Universidad Nacional del Centro de la Provincia De Buenos Aires
Facultad de Ciencias Exactas - Departamento de Computación y Sistemas
Ingeniería de Sistemas

# Sistema de recolección de datos de uso en dispositivos Android para la detección de oportunidades de ahorro de batería.

por
Pablo Vena, Leopoldo Rudenick

Director: Prof. Dr. Alejandro Zunino

Co-Director: Dr. Juan Manuel Rodríguez

Trabajo final de carrera presentada como requisito parcial
para optar por el título de
Ingeniero de Sistemas

Tandil, 08/25/15

## Resumen

Pequeño resumen.

## Agradecimientos

# Contents

# List of Figures

# List of Tables

# Lista de algoritmos

# Acrónimos

**GPS** Sistema de Posicionamiento Global

**SNR** Proporción de Señal por Ruido

Chapter 1

# Introducción

En los últimos años los avances tecnológicos han puesto en el centro de la escena a los dispositivos móviles, los cuales pasaron de ser terminales de propósito específico, como agendas electrónicas o teléfonos celulares, con capacidades limitadas de almacenamiento y procesamiento, a ser pequeñas computadoras de propósito general con grandes capacidades de procesamiento y almacenamiento. Una evidencia de esto es que actualmente los dispositivos móviles utilizan sistemas operativos similares a las computadoras personales, como por ejemplo Android (basado en Linux), Windows 8 o iOS (basado en MacOS). Por lo tanto, pueden ejecutar software similar al que hace unos años atrás solo se encontraba en computadoras personales, como suites de ofimática, juegos 3D y navegadores Web con soporte completo para HTML y JavaScript [14, 13, 24]. Debido a sus capacidades los dispositivos móviles son ampliamente utilizados. En el año 2011 ya existían más de dos mil millones de dispositivos móviles activos en el mundo. Adicionalmente, los usuarios de los países desarrollados generalmente poseían dos o más de este tipo de dispositivos [28].

A pesar de compartir características con las computadoras tradicionales, la utilización de dispositivos móviles introduce nuevos desafíos. Al aumentar la capacidad de procesamiento de los dispositivos móviles, se pudo empezar a ejecutar aplicaciones que antes no se hubiera podido, estas nuevas aplicaciones hicieron que los usuarios aumentaran el tiempo de uso que se le da a estos dispositivos, lo cual se traduce directamente en un mayor consumo de batería. Adicionalmente, a pesar de que las capacidades computacionales de los dispositivos móviles parecen crecer de forma exponencial, la capacidad de las baterías solo lo hizo de manera lineal [24]. Por estas razones, uno de los grandes desafíos a los que se enfrentan los fabricantes de dispositivos móviles, y desarrolladores de aplicaciones [2] y sistemas operativos para estos dispositivos, es encontrar alguna forma para prolongar la duración de la carga de la batería por el mayor tiempo posible.

En la Sección 1.1 se describe el contexto del problema que se analizará en este trabajo. En la Sección 1.2 se discuten las motivaciones del trabajo final propuesto. En la Sección 1.3 se plantea la solución propuesta y cuales son los objetivos que se desean alcanzar. Finalmente en la Sección 1.4se explica la estructura general del informe.

## 1.1 Marco teórico

Varias soluciones se han propuesto para mejorar el rendimiento de la CPU y para gestionar el disco y la pantalla de una manera inteligente para reducir el consumo de energía, pero las prestaciones de los dispositivos móviles de la actualidad no solo dependen de su capacidad computacional, sino de su capacidad para comunicarse con otros dispositivos. Para permitir la comunicación existen diferentes tecnologías con propósitos variados. A pesar de sus diferencias, la gran mayoría dependen del uso de ondas de radio para mantener la movilidad. Una de estas tecnologías es el WiFi utilizada para conectar los dispositivos a redes locales y acceder a Internet a través de estas. Se entiende por WiFi toda red local inalámbrica implementada mediante alguna de las variaciones del estándar IEEE 802.11. Otra tecnología que permite el acceso a Internet son las redes 3G, que a diferencia de las redes WiFi, poseen mayor alcance. Sin embargo, suelen proveer un acceso más lento y costoso. Por otro lado no permite acceso directo a otros dispositivos que se encuentren conectados a la misma red [36]. En contraste, la tecnología Bluetooth tiene como propósito la comunicación directa entre dispositivos cercanos para, por ejemplo, transferir archivos, implementar auriculares y teclados inalámbricos. Finalmente, otra tecnología que utiliza ondas de radio es el GPS, en este caso el objetivo es conectarse con satélites para determinar la ubicación del dispositivo.

La utilización de tecnologías basadas en ondas de radio produce una descarga de batería significativa en relación a otras características del dispositivo, por lo que una potencial estrategia para reducir el consumo de batería, es optimizar el uso de este tipo de tecnologías [26, 28]. Sin embargo, estas soluciones requieren cambios en la arquitectura de los dispositivos móviles, o de un nuevo hardware, lo que se traduce en un aumento de los costos, dejando de ser viables.

Por otro lado, técnicas de computación *offloading* han sido propuestas con el objetivo de migrar grandes volúmenes de procesamiento de estos dispositivos, con recursos limitados, a máquinas ricas en recursos. Esto evita tomar largos lapsos de tiempo de ejecución de aplicaciones en los dispositivos móviles, que da lugar a un alto consumo de energía, desgastándola en poco tiempo [3, 1]. Existen resultados que demuestran que la ejecución remota de la aplicación puede ahorrar energía de manera significativa [18, 23, 25].

Teniendo en cuenta lo enunciado anteriormente y en base al hecho de que Android es uno de los sistemas operativos más difundidos para dispositivos móviles [19], es que se

plantea utilizarlo como base de estudio para aprender sobre el consumo personalizado de batería, mediante el desarrollo de un software de recolección de información sobre el uso del teléfono, de manera eficiente. Los datos recolectados se utilizarán para construir modelos de uso de cada uno de los dispositivos, tratando de identificar patrones de comportamiento que conlleven al consumo innecesario de energía. Un ejemplo muy común es tener encendida la antena WiFi durante la mayor parte del día, aún en zonas sin conexión. El objetivo es, en un futuro, poder utilizar estos modelos en una aplicación que automáticamente active o desactive características del dispositivo, para reducir el consumo de batería.

## 1.2 Motivación

La batería de los móviles sigue siendo un rompecabezas que no ha sido resuelto por ningún fabricante. Si bien a medida que avanza la tecnología para su fabricación tienen mayor autonