararán la traza.
    - Deshabilitado (0)
    - Habilitado (1)
    - Conectado a SSID xxxx (2_xxxx)

- **Data:**
    - Datos móviles deshabilitados (0)
    - Datos móviles habilitados (1)
- **Sync:** Estado de la sincronización automática. No genera traza porque Android no permite que avise cuando cambia su estado.
    - Sincronización deshabilitada (0)
    - Sincronización habilitada (1)
- **Network:** Tipo de red.
    - Modo avión (0)
    - Sin cobertura (1)
    - 2g (2)
    - 3g (3)
- **GPS:**
    - Deshabilitado (0)
    - Habilitado (1)
    - En uso (2)
- **NLP (Network Location Provider):** Tipo de localización utilizada según si se usa gps o redes. Introducida en las últimas versiones de Android.
    - Deshabilitado (0)
    - Solo dispositivo (1) – Determina la ubicación solo mediante gps
    - Ahorro de batería (2) – Determina la ubicación con Wifi y redes móviles.
    - Máxima precisión (3) – Determina la ubicación mediante gps, Wifi y redes móviles.
- **Screen:**
    - Pantalla apagada (0)
    - Pantalla encendida (1)
- **Bluetooth:**
    - Bluetooth apagado (0)
    - Bluetooth encendido (1)
- **Nivel de batería (%):** Recoge el nivel actual de la batería en tanto por ciento. Puede generar traza con cada cambio del 1% en el nivel de la batería si el usuario lo desea.
- **Voltaje:** Voltaje actual de la batería en mV. Su cambio no genera traza.
- **Temperatura:** Temperatura actual de la batería en grados centígrados. Su cambio no genera traza.
- **Plugged:** Indica si el dispositivo está cargando o no y si es mediante corriente alterna o por usb.
    - Desenchufado (0)
    - Enchufado AC (1)
    - Enchufado USB (2)

- **Call:** Indica si se hace o se recibe una llamada
    - No llamada (0)
    - Llamada (1)
- **Foreground App:** Dispara una traza cuando cambia la aplicación en primer plano que se está ejecutando y recoge un *string* con su nombre.
- **Installed:** Dispara una traza cuando se instala una aplicación y recoge un *string* con su nombre.
- **Uninstalled:** Dispara una traza cuando se desinstala una aplicación y recoge un *string* con su nombre.

Además de ésta información ya mencionada, se pensó en recoger datos de algún componente más que se acabó rechazando, bien por la imposibilidad de Android de advertir de cambios en su estado o bien porque se consideró irrelevante o de poca importancia. Por ejemplo, los cambios en el brillo de la pantalla, los *wakelocks* adquiridos o los servicios en segundo plano no pueden detectarse por el sistema Android. También se pensó en recoger información sobre las celdas de red, el *ring* del teléfono cuando recibe una llamada o el nivel de volumen del dispositivo, pero se descartaron por el camino por su escasa o nula influencia en el resultado final.

Para recopilar estas trazas, se hace uso de lo que en la API de Android se llama *BroadcastReceiver*, que son componentes destinados a recibir y responder ante eventos del sistema (SMS recibido, llamada entrante, pantalla apagada, etc.) o de otras aplicaciones. Los *Receivers* registrados para un evento serán notificados cada vez que éstos ocurran. En este caso, todos los *BroadcastReceiver* están asociados a eventos del sistema y son los siguientes

- **Intent.ACTION_BATTERY_CHANGED**: Contiene información acerca del estado de la batería. Esta aplicación usa la información de nivel de la batería (porcentaje de carga/descarga), temperatura, voltaje y si el dispositivo esta enchufado a alguna fuente de alimentación y a cuál (a la corriente o por USB).
- **Intent.ACTION_SCREEN_OFF y ACTION_SCREEN_ON:** Se envían cuando la pantalla se apaga o se enciende respectivamente.
- **Intent.ACTION_POWER_CONNECTED y ACTION_POWER_DISCONNECTED:** Se envían cuando una fuente de alimentación se conecta o se desconecta respectivamente.
- **ConnectivityManager.CONNECTIVITY_ACTION**: Se envía cuando se produce un cambio en la conectividad de la red. La aplicación lo usa para detectar si se enciende o se apaga la red de datos y para detectar si el dispositivo está en modo avión o sin cobertura.
- **WifiManager.WIFI_STATE_CHANGED_ACTION:** Indica que el Wifi ha sido habilitado, deshabilitado, está habilitándose o deshabilitándose.

- **WifiManager.NETWORK_STATE_CHANGED_ACTION:** Indica que el estado de la conectividad del Wifi ha cambiado. La aplicación utiliza tanto este *Broadcast* como el anterior para saber cuándo el Wifi se habilita, se deshabilita o se conecta a una red (y a cuál).
- **LocationManager.PROVIDERS_CHANGED_ACTION:** Se envía cuando se produce un cambio en los proveedores de localización. Indica si el gps está deshabilitado, habilitado sin usar o en uso.
- **LocationManager.MODE_CHANGED_ACTION:** Indica si cambia el modo de localización entre deshabilitado, solo dispositivo (gps), ahorro de batería (redes) o máxima precisión (redes y gps). Función añadida en la versión 19 de la API.
- **TelephonyManager.ACTION_PHONE_STATE_CHANGED:** Se envía si se produce un cambio en el estado de la llamada. Con esto se recoge cuando se produce una llamada y cuando finaliza.
- **BluetoothAdapter.ACTION_STATE_CHANGED:** Indica si se ha producido algún cambio en el adaptador *bluetooth*.
- **Intent.ACTION_PACKAGE_ADDED e Intent.ACTION_PACKAGE_REMOVED:** Indica si se ha instalado o desinstalado una aplicación en el dispositivo respectivamente. Contiene información del nombre del paquete. Si una aplicación sufre una actualización la aplicación recibirá un primer *Broadcast* de que la aplicación se ha desinstalado y seguidamente, cuando se completa la actualización, otro *Broadcast* avisando que la aplicación se ha instalado.

Otra información que era interesante recoger en la traza era la aplicación que estaba ejecutándose en primer plano en cada momento, pero Android no tiene *BroadcastReceivers* que recojan esto. Para conseguirlo se emplean los servicios de accesibilidad que proporciona la API de Android. Estas funcionalidades están pensadas para proporcionar mejoras o añadidos en las interfaces de usuario para personas con discapacidades (invidentes, sordos, daltónicos, etc.) o personas que, temporalmente, van a tener limitaciones a la hora de interactuar con el dispositivo (conduciendo, en un ambiente muy ruidoso, etc.).

Mediante estos servicios podemos usar un *listener* de eventos de accesibilidad y reaccionar cuando se reciba un **AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED** que nos indica que se ha abierto una ventana de menú, una ventana de tipo *popup* o un diálogo. Una vez recibido, se comprueba si el evento corresponde con una actividad (*activity*), que es como se conoce en Android a las pantallas de las aplicaciones, y si esa actividad es de una aplicación diferente a la que estaba ya en primer plano, se almacena una traza.

La única deficiencia de este método con respecto a los *Receivers* es que no se puede activar el servicio de accesibilidad para una aplicación mediante código, debe ser el

usuario el que la active. Como solución a este pequeño inconveniente, la aplicación dispone de un botón que indica si el servicio está activado o no y, si se pulsa, lleva al usuario directamente al menú de ajustes de accesibilidad donde se puede cambiar dicho estado.

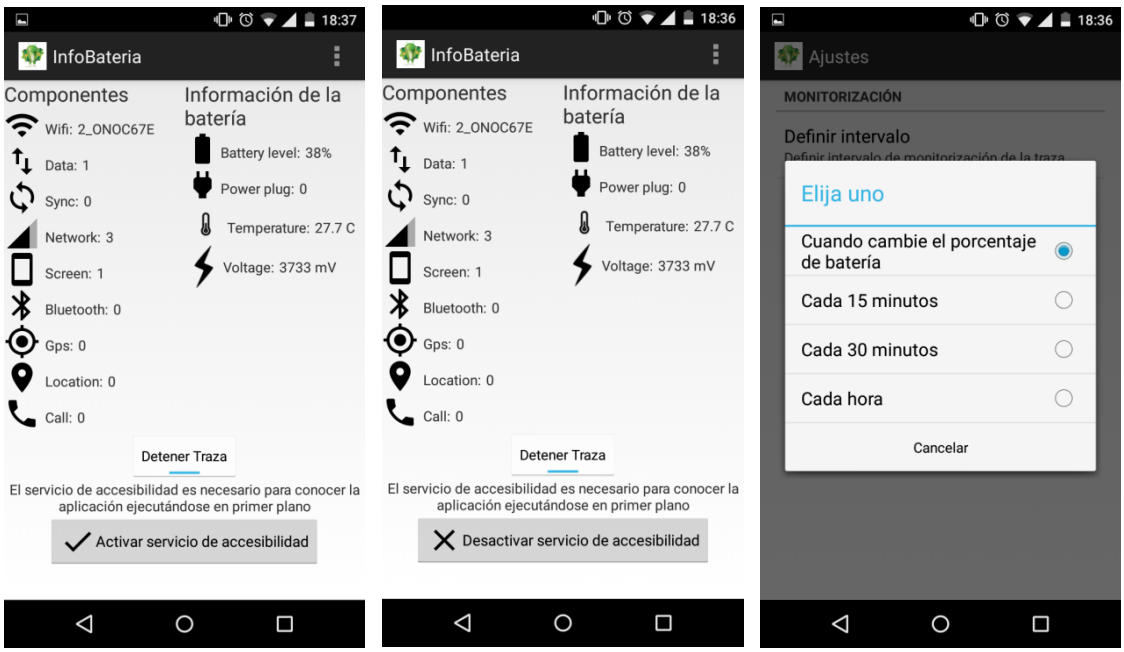

Figura 13: Capturas de la aplicación InfoBateria. Servicio de accesibilidad desactivado (izqda.), activado (centro) y pantalla para seleccionar el intervalo de monitorización automática de la traza (drcha.)

Para que esto se ejecute en segundo plano mientras el usuario realiza un uso normal del dispositivo, se utiliza un Servicio, que es el mecanismo que tiene Android para ejecutar acciones en *background.* Este servicio se inicia cuando el usuario pulsa el botón 'Comenzar Traza' en la pantalla principal de la aplicación, y desde él se llama a la clase que contiene los *Receivers*, que se mantendrán escuchando aunque el usuario salga de la aplicación hasta que se vuelva a pulsar el botón (donde ahora pondrá 'Detener Traza').

La gestión de memoria de Android puede empezar a terminar procesos si detecta que la memoria disponible en el dispositivo es críticamente baja, para así liberarla. Esto podría provocar que el sistema matara el proceso de ejecución de nuestra aplicación, lo que pararía el servicio y dejaría de almacenar trazas. Para evitar esta situación se utiliza la etiqueta *START_STICKY* como respuesta a la función de la clase Service 'onStartCommand'. Esto le dirá al sistema que, cuando la memoria del dispositivo vuelva a ser estable y el sistema esté listo, relance el servicio.

4.2. Almacenamiento y tratamiento de la información recopilada

El siguiente paso a considerar era cómo almacenar y tratar la información recogida por la aplicación InfoBatería, de manera que fuera sencillo de acceder y manipular para su posterior análisis.

En un principio se pensó en guardar una línea de texto por cada traza, con unos códigos de 2 o 3 letras para cada componente vigilado. Este sistema ocupaba poco espacio de almacenamiento, pero contaba con el inconveniente de que era necesario "traducir" esa información de formato texto a valores de variables con algún otro programa. Así que finalmente se optó por almacenar los valores en una base de datos.

Android utiliza SQLite como sistema gestor de bases de datos relacional. SQLite está escrito en C, es de código libre y, a diferencia de los sistemas de gestión de base de datos cliente-servidor, su motor no es un proceso independiente con el que el programa se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo, lo cual reduce la latencia en el acceso a la base de datos.

El almacenamiento en base de datos ocupa más que en texto plano, pero con la capacidad actual de los dispositivos móviles sigue siendo un tamaño insignificante. Por ejemplo, las trazas de una semana con el programa corriendo ininterrumpidamente ocupan 350 KB. Además, la base de datos se almacena en la ruta del almacenamiento externo (si el dispositivo dispone de él, sino en la partición que tenga emulada para tal fin) con lo que no molesta a la memoria principal del sistema.

La ventaja que aporta la base de datos con respecto al archivo de texto, es una mayor capacidad de tratar la información realizando consultas sobre ellas a la hora de componer los diferentes escenarios de uso. También la visualización de los datos es más clara que con un texto con códigos para los distintos componentes y se puede guardar fácilmente como archivo de Excel, que también es una herramienta muy potente para tratar la información posteriormente. De hecho, el siguiente paso una vez almacenados y tratados los datos, y elegidos los distintos escenarios de uso, es sacar un modelo energético mediante operaciones matemáticas complejas, para lo cual Excel resulta muy útil.

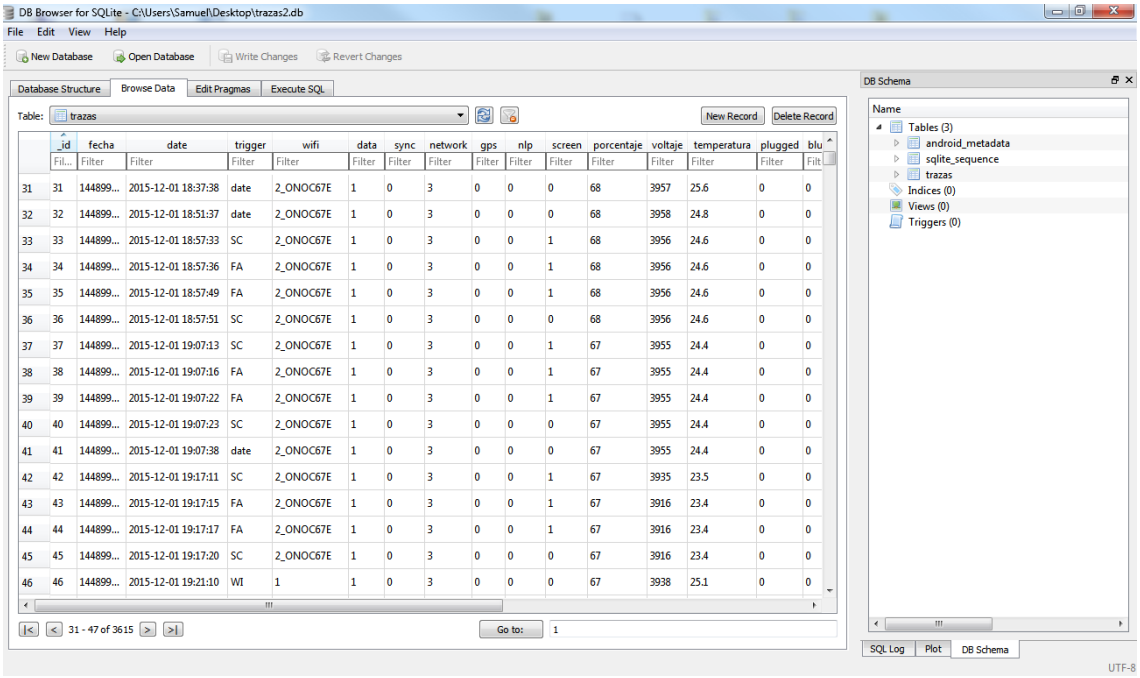

Figura 14: Ejemplo de la base de datos almacenada con las diferentes trazas, visualizada con el programa *DB Browser for SQLite* para Windows.

4.3. Análisis de los datos

La última fase de la metodología seguida en este proyecto consiste en analizar los datos obtenidos y tratados con anterioridad.

Partiendo del estudio de las aplicaciones disponibles en el mercado y de los trabajos previos relacionados, se ha optado por realizar dos enfoques diferentes a la hora de analizar los datos de las trazas. El primero, basado en trabajos relacionados ([1], [2] y [3]), consiste en realizar análisis matemáticos de regresión con distintos escenarios planteados para sacar el modelo energético. El segundo, inspirado en los datos que proporciona la aplicación *Battery Monitor Widget*, tratará de analizar los mismos escenarios que el caso anterior pero sin emplear métodos de regresión.

Antes de realizar cualquiera de los dos enfoques mencionados, es necesario realizar una estimación de la capacidad real de la batería y una medición de los tiempos.

4.3.1. Cálculo de tiempos

Para realizar los diferentes cálculos relacionados con la duración de los estados elegidos, se ha optado por hacer unas funciones sencillas en visual basic para Excel, opción que viene incluida en el propio programa de ofimática. Se han realizado tres funciones: **screenOn**, **screenOff** y **timeApp.** (Código completo en Anexo 7.3.1)

- **screenOn** calcula el tiempo transcurrido con la pantalla encendida durante una descarga de la batería del dispositivo. Para ello basta con filtrar la columna *Trigger* de las trazas de una descarga para que aparezcan sólo las trazas con *trigger* SC (screen). Después se selecciona de esas trazas filtradas la columna 'Date' que indica el tiempo en milisegundos transcurrido desde el 1 de Enero de 1970. Por último se introducen como parámetros de entrada de la función las columnas de tiempos en milisegundos desde el inicio de la descarga que se acaba de crear y screen (0 o 1) y la función devuelve el tiempo transcurrido con la variable *screen* a 1.
- **screenOff** calcula el tiempo transcurrido con la pantalla apagada durante una descarga de la batería del dispositivo. El funcionamiento es igual que para la función anterior pero contando el tiempo con la variable *screen* a 0 en lugar de a 1.
- **timeApp** calcula el tiempo durante el cual una aplicación determinada ha estado en primer plano, con la pantalla encendida, para una descarga de la batería. La metodología para preparar los datos es similar a las anteriores, pero ahora se filtrarán las trazas por los *triggers* SC (screen) y FA (Foreground App). De manera que se seleccionarán las fechas (en milisegundos transcurrido desde el 1 de Enero de 1970) de aquellas trazas en las que aparezca en la columna *Foreground App* la aplicación deseada y la traza sucesiva (que será o bien un cambio de aplicación en primer plano o que se apagó la pantalla), de esta forma podremos calcular la diferencia de tiempo entre ambas, que será una parte del tiempo que esa aplicación estuvo en primer plano. Finalmente se pasará como parámetro de entrada de la función esa columna de tiempos.

4.3.2. Medición de la capacidad real de la batería

La capacidad de una batería es la cantidad de energía que puede almacenar durante la carga y devolver durante la descarga. En el caso de los dispositivos móviles se suele medir en miliAmperios-hora (mAh), que representa la cantidad de energía que, en una hora es capaz de cargar la batería con una corriente continua de 1 mA.

Este dato es crucial para realizar los análisis de las trazas que se pretende (en los apartados siguientes se detalla en qué se emplea la capacidad para cada enfoque de análisis), así que será lo primero que se debe realizar.

Con el paso del tiempo y de los usos, las baterías van perdiendo su capacidad inicial, esto es, la que da el fabricante en la hoja de características. Para ser más precisos en los análisis se va a obtener la capacidad real de la batería utilizando los datos de las cargas del dispositivo.

Para estimar la capacidad de la batería después de una carga, necesitamos obtener de las trazas la duración de la carga en horas, el amperaje del cargador en mA y el porcentaje de batería inicial y final. Con estos datos, multiplicamos el amperaje del

cargador (mA) por la duración (h) y tendremos los mAh de la carga. Ahora basta con extrapolar esos mAh en un determinado porcentaje de batería (porcentaje final menos porcentaje inicial) al 100% de la batería.

$$Capacidad_{carga}(mAh) = Tiempo\ de\ Carga(h) \times\ Amperaje\ del\ cargador\ (mA)$$

$$Capacidad_{Total}(mAh) = \frac{Capacidad_{carga}(mAh) \times\ 100}{(NivelBatería_{final} - NivelBatería_{inicial})}$$

Debido a que cuando la batería está a un nivel muy bajo o a un nivel muy alto de carga el comportamiento de las cargas y las descargas es diferente (para proteger la batería y alargar su vida útil), para estimar la capacidad se tomarán los datos de carga cuando la batería está entre el 15% y el 90% de su carga.

4.3.3. Análisis mediante modelos de regresión

El primer enfoque a la hora de analizar la información recabada con la aplicación InfoBatería, es el de aplicar técnicas de regresión lineal múltiple a diferentes escenarios que iremos definiendo. Esta técnica es la que utilizan la mayoría de trabajos previos relacionados con modelos energéticos. Lo que les diferencia de este proyecto es que en todos los casos anteriores basan sus modelos de regresión en medidas de potencia, normalmente conseguidas mediante instrumentación externa (vatímetro).

Aquí se pretende simplificar este método todo lo posible y hacerlo así accesible a más gente que no puede/quiere/sabe utilizar instrumentos de medida o, incluso, *rootear* su dispositivo (obtener permisos de administrador para acceder a todos sus recursos). Es por eso que se van a utilizar tiempos en los estados (en horas) y la capacidad de la batería (mAh) para sacar el consumo de las variables que se considere (en mA). La fórmula queda de la siguiente manera:

$$Creal_i(mAh) = t_{i1}(h) \times I_{i1}(mA) + t_{i2}(h) \times I_{i2}(mA) + \cdots + t_{in}(h) \times I_{in}(mA) + \varepsilon_i$$

, donde:

- $Creal_i$ es la capacidad real de la batería en el porcentaje de descarga de la traza i. Se obtiene a partir de la estimación previa de la capacidad total de la batería dividida por el porcentaje medido. Por ejemplo, si la capacidad total de la batería es de 2000 mAh y la traza empieza en el 95% de la carga y termina en el 25% (en ese momento ponemos el dispositivo a cargar) habría que dividir los 2000 mAh por 0,70 ((95-25)/100).
- t_{ij} es el tiempo, en horas, en el escenario j para la traza i.
- I_{ij}es el coeficiente de corriente consumida, en mA, por el escenario j en la traza i. Es lo que se trata de estimar mediante la regresión.

- n es el número de escenarios definidos.
- ε_i es el error aleatorio de la traza i.

Se van a definir algunos escenarios con los que trabajar, con la limitación de que el número de trazas debe ser mayor al número de escenarios para que la regresión sea correcta.

El modelo más básico sería contemplar un único escenario, con lo que para calcular el único coeficiente de consumo bastaría con dividir los mAh consumidos en la descarga entre el tiempo total. Este caso sería muy variable, ya que no tiene en cuenta si la pantalla está encendida o apagada, o las aplicaciones utilizadas o los componentes del móvil empleados (gps, wifi, bluetooth, etc).

El siguiente modelo sería contemplar dos escenarios, pantalla encendida y pantalla apagada. Aquí ya se podría sacar algún resultado más interesante, como el consumo del móvil en reposo (pantalla apagada), aunque el coeficiente de pantalla encendida seguirá siendo variable dependiendo del uso que le demos al dispositivo.

A partir de aquí se seguirán definiendo escenarios más complejos como, por ejemplo, un modelo con cuatro escenarios que serían: pantalla apagada, pantalla encendida con la aplicación más utilizada en primer plano, pantalla encendida con la segunda aplicación más utilizada en primer plano y pantalla encendida con el resto de aplicaciones.

4.3.4. Análisis sin utilizar modelos de regresión

Como enfoque alternativo al del análisis mediante métodos de regresión, se va a proponer uno basado en medidas y cálculos simples. La idea se ha obtenido a partir de las observaciones hechas a una de las más completas aplicaciones del mercado relacionadas con el consumo de batería: *Battery Monitor Widget*.

En dicha aplicación, mencionada en el apartado de estudio de aplicaciones, hay una pantalla de estimaciones en las que aparecen medias y predicciones acerca de tiempos de carga (AC o USB) y descarga. De estos últimos, resultan especialmente interesantes las estimaciones sobre el tiempo de descarga de la batería completa con la pantalla encendida, con la pantalla apagada y la media según todos los datos de descargas recopilados por la aplicación, así como la estimación de la capacidad de la batería.

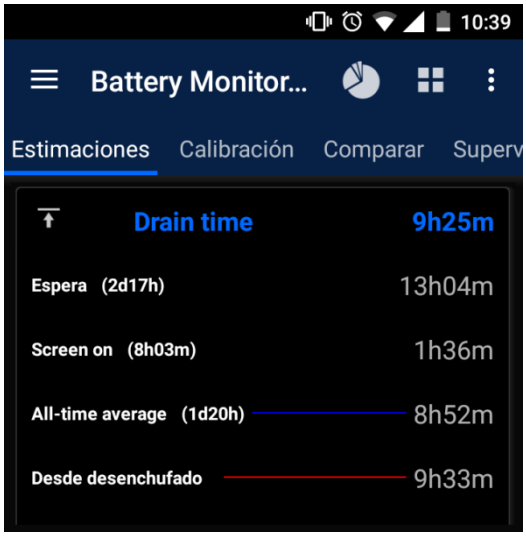

Figura 15: Captura de la aplicación *Battery Monitor Widget*. En ella se puede ver las distintas estimaciones que realiza.

Viendo que, esta aplicación da unas estimaciones de manera sencilla, se pensó en obtener los datos de consumo de pantalla encendida y apagada mediante alguna técnica más simple. De manera que, si se consigue, definiendo los estados de pantalla encendida y apagada como escenarios, se pueden sacar resultados también para otros escenarios diferentes y así poder compararlos con el enfoque anterior (regresiones). La solución adoptada se describe a continuación.

Partiendo de dos únicos escenarios, pantalla encendida y pantalla apagada, se van a contabilizar el número de ocasiones en los que ha disminuido el porcentaje de batería en un 1% en cada uno de los dos escenarios. Para ello, se elegirá en la aplicación InfoBatería la opción de intervalo de monitorización cuando cambie el porcentaje de batería (es la opción por defecto). Con esta opción obtendremos una traza con toda la información recopilada cada vez que el porcentaje de batería disminuya en una unidad. Para calcular los mA que consume cada escenario bastará con multiplicar el número de veces que ha disminuido el porcentaje de batería para cada escenario por los mAh que corresponden al 1% de la batería (mAh totales estimados previamente dividido para 100) y ese resultado dividirlo entre el tiempo que ha ocupado dicho escenario en la descarga de la batería. $Consumo_{ij} = \frac{(Cap_{total}/100) \times número\ de\ \nabla\%_{ij}}{t_{ij}}$

, donde:

- $Consumo_{ij}$ es el consumo del escenario *i* en la traza *j*.
- Cap_{total} es la capacidad total de la batería estimada previamente.
- $número\ de\ \nabla\%_{ij}$ es el número de cambios en el porcentaje de la batería que incluyen al escenario *i* en la traza *j*.
- t_{ij} es el tiempo empleado por el escenario *i* en la traza *j*.

Con las trazas disponibles se podrá ir realizando la media la media de estos consumos. Cuantas más trazas haya disponibles, más precisa será esta estimación.

Una vez obtenidos resultados de ambos enfoques, se compararán con el fin de conocer cuál de los dos es más aproximado a la realidad e incluso pueden llegar a ser compatibles entre sí. En un principio se supone que el análisis mediante regresiones será más ajustado y completo, pero si el otro análisis no varía mucho quizás merecería la pena quedarse con ese, ya que es más sencillo.

5. Resultados

En este apartado se detallan los resultados de los análisis de la información recopilada por la aplicación InfoBateria.

Van a distinguirse principalmente los dos enfoques propuestos con anterioridad: el análisis mediante modelos de regresión y sin ellos. Dentro de cada enfoque se propondrán varios escenarios de uso posibles, para los que se obtendrá un modelo energético.

Pueden verse los resultados completos de los análisis en el Anexo 7.1

5.1 Análisis mediante regresión

Para el análisis mediante regresión lineal múltiple se han realizado pruebas con 2, 4 y 5 escenarios para 15 conjuntos de trazas de descarga de la batería. Los escenarios se han definido, respectivamente, como:

- 2 escenarios: pantalla encendida y pantalla apagada.
- 4 escenarios: pantalla apagada, pantalla encendida usando la aplicación Whatsapp, pantalla encendida utilizando la aplicación Twitter y pantalla encendida con el resto de usos.
- 5 escenarios: los mismos que para el caso anterior añadiendo el escenario pantalla encendida con la aplicación Chrome.

En los tres casos vemos que el modelo de regresión se ajusta bien a la realidad, ya que el coeficiente R^2 ajustado se acerca bastante a 1 (0'92, 0'90 y 0'89 respectivamente). Los indicadores de probabilidad del estadístico t y de F también son bastante buenos, ya que dan valores bastante por debajo de 0'05, que es el valor que se suele estimar como límite de aceptación. Todos excepto el valor para el escenario pantalla encendida con el resto de aplicaciones del caso de estudio de 5 escenarios. Eso puede ser debido a que al aumentar el número de escenarios estudiados (variables dependientes) se necesitan más muestras para que el resultado sea adecuado. Al emplear 15 muestras para todos los casos de estudio, a medida que aumentamos el número de escenarios puede aparecer algún valor considerado como dato atípico.

Estadísticas de la regresión	
Coef. de correl. múlt.	0.996804833
Coef. R^2	0.993619875
R^2 ajustado	0.916206019
Error típico	131.1356972
Observaciones	15

Estadísticas de la regresión	
Coef. de correl. múlt.	0.998154674
Coef. R^2	0.996312752
R^2 ajustado	0.904398048
Error típico	108.3759235
Observaciones	15

Estadísticas de la regresión	
Coef. de correl. múlt.	0.99824744
Coef. R^2	0.99649795
R^2 ajustado	0.89509713
Error típico	110.77436
Observaciones	15

Tabla 1: Resultados de los coeficientes de correlación para las regresiones con 2, 4 y 5 escenarios respectivamente

También el error típico de los coeficientes es algo elevado (en torno al 20-25% de desviación sobre la recta de regresión). Esto es entendible, puesto que siguen siendo escenarios muy variables. No es lo mismo estar con una aplicación con wifi o con datos, o estar con el navegador abierto leyendo un texto que reproduciendo video o audio online, por poner unos ejemplos. Esto se podría corregir añadiendo escenarios más precisos, pero sería necesario un número mayor de descargas de las que obtener trazas. Los coeficientes de consumos quedan de la siguiente manera:

- 2 escenarios

	Coeficientes	*Error típico*
Screen on (t)	215.122793	40.9508109
Screen off (t)	19.9610151	3.86956572

Tabla 2: Coeficientes y errores típicos para el caso con 2 escenarios.

- 4 escenarios

	Coeficientes	*Error típico*
Screen off (t)	15.7753196	3.6161655
Screen on whatsapp (t)	463.758782	132.667274
Screen on twitter (t)	193.596542	48.361977
Screen on resto (t)	275.135235	65.7371862

Tabla 3: Coeficientes y errores típicos para el caso con 4 escenarios.

- 5 escenarios

	Coeficientes	*Error típico*
Screen off (t)	16.4873	3.82366627
Screen on whatsapp (t)	478.904505	137.193419
Screen on twitter (t)	197.703191	49.753787
Screen on chrome (t)	321.371589	92.5058419
Screen on resto (t)	212.864296	108.845818

Tabla 4: Coeficientes y errores típicos para el caso con 5 escenarios.

5.1 Análisis sin regresión

Este enfoque es bastante menos preciso que el anterior, ya que requiere menos procesos matemáticos involucrados. Se han empleado dos casos de análisis, con dos escenarios y con cuatro (los mismos que para el análisis por regresión).

Para ambos casos, se ha contabilizado, para cada escenario, el número de ocasiones en las que la batería ha disminuido su porcentaje en un 1 por ciento con el dispositivo en ese escenario. Después, como conocemos el valor de la capacidad de la batería, basta con dividir el tiempo empleado en ese escenario (en horas) por el porcentaje de batería total consumido por ese estado (número de decrementos del 1% por el valor de ese 1% de la capacidad total).

Para cada una de las descargas de las que se tiene información recopilada (15, como en el modelo con regresión) se calculan estos coeficientes. Finalmente se obtiene la media aritmética de dichos coeficientes y su desviación típica, con el fin de conocer el error medio obtenido.

	coef_on	coef_off	coef_on_wha	coef_on_twit	coef_on_resto
Promedios	207.99065	18.878892	256.6942	200.6192	210.2306
Desviación típica	34.117973	2.7957947	107.6333	36.63438	55.8803

Tabla 5: Promedios y errores del análisis sin regresión.

En el caso de dos escenarios, los coeficientes y el error medio no varía demasiado. Los coeficientes de consumo son aproximadamente 208 y 18’9 para pantalla encendida y apagada respectivamente, mientras que en el modelo de regresión eran de 215 y 20 aproximadamente.

Cuando analizamos cuatro escenarios las diferencias aumentan. Para el enfoque mediante regresión los errores salen de un 25% aproximadamente para todos, mientras que aquí hay algunos del 18 % (Twitter) y otros que se van al 42% (Whatsapp). Esto es debido a que este método puede ser injusto a la hora de repartir los consumos cuando aumenta el número de escenarios. Por ejemplo, puede ser que un usuario esté utilizando una aplicación durante un rato sin que el porcentaje de batería cambie, pero en el momento que abre otra, coincide con que baja el porcentaje, y ese cambio se le asignaría al último escenario en lugar de al primero. Esto se podría solucionar afinando más este enfoque, por ejemplo asignando el decremento de la batería al estado que ha gastado más tiempo en ese 1% de batería, o asignarlo de manera proporcional también por tiempo transcurrido en cada estado.

6. Conclusiones y vías abiertas

Una vez realizados los análisis de la información recopilada con la aplicación InfoBatería, podemos sacar algunas conclusiones y plantear vías abiertas para el futuro.

El modelo de regresión se ajustaba muy bien a la realidad, lo que nos indica que la obtención del modelo energético para realizar estimaciones de consumos está bien planteada. Sería necesario afinar más los coeficientes de consumo añadiendo escenarios más complejos con un número mayor de trazas obtenidas, para así reducir el error de las estimaciones. Hay que tener en cuenta que, si no es con instrumentos de medida, es imposible medir de forma precisa el consumo. Ni el resto de aplicaciones estudiadas ni el propio sistema Android dan mediciones perfectas.

Las estimaciones sin emplear regresión daban resultados bastante buenos con pocos escenarios, pero empeoraban cuando se añadían más. De todas formas puede ser un buen método, por sencillo en cálculos sobre todos, para estimar el tiempo restante para el escenario de pantalla encendida o pantalla apagada, como realizaba la aplicación *Battery Monitor Widget*. Además, con más tiempo transcurrido recopilando información se sacarán medias más ajustadas, e incluso se podrían hacer medias móviles con las últimas 'x' descargas para poder apreciar fácilmente cambios en el comportamiento de la batería y asociar esos cambios a determinados eventos.

También puede ser de bastante utilidad la estimación de la capacidad real de la batería, ya que con el paso del tiempo y los usos éstas se van deteriorando.

Por último, una vez obtenida la información mediante la aplicación diseñada se pueden realizar distintos análisis y enfoques, además de los ya propuestos con mayor número de datos o con medias móviles. Se podrían realizar medias móviles también de las regresiones, igual que para el caso sin regresión. También sería posible sacar unas gráficas de consumos relacionadas a eventos, de manera que si el usuario nota cualquier cambio en las descargas pueda obtener un motivo de ese comportamiento diferente (instalación o actualización de una aplicación, actualización del sistema operativo, olvidarse activado el gps por error, etc.). Otra buena continuación del proyecto sería ampliar la aplicación creada añadiendo funcionalidades, como que el usuario pueda definir los escenarios y visualice en un gráfico la evolución del consumo de cada uno de esos escenarios.

6.1 Conclusiones a nivel personal.

Durante la realización de la carrera de Ingeniería Informática he recibido formación en diferentes campos de esta área. Sin embargo, existen 2 carencias que he observado a lo largo de estos años de carrera, que son las asignaturas relacionadas con el mundo de la investigación y las relacionadas con las tecnologías móviles. Es por esto que decidí realizar un proyecto que tuviera una buena parte de labor investigadora y que estuviera relacionada con la creación de aplicaciones para dispositivos móviles.

Esta decisión, junto a compaginar la realización del proyecto con el mundo laboral, es la que me ha llevado a dilatarme más de la cuenta en el tiempo. El proceso de búsqueda de información, lectura de artículos y análisis de propuestas de otros autores y de los resultados obtenidos son fases fundamentales en un proyecto con carga investigadora.

También el tener que aprender a programar para dispositivos móviles es una tarea que requiere tiempo, ya que he tenido que hacer un par de cursos online y múltiples consultas online sobre dudas y problemas. Además, todos los cursos que enseñan a programar par Android tienen otro enfoque, más orientado a crear aplicaciones básicas o juegos, mientras que esta aplicación debía acceder a componentes y llamadas del sistema, que son enfoques mucho más minoritarios sobre los que hay mucha menos información disponible.

Sin embargo, todas estas dificultades iniciales han hecho que enriquezca más mi formación como ingeniero y que aprenda cosas nuevas que no había visto en la carrera.

En definitiva, la realización de este proyecto ha sido una experiencia enriquecedora y satisfactoria con la que he podido iniciarme en el campo de la investigación (de hecho actualmente estoy trabando en un centro de investigación) y aprender a programar para dispositivos móviles, lo que ha mejorado mis cualidades como ingeniero y ha completado mi formación.

7. Anexos

7.1. Resultados de los análisis

Estimación de la capacidad de la batería

Fecha	% inicial	% final	Tiempo	Tiempo (h)	mA cargador	mAh carga	mAh batería	efectividad (%)	amperaje inicial de la bateria:	2070	mAh
04/12/2015	31	89	1:01:00	1.02	1000	1017	1753	84.68			
13/12/2015	18	93	1:25:00	1.42	1000	1417	1889	91.25			
15/12/2015	37	91	1:04:00	1.07	1000	1067	1975	95.43			
23/01/2016	20	90	1:19:31	1.33	1000	1325	1893	91.46			
25/01/2016	20	90	1:15:46	1.26	1000	1263	1804	87.15	mAh	efectividad	
27/01/2016	16	97	1:39:10	1.65	1000	1653	2040	98.57	Media: 1864	90.06	
27/01/2016	20	90	1:15:26	1.26	1000	1257	1796	86.76			
28/01/2016	27	90	1:12:42	1.21	1000	1212	1923	92.91			
01/02/2016	20	90	1:16:01	1.27	1000	1267	1810	87.44			
03/02/2016	17	90	1:18:51	1.31	1000	1314	1800	86.97			
05/02/2016	18	90	1:19:24	1.32	1000	1323	1838	88.79			
06/02/2016	27	90	1:15:14	1.25	1000	1254	1990	96.15			
08/02/2016	15	90	1:22:46	1.38	1000	1379	1839	88.85			
10/02/2016	12	90	1:24:47	1.41	1000	1413	1812	87.52			
11/02/2016	15	90	1:24:40	1.41	1000	1411	1881	90.89			
13/02/2016	15	90	1:22:07	1.37	1000	1369	1825	88.16			
15/02/2016	15	90	1:21:59	1.37	1000	1366	1822	88.01			

	B	C	D	E	F	G	H	I	J
1	DATOS PARA LA REGRESIÓN								
2									
3	Screen on (t)	Screen off (t)	Screen on whatsapp (t)	Screen on twitter (t)	Screen on chrome (t)	Screen on resto (t)	t total	% Bat	mAh
4	3.11616167	37.29765222	0.701073611	1.6236175	0.144849444	0.64662111	40.413814	72	1342
5	2.38323778	40.36893444	0.941984444	0.449292222	0.230541389	0.76141972	42.752172	76	1417
6	3.849855833	41.97993472	0.453858333	1.204393056	0.639907222	1.55169722	45.829791	83	1547
7	4.135099167	41.86565167	0.299939722	2.486706667	0.423689444	0.92476333	46.000751	83	1547
8	3.814606667	36.0629575	0.8055375	1.694271389	0.454706944	0.86009083	39.877564	87	1622
9	2.3642225	36.16066833	0.400139722	0.960925	0.281401111	0.72175667	38.524891	60	1119
10	4.033918889	38.06279806	0.145298889	2.7801425	0.160321944	0.94815556	42.096717	81	1510
11	2.688510278	43.61352333	0.3892925	1.236943611	0.472166667	0.59010750	46.302034	82	1529
12	2.550926111	25.37967722	0.832573611	0.465959722	0.213463611	1.03892917	27.930603	73	1361
13	4.176658056	36.56327611	0.479845833	2.279613056	0.126143056	1.29105611	40.739934	85	1585
14	4.578814444	37.49396778	0.392137222	2.358833056	0.697732778	1.13011139	42.072782	92	1715
15	3.847139722	25.35680306	0.330775278	1.648648056	1.281043611	0.58667278	29.203943	79	1473
16	3.962390556	41.53922778	0.533041111	1.682135278	0.43449	1.31272417	45.501618	94	1752
17	3.911792778	36.81194056	0.189465556	2.707609722	0.169594722	0.84512278	40.723733	83	1547
18	3.527655556	46.20559	0.2871275	1.878128056	0.337507778	1.02489222	49.733246	93	1734

22	REGRESIÓN CON 2 ESCENARIOS								
23									
24	*Estadísticas de la regresión*								
25	Coeficiente de correlación múltiple	0.996804833							
26	Coeficiente de determinación R^2	0.993619875							
27	R^2 ajustado	0.916206019							
28	Error típico	131.1356972							
29	Observaciones	15							
30									
31	ANÁLISIS DE VARIANZA								
32		*Grados de libertad*	*Suma de cuadrados*	*Promedio de los cuadrados*	*F*	*Valor crítico de F*			
33	Regresión	2	34815791.1	17407895.55	1012.28876	4.18487E-14			
34	Residuos	13	223555.4242	17196.57109					
35	Total	15	35039346.52						
36									
37		*Coeficientes*	*Error típico*	*Estadístico t*	*Probabilidad*	*Inferior 95%*	*Superior 95%*	*nferior 95.0%*	*uperior 95.0%*
38	Intercepción	0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
39	Screen on (t)	215.1227931	40.95081091	5.253199835	0.000156	126.6539448	303.591641	126.653945	303.591641
40	Screen off (t)	19.96101506	3.869565723	5.15846389	0.00018393	11.60132655	28.3207036	11.6013266	28.3207036

REGRESIÓN CON 4 ESCENARIOS

Estadísticas de la regresión	
Coeficiente de correlación múltiple	0.998154674
Coeficiente de determinación R^2	0.996312752
R^2 ajustado	0.904398048
Error típico	108.3759235
Observaciones	15

ANÁLISIS DE VARIANZA

	Grados de libertad	*Suma de cuadrados*	*Promedio de los cuadrados*	*F*	*Valor crítico de F*
Regresión	4	34910147.77	8727536.943	743.063747	2.53637E-12
Residuos	11	129198.7487	11745.34079		
Total	15	35039346.52			

	Coeficientes	*Error típico*	*Estadístico t*	*Probabilidad*	*Inferior 95%*	*Superior 95%*	*nferior 95.0%*	*uperior 95.0%*
Intercepción	0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
Screen off (t)	15.77531957	3.616165502	4.362444021	0.00113168	7.816192965	23.7344462	7.81619296	23.7344462
Screen on whatsapp (t)	463.7587815	132.6672742	3.495653199	0.00500847	171.7600797	755.757483	171.76008	755.757483
Screen on twitter (t)	193.596542	48.36197699	4.003073366	0.00207526	87.15254834	300.040536	87.1525483	300.040536
Screen on resto (t)	275.1352349	65.73718622	4.185381985	0.00152266	130.4486636	419.821806	130.448664	419.821806

159	REGRESIÓN CON 5 ESCENARIOS									
160										
161	*Estadísticas de la regresión*									
162	Coeficiente de correlación múltiple	0.998247439								
163	Coeficiente de determinación R^2	0.996497949								
164	R^2 ajustado	0.895097128								
165	Error típico	110.77436								
166	Observaciones	15								
167										
168	ANÁLISIS DE VARIANZA									
169		*Grados de libertad*	*Suma de cuadrados*	*Promedio de los cuadrados*	*F*	*Valor crítico de F*				
170	Regresión	5	34916636.93	6983327.386	569.093865	5.71984E-11				
171	Residuos	10	122709.5883	12270.95883						
172	Total	15	35039346.52							
173										
174		*Coeficientes*	*Error típico*	*Estadístico t*	*Probabilidad*	*Inferior 95%*	*Superior 95%*	*nferior 95.0%*	*uperior 95.0%*	
175	Intercepción	0	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	
176	Screen off (t)	16.48729997	3.823666266	4.311908735	0.00153248	7.967640607	25.0069593	7.96764061	25.0069593	
177	Screen on whatsapp (t)	478.9045049	137.1934193	3.490725047	0.00581596	173.2185172	784.590493	173.218517	784.590493	
178	Screen on twitter (t)	197.7031915	49.75378699	3.973631023	0.00262787	86.84484568	308.561537	86.8448457	308.561537	
179	Screen on chrome (t)	321.3715891	92.50584188	3.474068044	0.00598025	115.2557288	527.487449	115.255729	527.487449	
180	Screen on resto (t)	212.8642961	108.8458181	1.955649742	0.07900452	-29.6593001	455.387892	-29.6593001	455.387892	

2	**SIN REGRESIÓN**						coef_on	coef_off	coef_on_wha	coef_on_twit	coef_on_resto						
3			1% de bateria	18.64205924	mAh	Promedios	207.99065	18.878892	256.6942	200.6192	210.2306						
4						Desviación típica	34.117973	2.7957947	107.6333	36.63438	55.8803						
5						Porcentaje	16.403609	14.809104	41.93053	18.26066	26.58047						
6						Varianza	1164.0361	7.8164679	11584.92	1342.078	3122.607						
7		Screen on (t)	Screen off (t)	Screen on whatsapp (t)	Screen on twitter (t)	Screen on resto (t)	número de decrementos 1% screenOn	número de decrementos 1% screenOff	número de decrementos 1% whatsapp	número de decrementos 1% twitter	número de decrementos 1% resto	coef_on	coef_off	coeficiente on_whatsapp	coeficiente on_twitter	coeficiente on_resto	
8	19/01/2016	3.11616167	37.29765222	0.701073611	1.6236175	0.79147056	31	33	7	12	12	185.4537403	16.49401285	186.135111	137.781658	282.644388	
9	21/01/2016	2.38323778	40.36893444	0.941984444	0.449292222	0.99196111	34	42	16	6	12	265.9533262	19.3952726	316.643178	248.952352	225.517622	
10	23/01/2016	3.849855833	41.97993472	0.453858333	1.204393056	2.19160444	29	46	6	12	11	140.4259643	20.42725246	246.447729	185.740618	93.5673644	
11	25/01/2016	4.135099167	41.86565167	0.299939722	2.486706667	1.34845278	44	36	3	27	14	198.3629832	16.03018479	186.458057	202.410524	193.546881	
12	29/01/2016	3.814606667	36.0629575	0.8055375	1.694271389	1.31479778	39	39	8	18	13	190.5937817	20.16030744	185.139083	198.053906	184.322467	
13	30/01/2016	2.3642225	36.16066833	0.400139722	0.960925	1.00315778	20	33	3	9	8	157.7013943	17.01262679	139.766623	174.60107	148.667017	
14	01/02/2016	4.033918889	38.06279806	0.145298889	2.7801425	1.10847750	41	35	4	27	10	189.4744168	17.1419892	513.205831	181.046691	168.177155	
15	03/02/2016	2.688510278	43.61352333	0.3892925	1.236943611	1.06227417	35	39	7	16	12	242.6890753	16.67006595	335.209167	241.137062	210.590371	
16	05/02/2016	2.550926111	25.37967722	0.832573611	0.465959722	1.25239278	32	37	8	4	20	233.8546354	27.17750056	179.127073	160.031508	297.703078	
17	06/02/2016	4.176658056	36.56327611	0.479845833	2.279613056	1.41719917	42	35	5	23	14	187.4624347	17.84501124	194.250507	188.087782	184.158187	
18	08/02/2016	4.578814444	37.49396778	0.392137222	2.358833056	1.82784417	56	36	4	26	26	227.9968603	17.89925613	190.158528	205.480222	265.172244	
19	10/02/2016	3.847139722	25.35680306	0.330775278	1.648648056	1.86771639	51	27	5	21	25	247.1303591	19.85012063	281.793418	237.457135	249.530113	
20	11/02/2016	3.962390556	41.53922778	0.533041111	1.682135278	1.74721417	47	45	7	25	15	221.1232769	20.19519164	244.811164	277.059454	160.043854	
21	13/02/2016	3.911792778	36.81194056	0.189465556	2.707609722	1.01471750	44	33	2	28	14	209.6866202	16.71164154	196.785734	192.78172	257.203438	
22	15/02/2016	3.527655556	46.20559	0.2871275	1.878128056	1.36240000	42	50	7	18	17	221.9509461	20.17294795	454.482467	178.665701	232.615243	

7.2. Tecnologías utilizadas

- **Eclipse**. Entorno de desarrollo multilenguaje, utilizado para la implementación de este proyecto. Incluye la posibilidad de ser extendido por *plugins*. Se utiliza como entorno de desarrollo.
- **AndroidSDK**. Kit de Desarrollo de Software de Android que incluye un conjunto de herramientas de desarrollo. Contiene un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Se integra en el entorno de desarrollo Eclipse.
- **Java**. Lenguaje de programación multiplataforma y de propósito general utilizado para implementar la aplicación InfoBateria.
- **Android 5.1**. Versión del sistema operativo de Android donde se ha instalado y ejecutado la aplicación desarrollada.
- **Microsoft Excel**. Programa de ofimática empleado para el análisis de la información recogida por la aplicación.
- **SqliteBrowser**. Herramienta de código abierto para crear, diseñar y editar bases de datos compatible con SQLite. Se ha usado para visualizar la base de datos generada por la aplicación y para exportar la base de datos a un archivo Excel.
- **Microsoft OneDrive**. Herramienta de almacenamiento en la nube de Microsoft utilizada como repositorio y control de versiones del proyecto.
- **SQLiteViewer**. Aplicación móvil para dispositivos Android de visualización de bases de datos SQLite.

7.3. Código

7.3.1. Funciones creadas en Microsoft Excel

- **Función screenOn:**

```
Function screenOn(time As Range, Screen As Range) As Double
    Dim timeElem As Variant
    Dim screenElem As Variant
    Dim i As Integer
    Dim aux As Double

    i = 1
    For Each timeElem In time
        If i > 1 And Screen(i).Value = 0 Then
            aux = time(i).Value - time(i - 1).Value
            screenOn = screenOn + aux
        End If
```

```
        i = i + 1
    Next timeElem
End Function
```

- **Función screenOff:**

```
Function screenOff(time As Range, Screen As Range) As Double
    Dim timeElem As Variant
    Dim screenElem As Variant
    Dim i As Integer
    Dim aux As Double

    i = 2
    For Each timeElem In time
        If Screen(i).Value = 1 Then
            aux = time(i).Value - time(i - 1).Value
            screenOff = screenOff + aux
        End If
        i = i + 1
    Next timeElem
End Function
```

- **Función timeApp:**

```
Function timeApp(time As Range) As Double
    Dim timeElem As Variant
    Dim i As Integer
    Dim aux As Double

    i = 1
    For Each timeElem In time
        If i Mod 2 = 0 Then
            aux = time(i).Value - time(i - 1).Value
            timeApp = timeApp + aux
        End If
        i = i + 1
    Next timeElem
End Function
```

- **Función timesScreenOn:**

```
Function timesScreenOn(time As Range, Screen As Range, Trigger As Range) As
Integer
    Dim timeElem As Variant
    Dim screenElem As Variant
    Dim i As Integer
    Dim aux As Double
    Dim timeOn As Double
    Dim timeOff As Double

    i = 1
    timeOn = 0
    timeOff = 0
    For Each timeElem In time
```

```
      If Trigger(i).Value = "BAT" Then
         If timeOn > timeOff Then
            timesScreenOn = timesScreenOn + 1
         Else
            If timeOn = timeOff And Screen(i).Value = 1 Then
               timesScreenOn = timesScreenOn + 1
            End If
         End If
         timeOn = 0
         timeOff = 0
      Else
         If i > 1 Then
            If Screen(i).Value = 0 Then
               aux = time(i).Value - time(i - 1).Value
               timeOn = timeOn + aux
            Else
               aux = time(i).Value - time(i - 1).Value
               timeOff = timeOff + aux
            End If
         End If
      End If
      i = i + 1
   Next timeElem
End Function
```

- **Función timesScreenOff:**

```
Function timesScreenOff(time As Range, Screen As Range, Trigger As Range) As Integer
   Dim timeElem As Variant
   Dim screenElem As Variant
   Dim i As Integer
   Dim aux As Double
   Dim timeOn As Double
   Dim timeOff As Double

   i = 1
   timeOn = 0
   timeOff = 0
   For Each timeElem In time
      If Trigger(i).Value = "BAT" Then
         If timeOff > timeOn Then
            timesScreenOff = timesScreenOff + 1
         Else
            If timeOff = timeOn And Screen(i).Value = 0 Then
               timesScreenOff = timesScreenOff + 1
            End If
         End If
         timeOn = 0
         timeOff = 0
      Else
```

```
        If i > 1 Then
          If Screen(i).Value = 0 Then
            aux = time(i).Value - time(i - 1).Value
            timeOn = timeOn + aux
          Else
            aux = time(i).Value - time(i - 1).Value
            timeOff = timeOff + aux
          End If
        End If
      End If
      i = i + 1
    Next timeElem
End Function
```

7.3.2. Código de la aplicación InfoBatería

InfoBateria.java

```
package com.samuel.infobateria;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.Calendar;
import android.annotation.TargetApi;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.bluetooth.BluetoothAdapter;
import android.content.ContentResolver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.PowerManager;
import android.os.SystemClock;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.provider.Settings.Global;
import android.provider.Settings.Secure;
```

```
import android.provider.Settings.SettingNotFoundException;
import android.telephony.TelephonyManager;
import android.text.TextUtils;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

public class InfoBateria extends Activity {
	private static  TextView wifiInfo;
	private static  TextView dataInfo;
	private static  TextView syncInfo;
	private static  TextView networkInfo;
	private static  TextView screenInfo;
	private static  TextView bluetoothInfo;
	private static  TextView gpsInfo;
	private static  TextView nlpInfo;
	private static  TextView callInfo;
	private static  TextView levelInfo;
	private static  TextView plugInfo;
	private static  TextView tempInfo;
	private static  TextView voltInfo;
	private ToggleButton toggleButton;
	private Button botonDatos;
	private Button botonAcc;
	private static int batLevel=0;
	private static int volt=0;
	private static int plugged=0;
	private static double temp=0;
	private static int gps = 0;
	private static int nlp = 0;
	private static String wifi = "0";
	private static int network=0;
	private static int screen=0;
	private static int data=0;
	private static int sync=0;
	private static int call=0;
	private static int bluetooth=0;
	private static int batSaver=0;

	private static final int RESULT_SETTINGS = 1;
	private static final int TO_MILIS = 60000;
	private static int INTERVAL = 15;

	private LocationManager locationManager;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);
```

```
            locationManager = (LocationManager)
this.getSystemService(LOCATION_SERVICE);

            //inicializamos la BD
            Trazas.inicializaBD(this);
            //inicializamos los campos de la traza
            wifi = setWifiState();
            data = setDataState();
            sync = setSyncState();
            network = setNetworkType();
            gps = setGpsState();
            nlp = setNlpState();
            screen = setScreenState();
            bluetooth = setBluetoothState();

            final Intent IntentServicio = new Intent(this, Servicio.class);

            final SharedPreferences preferences =
getPreferences(MODE_PRIVATE);
            boolean tgpref = preferences.getBoolean("tgpref", false);
//default is true

            botonAcc = (Button) findViewById(R.id.buttonAcc);
            if (isAccessibilitySettingsOn(getApplicationContext())){
                  botonAcc.setText(R.string.botonAccOff);
                  botonAcc.setCompoundDrawablesWithIntrinsicBounds(
R.drawable.ic_done_black_36dp, 0, 0, 0);
            }
            else{
                  botonAcc.setText(R.string.botonAccOn);
                  botonAcc.setCompoundDrawablesWithIntrinsicBounds(
R.drawable.ic_clear_black_36dp, 0, 0, 0);
            }

            botonAcc.setOnClickListener(new OnClickListener() {
                  @Override
           public void onClick(View v) {
                        Intent intent = new
Intent(android.provider.Settings.ACTION_ACCESSIBILITY_SETTINGS);
                        startActivityForResult(intent, 0);
                  }
            });

            toggleButton = (ToggleButton) findViewById(R.id.toggleButton1);

            toggleButton.setChecked(tgpref);
            toggleButton.setOnClickListener(new OnClickListener() {

           @Override
           public void onClick(View v) {
               // TODO Auto-generated method stub
            SharedPreferences.Editor editor = preferences.edit();
               editor.putBoolean("tgpref", toggleButton.isChecked()); //
value to store
               editor.commit();
                if (toggleButton.isChecked()) {
                    scheduleAlarm();
                    startService(IntentServicio);
```

```
                    } else {
                          cancelAlarm();
                          stopService(IntentServicio);
                    }
               }
          });
               wifiInfo=(TextView)findViewById(R.id.textViewWifiInfo);
               dataInfo=(TextView)findViewById(R.id.textViewDataInfo);
               syncInfo=(TextView)findViewById(R.id.textViewSyncInfo);
               networkInfo=(TextView)findViewById(R.id.textViewNetworkInfo);
               bluetoothInfo=(TextView)findViewById(R.id.textViewBTInfo);
               gpsInfo=(TextView)findViewById(R.id.textViewGpsInfo);
               nlpInfo=(TextView)findViewById(R.id.textViewNlpInfo);
               screenInfo=(TextView)findViewById(R.id.textViewScreenInfo);
               callInfo=(TextView)findViewById(R.id.textViewCallInfo);
               levelInfo=(TextView)findViewById(R.id.textViewLevelInfo);
               plugInfo=(TextView)findViewById(R.id.textViewPluggedInfo);
               tempInfo=(TextView)findViewById(R.id.textViewTempInfo);
               voltInfo=(TextView)findViewById(R.id.textViewVoltInfo);
       }

       // To check if service is enabled
       private boolean isAccessibilitySettingsOn(Context mContext) {
                    String TAG = "isAccessibilitySettingsOn";
               int accessibilityEnabled = 0;
               final String service =
"com.samuel.infobateria/com.samuel.WindowChangeDetectingService";
               boolean accessibilityFound = false;
               try {
                   accessibilityEnabled = Settings.Secure.getInt(

mContext.getApplicationContext().getContentResolver(),

android.provider.Settings.Secure.ACCESSIBILITY_ENABLED);
                   Log.v(TAG, "accessibilityEnabled = " +
accessibilityEnabled);
               } catch (SettingNotFoundException e) {
                   Log.e(TAG, "Error finding setting, default accessibility
to not found: "
                                   + e.getMessage());
               }
               TextUtils.SimpleStringSplitter mStringColonSplitter = new
TextUtils.SimpleStringSplitter(':');

               if (accessibilityEnabled == 1) {
                   Log.v(TAG, "***ACCESSIBILIY IS ENABLED*** ----------------
-");
                   String settingValue = Settings.Secure.getString(

mContext.getApplicationContext().getContentResolver(),
                           Settings.Secure.ENABLED_ACCESSIBILITY_SERVICES);
                   if (settingValue != null) {
                       TextUtils.SimpleStringSplitter splitter =
mStringColonSplitter;
                       splitter.setString(settingValue);
                       while (splitter.hasNext()) {
                           String accessabilityService = splitter.next();
```

```
                        Log.v(TAG, "-------------- > accessabilityService
:: " + accessabilityService);
                        if
(accessabilityService.equalsIgnoreCase(service)) {
                            Log.v(TAG, "We've found the correct setting -
accessibility is switched on!");

                        }
                    }
                }
                return true;
            } else {
                Log.v(TAG, "***ACCESSIBILIY IS DISABLED***");
            }

            return accessibilityFound;
        }

    public boolean onCreateOptionsMenu(Menu menu) {

        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
      case R.id.action_settings:
      startActivityForResult(new Intent(this, PreferencesFromCode.class),
RESULT_SETTINGS);
      return true;
      default:
      return super.onOptionsItemSelected(item);
      }
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);

        switch (requestCode) {
        case RESULT_SETTINGS:
             SharedPreferences sharedPrefs =
PreferenceManager.getDefaultSharedPreferences(this);
             String option = sharedPrefs.getString("list_preference", "0");
             INTERVAL = Integer.valueOf(option);
             Log.i("INTERVAL CHANGED", "NEW INTERVAL: " + INTERVAL);
             RecibeCambios.batTrigger = INTERVAL==0;
            if (toggleButton.isChecked()) {
                cancelAlarm();
                if (INTERVAL>0){
                    scheduleAlarm();
                }
            }
            break;
        }
```

```
    }

       // Setup a recurring alarm every fifteen minutes
         public void scheduleAlarm() {
             Log.i("AlarmService", "INTERVAL: " + INTERVAL);
           // Construct an intent that will execute the AlarmReceiver
           Intent intent = new Intent(getApplicationContext(),
AlarmReceiver.class);
           // Create a PendingIntent to be triggered when the alarm goes off
           final PendingIntent pIntent = PendingIntent.getBroadcast(this,
AlarmReceiver.REQUEST_CODE,
               intent, PendingIntent.FLAG_UPDATE_CURRENT);
           // Setup periodic alarm every 5 seconds
           long firstMillis = System.currentTimeMillis(); // alarm is set
right away
//         Log.i("AlarmService", "firstMillis: " +
String.valueOf(firstMillis));
           AlarmManager alarm = (AlarmManager)
this.getSystemService(Context.ALARM_SERVICE);
           // First parameter is the type: ELAPSED_REALTIME,
ELAPSED_REALTIME_WAKEUP, RTC_WAKEUP
           // Interval can be INTERVAL_FIFTEEN_MINUTES, INTERVAL_HALF_HOUR,
INTERVAL_HOUR, INTERVAL_DAY
//         Calendar calendar = Calendar.getInstance();
           alarm.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
SystemClock.elapsedRealtime(),
               INTERVAL*TO_MILIS, pIntent);
//         Log.i("AlarmService", "call: " +
String.valueOf(System.currentTimeMillis()));
         }

         public void cancelAlarm() {
                 Intent intent = new Intent(getApplicationContext(),
AlarmReceiver.class);
                 final PendingIntent pIntent =
PendingIntent.getBroadcast(this, AlarmReceiver.REQUEST_CODE,
                    intent, PendingIntent.FLAG_UPDATE_CURRENT);
                 AlarmManager alarm = (AlarmManager)
this.getSystemService(Context.ALARM_SERVICE);
                 alarm.cancel(pIntent);
                 Log.i("AlarmService", "Service stopped");
             }

         public void updateData(){
               wifi = setWifiState();
             data = setDataState();
             sync = setSyncState();
             network = setNetworkType();
             gps = setGpsState();
             nlp = setNlpState();
             screen = setScreenState();
             bluetooth = setBluetoothState();
       actualizarTextoValores();
         }

       @TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR1)
        public boolean airplane17(){ // API 17 o mayor
              return Settings.Global.
```

```
                                getInt(this.getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) == 1;
        }

        public boolean airplane(){ // API 16 o menor
                return Settings.System.
                                getInt(this.getContentResolver(),
Settings.System.AIRPLANE_MODE_ON, 0) == 1;
        }


        public int setNetworkType (){

                boolean isAirplaneEnabled;
                if(Build.VERSION.SDK_INT <= Build.VERSION_CODES.JELLY_BEAN_MR1){
                        isAirplaneEnabled = airplane();
                }else{
                        isAirplaneEnabled = airplane17();
                }
                if (isAirplaneEnabled) {
                        return 0;
                }else {
                        TelephonyManager phoneInfo = (TelephonyManager)
getApplicationContext().getSystemService(Context.TELEPHONY_SERVICE);
                        int networkType = phoneInfo.getNetworkType();

                        switch(networkType){
                case TelephonyManager.NETWORK_TYPE_1xRTT:
                    return 2; // ~ 50-100 kbps
                case TelephonyManager.NETWORK_TYPE_CDMA:
                    return 2; // ~ 14-64 kbps
                case TelephonyManager.NETWORK_TYPE_EDGE:
                    return 2; // ~ 50-100 kbps
                case TelephonyManager.NETWORK_TYPE_EVDO_0:
                    return 3; // ~ 400-1000 kbps
                case TelephonyManager.NETWORK_TYPE_EVDO_A:
                    return 3; // ~ 600-1400 kbps
                case TelephonyManager.NETWORK_TYPE_GPRS:
                    return 2; // ~ 100 kbps
                case TelephonyManager.NETWORK_TYPE_HSDPA:
                    return 3; // ~ 2-14 Mbps
                case TelephonyManager.NETWORK_TYPE_HSPA:
                    return 3; // ~ 700-1700 kbps
                case TelephonyManager.NETWORK_TYPE_HSUPA:
                    return 3; // ~ 1-23 Mbps
                case TelephonyManager.NETWORK_TYPE_UMTS:
                    return 3; // ~ 400-7000 kbps
                /*
                 * Above API level 7, make sure to set android:targetSdkVersion
                 * to appropriate level to use these
                 */
                case TelephonyManager.NETWORK_TYPE_EHRPD: // API level 11
                    return 3; // ~ 1-2 Mbps
                case TelephonyManager.NETWORK_TYPE_EVDO_B: // API level 9
                    return 3; // ~ 5 Mbps
                case TelephonyManager.NETWORK_TYPE_HSPAP: // API level 13
                    return 3; // ~ 10-20 Mbps
                case TelephonyManager.NETWORK_TYPE_IDEN: // API level 8
```

```
                return 2; // ~25 kbps
            case TelephonyManager.NETWORK_TYPE_LTE: // API level 11
                return 2; // ~ 10+ Mbps
            // Unknown
            case TelephonyManager.NETWORK_TYPE_UNKNOWN:
            default:
                return 1;
            }
             }

        }

        public int setScreenState() {
              PowerManager powerManager = (PowerManager)
getSystemService(POWER_SERVICE);
              if (powerManager.isScreenOn()) {
                    return 1;
              } else {
                    return 0;
              }
        }

        public static int setSyncState() {
              boolean isMasterSyncEnabled =
ContentResolver.getMasterSyncAutomatically();
              if (isMasterSyncEnabled) {
                    return 1;
              } else {
                    return 0;
              }
        }

        public String setWifiState() {
              WifiManager wifi =
(WifiManager)this.getSystemService(Context.WIFI_SERVICE);
              if (wifi.isWifiEnabled()){
              //wifi is enabled
                    if
(wifi.getConnectionInfo().getSupplicantState().toString().equals("DISCONNECTE
D") ||
wifi.getConnectionInfo().getSupplicantState().toString().equals("SCANNING")){
                          return "1";
                    }else{
                          String SSID = wifi.getConnectionInfo().getSSID();
                          return "2_" + SSID.substring(1, SSID.length()-1);
                    }
              }
              else{
                    return "0";
              }
        }

        public int setBluetoothState() {
              BluetoothAdapter bluetoothAdapter;
              bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
              if (bluetoothAdapter == null){
                    return -1;
              }else{
```

```
                    if (bluetoothAdapter.isEnabled()) {
                          return 1;
                    } else {
                          return 0;
                    }
              }

       }

       public int setDataState() {
             boolean mobileDataEnabled = false; // Assume disabled
                ConnectivityManager cm = (ConnectivityManager)
this.getSystemService(Context.CONNECTIVITY_SERVICE);
                try {
                    Class cmClass = Class.forName(cm.getClass().getName());
                    Method method =
cmClass.getDeclaredMethod("getMobileDataEnabled");
                    method.setAccessible(true); // Make the method callable
                    // get the setting for "mobile data"
                    mobileDataEnabled = (Boolean)method.invoke(cm);
                } catch (Exception e) {
                    // Some problem accessible private API
                }
                if (mobileDataEnabled) {
                   return 1;
                } else {
                   return 0;
                }
       }

       public int setGpsState() {
             boolean isGPSEnabled = false;
                   isGPSEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
                   if (isGPSEnabled) {
                          return 1;
                   }
                   else {
                          return 0;
                   }
       }

       public int setNlpState(){
             if (Settings.Secure.getInt(this.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 0) {
                          return 0;
                   }
                   else if
(Settings.Secure.getInt(this.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 1){
                          return 1;
                   }
                   else if
(Settings.Secure.getInt(this.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 2){
                          return 2;
                   }
```

```
                else if
(Settings.Secure.getInt(this.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 3){
                        return 3;
                }
                else{
                        return -1;
                }
        }

        public static void actualizarTextoValores() {
                wifiInfo.setText(wifi);
                dataInfo.setText(String.valueOf(data));
                syncInfo.setText(String.valueOf(sync));
                networkInfo.setText(String.valueOf(network));
                gpsInfo.setText(String.valueOf(gps));
                nlpInfo.setText(String.valueOf(nlp));
                screenInfo.setText(String.valueOf(screen));
                bluetoothInfo.setText(String.valueOf(bluetooth));
                callInfo.setText(String.valueOf(call));
                levelInfo.setText(batLevel+"%");
                plugInfo.setText(String.valueOf(plugged));
                tempInfo.setText(""+temp+" C");
                voltInfo.setText(volt+" mV");
        }

        @Override
        protected void onResume(){
                super.onResume();
                updateData();
                if (isAccessibilitySettingsOn(getApplicationContext())){
                        botonAcc.setText(R.string.botonAccOn);
                        botonAcc.setCompoundDrawablesWithIntrinsicBounds(
R.drawable.ic_clear_black_36dp, 0, 0, 0);
                }
                else{
                        botonAcc.setText(R.string.botonAccOff);
                        botonAcc.setCompoundDrawablesWithIntrinsicBounds(
R.drawable.ic_done_black_36dp, 0, 0, 0);
                }
        }

        @Override
        protected void onPause() {
                super.onPause();
        }

        @Override
        protected void onDestroy() {
                super.onDestroy();
                toggleButton.setChecked(false);
        }

        public static int getBatLevel() {
                return batLevel;
        }
        public static void setBatLevel(int level) {
                batLevel = level;
```

```
}
public static int getVolt() {
	return volt;
}
public static void setVolt(int vol) {
	volt = vol;
}
public static double getTemp() {
	return temp;
}
public static void setTemp(double tem) {
	temp = tem;
}
public static int getPlugged() {
	return plugged;
}
public static void setPlugged(int plug) {
	plugged = plug;
}
public static int getData() {
	return data;
}
public static void setData(int d) {
	data = d;
}
public static String getWifi() {
	return wifi;
}
public static void setWifi(String w) {
	wifi = w;
}
public static int getSync() {
	return sync;
}
public static void setSync(int s) {
	sync = s;
}
public static int getNetwork() {
	return network;
}
public static void setNetwork(int n) {
	network = n;
}
public static int getGps() {
	return gps;
}
public static void setGps(int g) {
	gps = g;
}
public static int getNlp() {
	return nlp;
}
public static void setNlp(int n) {
	nlp = n;
}
public static int getScreen() {
	return screen;
}
```

```
	public static void setScreen(int s) {
		screen = s;
	}
	public static int getBluetooth() {
		return bluetooth;
	}
	public static void setBluetooth(int b) {
		bluetooth = b;
	}
	public static int getCall() {
		return call;
	}
	public static void setCall(int c) {
		call = c;
	}
	public static int getBatSaver() {
		return batSaver;
	}
	public static void setBatSaver(int bs) {
		batSaver = bs;
	}

}
```

RecibeCambios.java

```
package com.samuel.infobateria;

import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.location.GpsStatus;
import android.location.GpsStatus.Listener;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.net.wifi.WifiManager;
import android.os.BatteryManager;
import android.os.Build;
import android.os.PowerManager;
import android.provider.Settings;
import android.provider.Settings.Secure;
import android.provider.Settings.SettingNotFoundException;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.Toast;

public class RecibeCambios extends BroadcastReceiver {
	Context contexto;
	static String app_package = "";
	static boolean batTrigger = false;
	public RecibeCambios (Context ctx){
		super();
```

```
            this.contexto=ctx;
      }

    @Override
    public void onReceive(Context context, Intent intent) {
      if (intent.getAction().equals(Intent.ACTION_BATTERY_CHANGED)) {
//valores bateria
            int antLevel = InfoBateria.getBatLevel();

      InfoBateria.setBatLevel(intent.getIntExtra(BatteryManager.EXTRA_LEVEL,
0));

      InfoBateria.setPlugged(intent.getIntExtra(BatteryManager.EXTRA_PLUGGED
,0));


      InfoBateria.setTemp((intent.getIntExtra(BatteryManager.EXTRA_TEMPERATU
RE,0))/10.0);

      InfoBateria.setVolt(intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE,0)
);
                                    if (batTrigger && antLevel !=
InfoBateria.getBatLevel()){
                                          crearTraza("bat");
                                    }
                                    InfoBateria.actualizarTextoValores();
            } else if (intent.getAction().equals(Intent.ACTION_SCREEN_ON)) {
// pantalla apagada
                  InfoBateria.setScreen(1);
                  crearTraza("screen");
            } else if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF))
{ // pantalla encendida
                  InfoBateria.setScreen(0);
                  crearTraza("screen");
            } else if
(intent.getAction().equals(Intent.ACTION_POWER_CONNECTED) ||
intent.getAction().equals(Intent.ACTION_POWER_DISCONNECTED)) { // enchufar o
desenchufar el cable

                  InfoBateria.setPlugged(InfoBateria.getPlugged());
                  crearTraza("plugged");
            }
            else
if(intent.getAction().equals(WifiManager.WIFI_STATE_CHANGED_ACTION)){ // wifi
desconectado
                  Log.i("wifi_state", "WIFI_STATE: " +
intent.getIntExtra(WifiManager.EXTRA_WIFI_STATE, 0));

                  WifiManager wm = (WifiManager)
context.getSystemService(Context.WIFI_SERVICE);
                  Log.i("wifi_state", "WIFI_INFO_SUPL: " +
wm.getConnectionInfo().getSupplicantState().toString());
               if(wm.isWifiEnabled() &&
(wm.getConnectionInfo().getSupplicantState().toString().equals("DISCONNECTED"
) ||
wm.getConnectionInfo().getSupplicantState().toString().equals("SCANNING"))
                           && !("1").equals(InfoBateria.getWifi())) {
                    InfoBateria.setWifi("1");
```

```
                    crearTraza("wifi");
                }else if(!wm.isWifiEnabled() &&
!("0").equals(InfoBateria.getWifi())) {
                    InfoBateria.setWifi("0");
                    crearTraza("wifi");
                }
            }
            else
if(intent.getAction().equals(WifiManager.NETWORK_STATE_CHANGED_ACTION)){ //
wifi conectado
                NetworkInfo netInfo = (NetworkInfo)
intent.getParcelableExtra(WifiManager.EXTRA_NETWORK_INFO);
                WifiManager wm = (WifiManager)
context.getSystemService(Context.WIFI_SERVICE);
                Log.i("wifi_state", "NET_STATE: " +
netInfo.getDetailedState().toString());
              if(netInfo.getDetailedState().toString().equals("CONNECTED") &&
!(InfoBateria.getWifi().startsWith("2"))) {
               String SSID = wm.getConnectionInfo().getSSID();
                    InfoBateria.setWifi("2_" + SSID.substring(1,
SSID.length()-1));
                    crearTraza("wifi");
                }
              else
if((netInfo.getDetailedState().toString().equals("DISCONNECTED") ||
netInfo.getDetailedState().toString().equals("SCANNING")) &&
InfoBateria.getWifi().startsWith("2")) {
                    InfoBateria.setWifi("1");
                    crearTraza("wifi");
                }
            }
            else
if(intent.getAction().equals(ConnectivityManager.CONNECTIVITY_ACTION)){ //
data on/off and network type

               NetworkInfo netInfo;
               if(Build.VERSION.SDK_INT < 14){
                    netInfo = (NetworkInfo)
intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
                }else{
                      ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
                      netInfo = cm.getActiveNetworkInfo();
                }
               int netAnt = InfoBateria.getNetwork();
               boolean isAirplaneEnabled;
                    if(Build.VERSION.SDK_INT <=
Build.VERSION_CODES.JELLY_BEAN_MR1){
                      isAirplaneEnabled = airplane();
                }else{
                      isAirplaneEnabled = airplane17();
                }
                if (isAirplaneEnabled) {
                      InfoBateria.setNetwork(0);
                crearTraza("network");
                }
                else {
                      if (netInfo == null){
```

```
                                if (InfoBateria.getData() != 0){
                                      InfoBateria.setData(0);
                                Log.i("data_state", "NET_INFO: ");
                                  crearTraza("data");
                                }
                           }
                           else {
                     if (netAnt == setNetworkType()) {
                           if (netInfo.isConnected()){
                                  Log.i("data_state", "NET_INFO: " +
netInfo.getDetailedState().toString());
                                  if (InfoBateria.getData() != 1 &&
InfoBateria.getWifi().startsWith("2")) {
                                  InfoBateria.setData(1);
                                  crearTraza("data");
                                  }
                           }
                     } else {
                           if(netInfo.getType()==0){ //TYPE_MOBILE
                                  if (netInfo.isAvailable()){
                                  InfoBateria.setNetwork(setNetworkType());
                                  crearTraza("network");
                                }
                                else{
                                      InfoBateria.setNetwork(1);
                                      crearTraza("network");
                                }
                           }
                     }
                           }
                     }
               }
               else
if(intent.getAction().equals(TelephonyManager.ACTION_PHONE_STATE_CHANGED)){
                     TelephonyManager phoneInfo = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
                     if
(phoneInfo.getCallState()==TelephonyManager.CALL_STATE_OFFHOOK){
                           if(InfoBateria.getCall() == 0){
                                  InfoBateria.setCall(1);
                                  crearTraza("call");
                           }
                     }else if
(phoneInfo.getCallState()==TelephonyManager.CALL_STATE_IDLE){
                           if(InfoBateria.getCall() == 1){
                                  InfoBateria.setCall(0);
                                  crearTraza("call");
                           }
                     }
               }
               else if
(intent.getAction().equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
                     BluetoothAdapter bluetooth =
BluetoothAdapter.getDefaultAdapter();
                     if (bluetooth.getState() == 12){ //STATE_ON
                           InfoBateria.setBluetooth(1);
                           crearTraza("bluetooth");
                     } else if (bluetooth.getState() == 10) { //STATE_OFF
```

```
                        InfoBateria.setBluetooth(0);
                        crearTraza("bluetooth");
                    }
                }
                else
if(intent.getAction().equals(LocationManager.PROVIDERS_CHANGED_ACTION)){
                    boolean isGPSEnabled = false;
                    LocationManager locationManager = (LocationManager)
contexto.getSystemService(Context.LOCATION_SERVICE);
                    Listener gpsListener = new Listener(){
                                public void onGpsStatusChanged(int event) {

                                    switch(event)
                                {
                                case GpsStatus.GPS_EVENT_STARTED:
                                     if (InfoBateria.getGps() == 1){
                                           InfoBateria.setGps(2);
                                           crearTraza("gps");
                                     }
                                    break;
                                case GpsStatus.GPS_EVENT_STOPPED:
                                     if (InfoBateria.getGps() == 2){
                                           InfoBateria.setGps(1);
                                           crearTraza("gps");
                                     }
                                    break;
                                default:
                                     break;
                                }
                            }
                    };
                            isGPSEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
                            if (isGPSEnabled) {
                                  if (InfoBateria.getGps() != 1){
                                        InfoBateria.setGps(1);
                                        crearTraza("gps");
                                  }

      locationManager.addGpsStatusListener(gpsListener);
                            }
                            else {

      locationManager.removeGpsStatusListener(gpsListener);
                                  if (InfoBateria.getGps() != 0){
                                        InfoBateria.setGps(0);
                                        crearTraza("gps");
                                  }
                            }
                }
                else if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
intent.getAction().equals(LocationManager.MODE_CHANGED_ACTION)){
                    Log.i("nlp_state", "LOCATION_MODE: " +
Settings.Secure.getInt(contexto.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0));
                            if
(Settings.Secure.getInt(contexto.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 0) {
```

```
                                    InfoBateria.setNlp(0);
                                    crearTraza("nlp");
                              }
                              else if
(Settings.Secure.getInt(contexto.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 1 && InfoBateria.getNlp()!=1){
                                    InfoBateria.setNlp(1);
                                    crearTraza("nlp");
                              }
                              else if
(Settings.Secure.getInt(contexto.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 2){
                                    InfoBateria.setNlp(2);
                                    crearTraza("nlp");
                              }
                              else if
(Settings.Secure.getInt(contexto.getContentResolver(),
Settings.Secure.LOCATION_MODE, 0) == 3 && InfoBateria.getNlp()!=3){
                                    InfoBateria.setNlp(3);
                                    crearTraza("nlp");
                              }
                  }
                  else if (intent.getAction().equals(Intent.ACTION_PACKAGE_ADDED))
{
                     Uri data = intent.getData();
                     app_package = data.getEncodedSchemeSpecificPart();
                     Log.i("app_install", "PACKAGE: " + app_package);
                     crearTraza("ins");
                  }
                  else if
(intent.getAction().equals(Intent.ACTION_PACKAGE_REMOVED)) {
                     Uri data = intent.getData();
                     app_package = data.getEncodedSchemeSpecificPart();
                     Log.i("app_uninstall", "PACKAGE: " + app_package);
                     crearTraza("uins");
                  }
                  else if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP &&
intent.getAction().equals(PowerManager.ACTION_POWER_SAVE_MODE_CHANGED)){
                     PowerManager pm = (PowerManager)
context.getSystemService(Context.POWER_SERVICE);
                     if ( pm.isPowerSaveMode()){
                           InfoBateria.setBatSaver(1);
                           crearTraza("bsv");
                     } else{
                           InfoBateria.setBatSaver(0);
                           crearTraza("bsv");
                     }
                  }
          }

          @TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR1)
              public boolean airplane17(){ // API 17 o mayor
                    return Settings.Global.
                                    getInt(contexto.getContentResolver(),
Settings.Global.AIRPLANE_MODE_ON, 0) == 1;
              }

             public boolean airplane(){ // API 16 o menor
```

```
            return Settings.System.
                                    getInt(contexto.getContentResolver(),
Settings.System.AIRPLANE_MODE_ON, 0) == 1;
        }

       public int setNetworkType (){

             boolean isAirplaneEnabled;
             if(Build.VERSION.SDK_INT <= Build.VERSION_CODES.JELLY_BEAN_MR1){
                    isAirplaneEnabled = airplane();
             }else{
                    isAirplaneEnabled = airplane17();
             }
             if (isAirplaneEnabled) {
                    return 0;
             }else {
                    TelephonyManager phoneInfo = (TelephonyManager)
contexto.getSystemService(Context.TELEPHONY_SERVICE);
                    int networkType = phoneInfo.getNetworkType();

                    switch(networkType){
            case TelephonyManager.NETWORK_TYPE_1xRTT:
                return 2; // ~ 50-100 kbps
            case TelephonyManager.NETWORK_TYPE_CDMA:
                return 2; // ~ 14-64 kbps
            case TelephonyManager.NETWORK_TYPE_EDGE:
                return 2; // ~ 50-100 kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_0:
                return 3; // ~ 400-1000 kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_A:
                return 3; // ~ 600-1400 kbps
            case TelephonyManager.NETWORK_TYPE_GPRS:
                return 2; // ~ 100 kbps
            case TelephonyManager.NETWORK_TYPE_HSDPA:
                return 3; // ~ 2-14 Mbps
            case TelephonyManager.NETWORK_TYPE_HSPA:
                return 3; // ~ 700-1700 kbps
            case TelephonyManager.NETWORK_TYPE_HSUPA:
                return 3; // ~ 1-23 Mbps
            case TelephonyManager.NETWORK_TYPE_UMTS:
                return 3; // ~ 400-7000 kbps
            /*
             * Above API level 7, make sure to set android:targetSdkVersion
             * to appropriate level to use these
             */
            case TelephonyManager.NETWORK_TYPE_EHRPD: // API level 11
                return 3; // ~ 1-2 Mbps
            case TelephonyManager.NETWORK_TYPE_EVDO_B: // API level 9
                return 3; // ~ 5 Mbps
            case TelephonyManager.NETWORK_TYPE_HSPAP: // API level 13
                return 3; // ~ 10-20 Mbps
            case TelephonyManager.NETWORK_TYPE_IDEN: // API level 8
                return 2; // ~25 kbps
            case TelephonyManager.NETWORK_TYPE_LTE: // API level 11
                return 2; // ~ 10+ Mbps
            // Unknown
            case TelephonyManager.NETWORK_TYPE_UNKNOWN:
            default:
```

```
                return 1;
            }
             }

        }

        public static void crearTraza (String trigger){
              // crear nueva traza
                    Traza  traza, traza_nueva;
                    traza_nueva = new Traza();
                    // introducir en base de datos
                    int id = Trazas.nueva();
                    if (id>1){
                          int id_ant = id-1;
                          traza = Trazas.elemento(id_ant);
                          traza.setFecha(traza_nueva.getFecha());
                          traza.setDate(traza_nueva.getDate());
                          traza.setPorcentaje(InfoBateria.getBatLevel());
                          traza.setVoltage(InfoBateria.getVolt());
                          traza.setTemperatura(InfoBateria.getTemp());
                          traza.setPlugged(InfoBateria.getPlugged());
                          InfoBateria.setSync(InfoBateria.setSyncState());
                          traza.setSync(InfoBateria.getSync());
                          traza.setInstalled("");
                          traza.setUninstalled("");

                          if (trigger.equals("bluetooth")){
                                traza.setTrigger("'BT'");

traza.setBluetooth(InfoBateria.getBluetooth());
                          }
                          else if (trigger.equals("screen")){
                                traza.setTrigger("'SC'");
                                traza.setScreen(InfoBateria.getScreen());
                          }
                          else if (trigger.equals("plugged")){
                                traza.setTrigger("'PL'");
                                traza.setPlugged(InfoBateria.getPlugged());
                          }
                          else if (trigger.equals("wifi")){
                                traza.setTrigger("'WI'");
                                traza.setWifi(InfoBateria.getWifi());
                          }
                          else if (trigger.equals("data")){
                                traza.setTrigger("'DA'");
                                traza.setData(InfoBateria.getData());
                          }
                          else if (trigger.equals("network")){
                                traza.setTrigger("'NE'");
                                traza.setNetwork(InfoBateria.getNetwork());
                          }
                          else if (trigger.equals("gps")){
                                traza.setTrigger("'GPS'");
                                traza.setGps(InfoBateria.getGps());
                          }
                          else if (trigger.equals("nlp")){
                                traza.setTrigger("'NLP'");
                                traza.setNlp(InfoBateria.getNlp());
```

```
                        }
                        else if (trigger.equals("call")){
                              traza.setTrigger("'CA'");
                              traza.setCall(InfoBateria.getCall());
                        }
                        else if (trigger.equals("ins")){
                              traza.setTrigger("'INS'");
                              traza.setInstalled(app_package);
                              app_package = "";
                        }
                        else if (trigger.equals("uins")){
                              traza.setTrigger("'UINS'");
                              traza.setUninstalled(app_package);
                              app_package = "";
                        }
                        else if (trigger.equals("bsv")){
                              traza.setTrigger("'BATS'");

traza.setBatSaver(InfoBateria.getBatSaver());
                        }
                        else if (trigger.equals("bat")){
                              traza.setTrigger("'BAT'");
                        }
                        else if (trigger.equals("foreground")){
                              traza.setTrigger("'FA'");
                              traza.setForeground(app_package);
                              app_package = "";
                        }
                        Trazas.actualizaTraza(id, traza);
                        InfoBateria.actualizarTextoValores();
                  }
                  else {
                        Trazas.borrar(id);
                  }
      }
}
```

Servicio.java

```
package com.samuel.infobateria;

import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiManager;
import android.os.BatteryManager;
import android.os.Handler;
import android.os.IBinder;
import android.os.PowerManager;
import android.telephony.TelephonyManager;
import android.widget.Toast;
import android.accessibilityservice.*;
```

```
public class Servicio extends Service {
    private RecibeCambios cambios;
    private RecibeCambios cambios_data;
    IntentFilter filter;
    IntentFilter filter_data;

      @Override
      public IBinder onBind(Intent intent) {
            return null;
      }

      @Override
      public void onCreate()
    {
            cambios = new RecibeCambios(this);
            cambios_data = new RecibeCambios(this);
            filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
        filter.addAction(Intent.ACTION_SCREEN_OFF);
        filter.addAction(Intent.ACTION_SCREEN_ON);
        filter.addAction(Intent.ACTION_POWER_CONNECTED);
        filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
        filter.addAction(ConnectivityManager.CONNECTIVITY_ACTION);
        filter.addAction(WifiManager.WIFI_STATE_CHANGED_ACTION);
        filter.addAction(WifiManager.NETWORK_STATE_CHANGED_ACTION);
        filter.addAction(LocationManager.PROVIDERS_CHANGED_ACTION);
        filter.addAction(LocationManager.MODE_CHANGED_ACTION);
        filter.addAction(TelephonyManager.ACTION_PHONE_STATE_CHANGED);
        filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
        filter.addAction(PowerManager.ACTION_POWER_SAVE_MODE_CHANGED);

        filter_data = new IntentFilter(Intent.ACTION_PACKAGE_ADDED);
        filter_data.addAction(Intent.ACTION_PACKAGE_REMOVED);
        filter_data.addDataScheme("package");

        registerReceiver(cambios_data, filter_data);
        registerReceiver(cambios, filter);
        Toast.makeText(getApplicationContext(), "Seguimiento de estadísticas
activado", Toast.LENGTH_SHORT).show();
    }

            @Override
            public void onDestroy()
          {
                  Toast.makeText(this, "Seguimiento de estadísticas
detenido", Toast.LENGTH_SHORT).show();
                  unregisterReceiver(cambios);
                  unregisterReceiver(cambios_data);
                  super.onDestroy();
          }

            @Override
            public int onStartCommand(Intent intent, int flags, int startId)
{
                  registerReceiver(cambios_data, filter_data);
              registerReceiver(cambios, filter);
              return START_STICKY;
      }
```

```
}
```

TrazasBD.java

```
package com.samuel.infobateria;

import java.sql.Date;
import java.text.SimpleDateFormat;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Environment;

public class TrazasBD extends SQLiteOpenHelper {
       static SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
       static Long today = System.currentTimeMillis();
    static Date resultdate = new Date(today);
    static String date = sdf.format(resultdate);

   public TrazasBD(Context contexto) {
         super(contexto,
Environment.getExternalStorageDirectory().getAbsolutePath()
                    + "/Databases/" + "trazas.db", null, 1);
   }

   @Override public void onCreate(SQLiteDatabase bd) {
      bd.execSQL("CREATE TABLE trazas ("+
             "_id INTEGER PRIMARY KEY AUTOINCREMENT, "+
             "fecha LONG, "+
             "date TEXT, " +
             "trigger TEXT, " +
             "wifi TEXT,  " +
             "data INTEGER,  " +
             "sync INTEGER, " +
             "network INTEGER, " +
             "gps INTEGER, " +
             "nlp INTEGER, " +
             "screen INTEGER, " +
             "porcentaje INTEGER, " +
             "voltaje INTEGER, " +
             "temperatura INTEGER, " +
             "plugged INTEGER, " +
             "bluetooth INTEGER, " +
             "call INTEGER, " +
             "installed TEXT, " +
             "uninstalled TEXT, " +
             "bat_saver INTEGER, " +
             "foreground TEXT)");
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   }
}
```

Trazas.java

```
package com.samuel.infobateria;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class Trazas {

       private static TrazasBD trazasBD;

    public static Traza elemento(int id) {
        Traza traza = null;
        SQLiteDatabase bd = trazasBD.getReadableDatabase();
        Cursor cursor = bd.rawQuery("SELECT * FROM trazas WHERE _id = " + id,
null);
        if (cursor.moveToNext()){
            traza = new Traza();
            traza.setFecha(cursor.getLong(1));
            traza.setDate(cursor.getString(2));
            traza.setTrigger(cursor.getString(3));
            traza.setWifi(cursor.getString(4));
            traza.setData(cursor.getInt(5));
            traza.setSync(cursor.getInt(6));
            traza.setNetwork(cursor.getInt(7));
            traza.setGps(cursor.getInt(8));
            traza.setNlp(cursor.getInt(9));
            traza.setScreen(cursor.getInt(10));
            traza.setPorcentaje(cursor.getInt(11));
            traza.setVoltage(cursor.getInt(12));
            traza.setTemperatura(cursor.getDouble(13));
            traza.setPlugged(cursor.getInt(14));
            traza.setBluetooth(cursor.getInt(15));
            traza.setCall(cursor.getInt(16));
            traza.setInstalled(cursor.getString(17));
            traza.setUninstalled(cursor.getString(18));
            traza.setBatSaver(cursor.getInt(19));
            traza.setForeground(cursor.getString(20));
        }
        cursor.close();
        bd.close();
        return traza;
    }

    public static int nueva() {
        int id = -1;
        int batLevel = InfoBateria.getBatLevel();
        int volt = InfoBateria.getVolt();
        double temp = InfoBateria.getTemp();
        int plugged = InfoBateria.getPlugged();
        Traza traza = new Traza(batLevel, volt, temp, plugged);
        SQLiteDatabase bd = trazasBD.getWritableDatabase();
        bd.execSQL("INSERT INTO trazas (fecha, date) VALUES ( "+
           traza.getFecha() + ", " + traza.getDate() +")");
        Cursor c = bd.rawQuery("SELECT _id FROM trazas WHERE fecha = " +
                                    traza.getFecha(), null);
        if (c.moveToNext()){
            id = c.getInt(0);
        }
```

```
        c.close();
        bd.close();
        return id;
    }

    public static void borrar(int id) {
        SQLiteDatabase bd = trazasBD.getWritableDatabase();
        bd.execSQL("DELETE FROM trazas WHERE _id = " + id );
        if (id==1){
             bd.execSQL("DELETE FROM SQLITE_SEQUENCE WHERE NAME = 'trazas'");
        }
        bd.close();
    }

    public static void inicializaBD(Context contexto){
        trazasBD = new TrazasBD(contexto);
     }

     public static Cursor listado() {
        SQLiteDatabase bd = trazasBD.getReadableDatabase();
        return bd.rawQuery("SELECT * FROM trazas", null);
     }

     public static void actualizaTraza(int id, Traza traza){
          SQLiteDatabase bd = trazasBD.getWritableDatabase();
          bd.execSQL("UPDATE trazas SET fecha = "+ traza.getFecha() +
             " , date = " + traza.getDate() +
             " , trigger = " + traza.getTrigger() +
              " , wifi = " + traza.getWifi()  +
              " , data = " + traza.getData()  +
              " , sync = " + traza.getSync() +
              " , network = " + traza.getNetwork()  +
              " , gps = " + traza.getGps()  +
              " , nlp = " + traza.getNlp()  +
              " , screen = " + traza.getScreen()  +
              " , porcentaje = " + traza.getPorcentaje()  +
              " , voltaje = " + traza.getVoltaje()  +
              " , temperatura = " + traza.getTemp()  +
              " , plugged = " + traza.getPlugged()  +
              " , bluetooth = " + traza.getBluetooth()  +
              " , call = " + traza.getCall()  +
              " , installed = " + traza.getInstalled()  +
              " , uninstalled = " + traza.getUnistalled()  +
              " , bat_saver = " + traza.getBatSaver()  +
              " , foreground = " + traza.getForeground()  +
              " WHERE _id = "+ id);
             bd.close();
      }

            public static int buscarNombre(String nombre) {
                int id = -1;
                SQLiteDatabase bd = trazasBD.getReadableDatabase();
                Cursor c = bd.rawQuery("SELECT * FROM trazas WHERE nombre =
'" + nombre + "'", null);
                if (c.moveToNext()){
                    id = c.getInt(0);
                }
                c.close();
```

```
                bd.close();
                return id;
            }

             public static int primerId() {
                    int id = -1;
                    SQLiteDatabase bd = trazasBD.getReadableDatabase();
                    Cursor c = bd.rawQuery("SELECT _id FROM trazas LIMIT
1", null);

                    if (c.moveToNext()){
                       id = c.getInt(0);
                    }
                    c.close();
                    bd.close();
                    return id;
                }
  }
```

AlarmService.java

```
package com.samuel.infobateria;

import android.app.IntentService;
import android.content.Intent;
import android.util.Log;

public class AlarmService extends IntentService {
       private Traza traza, traza_nueva;

    public AlarmService() {
       super("AlarmService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
      Log.i("AlarmService", "Service running");
      // crear nueva traza
                    traza_nueva = new Traza();
                    // introducir en base de datos
                    int id = Trazas.nueva();
                    if (id>1) {
                           int id_ant = id-1;
                           traza = Trazas.elemento(id_ant);
                           traza.setFecha(traza_nueva.getFecha());
                           traza.setDate(traza_nueva.getDate());
                           traza.setTrigger("'date'");
                           traza.setPorcentaje(InfoBateria.getBatLevel());
                           traza.setVoltage(InfoBateria.getVolt());
                           traza.setTemperatura(InfoBateria.getTemp());
                           traza.setPlugged(InfoBateria.getPlugged());
                           traza.setInstalled("");
                           traza.setUninstalled("");
                           Trazas.actualizaTraza(id, traza);

                    } else {
                           traza_nueva.setTrigger("'date1'");
                           traza_nueva.setWifi(InfoBateria.getWifi());
                           traza_nueva.setData(InfoBateria.getData());
```

```
                            traza_nueva.setSync(InfoBateria.getSync());
                            traza_nueva.setNetwork(InfoBateria.getNetwork());
                            traza_nueva.setGps(InfoBateria.getGps());
                            traza_nueva.setNlp(InfoBateria.getNlp());
                            traza_nueva.setScreen(InfoBateria.getScreen());

traza_nueva.setPorcentaje(InfoBateria.getBatLevel());
                            traza_nueva.setVoltage(InfoBateria.getVolt());
                            traza_nueva.setTemperatura(InfoBateria.getTemp());
                            traza_nueva.setPlugged(InfoBateria.getPlugged());

traza_nueva.setBluetooth(InfoBateria.getBluetooth());
                            traza_nueva.setCall(InfoBateria.getCall());
                            traza_nueva.setInstalled("");
                            traza_nueva.setUninstalled("");

traza_nueva.setBatSaver(InfoBateria.getBatSaver());
                            Trazas.actualizaTraza(id, traza_nueva);

                        }
    }
}
```

AlarmReceiver.java

```
package com.samuel.infobateria;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class AlarmReceiver extends BroadcastReceiver {
  public static final int REQUEST_CODE = 12345;

  // Triggered by the Alarm periodically (starts the service to run task)
  @Override
  public void onReceive(Context context, Intent intent) {
    Intent i = new Intent(context, AlarmService.class);
context.startService(i);
  }
}
```

PreferencesFromCode.java

```
package com.samuel.infobateria;

import android.content.Intent;
import android.content.res.TypedArray;
import android.net.Uri;
import android.os.Bundle;
import android.preference.CheckBoxPreference;
import android.preference.EditTextPreference;
import android.preference.ListPreference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceCategory;
import android.preference.PreferenceScreen;
import android.preference.SwitchPreference;
```

```
public class PreferencesFromCode extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setPreferenceScreen(createPreferenceHierarchy());
    }

    private PreferenceScreen createPreferenceHierarchy() {
        // Root
        PreferenceScreen root =
getPreferenceManager().createPreferenceScreen(this);

        // Dialog based preferences
        PreferenceCategory dialogBasedPrefCat = new PreferenceCategory(this);
        dialogBasedPrefCat.setTitle(R.string.monitor);
        root.addPreference(dialogBasedPrefCat);

        // List preference
        ListPreference listPref = new ListPreference(this);
        listPref.setEntries(R.array.entries_list_preference);
        listPref.setEntryValues(R.array.entryvalues_list_preference);
        listPref.setDialogTitle(R.string.dialog_title_list_preference);
        listPref.setKey("list_preference");
        listPref.setTitle(R.string.title_list_preference);
        listPref.setSummary(R.string.summary_list_preference);
        listPref.setValue("0");
        dialogBasedPrefCat.addPreference(listPref);
        return root;
    }
}
```

WindowChangeDtectingService.java

```
package com.samuel.infobateria;

import android.accessibilityservice.AccessibilityService;
import android.accessibilityservice.AccessibilityServiceInfo;
import android.content.ComponentName;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.os.Build;
import android.util.Log;
import android.view.accessibility.AccessibilityEvent;

public class WindowChangeDetectingService extends AccessibilityService {
      String currentActivity = "";

    @Override
    protected void onServiceConnected() {
        super.onServiceConnected();
        //Configure these here for compatibility with API 13 and below.
        AccessibilityServiceInfo config = new AccessibilityServiceInfo();
        config.eventTypes = AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED;
        config.feedbackType = AccessibilityServiceInfo.FEEDBACK_GENERIC;

        if (Build.VERSION.SDK_INT >= 16)
            //Just in case this helps
```

```
            config.flags =
AccessibilityServiceInfo.FLAG_INCLUDE_NOT_IMPORTANT_VIEWS;
        setServiceInfo(config);
    }

    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
        if (event.getEventType() ==
AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED) {
            ComponentName componentName = new ComponentName(
                event.getPackageName().toString(),
                event.getClassName().toString()
            );
            ActivityInfo activityInfo = tryGetActivity(componentName);
            boolean isActivity = activityInfo != null;
            if (isActivity){
             Log.i("CurrentActivity", componentName.flattenToShortString());
             if (!currentActivity.equals(componentName.getPackageName())){
                    RecibeCambios.app_package=componentName.getPackageName();
                    RecibeCambios.crearTraza("foreground");
             }
             currentActivity = componentName.getPackageName();
            }
        }
    }

    private ActivityInfo tryGetActivity(ComponentName componentName) {
        try {
            return getPackageManager().getActivityInfo(componentName, 0);
        } catch (PackageManager.NameNotFoundException e) {
            return null;
        }
    }

    @Override
    public void onInterrupt() {}
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.samuel.infobateria"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="21" />

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

```
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.REAL_GET_TASKS" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.READ_LOGS" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.samuel.infobateria.InfoBateria"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".PreferencesFromCode"
                android:label="@string/action_settings">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.SAMPLE_CODE"
/>
            </intent-filter>
        </activity>

             <receiver
                 android:name=".AlarmReceiver">
             </receiver>
             <service android:enabled="true" android:name=".AlarmService"
android:exported="false" />

             <receiver
                 android:name=".RecibeCambios">
             </receiver>
        <service android:enabled="true" android:name=".Servicio"
android:exported="false" />

        <service
        android:label="@string/accessibility_service_name"
        android:name=".WindowChangeDetectingService"
        android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
        <intent-filter>
            <action
android:name="android.accessibilityservice.AccessibilityService"/>
        </intent-filter>
        <meta-data
            android:name="android.accessibilityservice"
            android:resource="@xml/accessibilityservice"/>
    </service>

    </application>
```

```
</manifest>
```

Accessibilityservice.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- These options MUST be specified here in order for the events to be
received on first
 start in Android 4.1.1 -->
<accessibility-service
    xmlns:tools="http://schemas.android.com/tools"
    android:accessibilityEventTypes="typeWindowStateChanged"
    android:accessibilityFeedbackType="feedbackGeneric"
    android:accessibilityFlags="flagIncludeNotImportantViews"
    android:description="@string/accessibility_service_description"
    xmlns:android="http://schemas.android.com/apk/res/android"
    tools:ignore="UnusedAttribute"/>
```

8. Bibliografía

http://batteryuniversity.com/

http://es.wikipedia.org/wiki/Bateria_electrica

http://en.wikipedia.org/wiki/Lithium-ion_battery

http://developer.android.com/intl/es/index.html

http://andro4all.com/2015/06/doze-android-m-gestion-bateria

https://es.wikipedia.org/wiki/Amperio-hora

https://es.wikipedia.org/wiki/Analisis_de_la_regresion

https://es.wikipedia.org/wiki/Regresion_lineal

http://rootear.com/android/mejor-governor

9. Referencias bibliográficas

[1] - Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao and Lei Yang, *"Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones"*, 2010.

[2] - Alex Shye, Bejamin Scholbrock and Gokham Memik, *"Into the wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures"*, 2009.

[3] – Jemin Lee, Hyunwoo, Joe and Hyungshin Kim, *"Automated Power Model Generation Method for Smartphones"*, 2014.

10. Índice de figuras

11. Índice de tablas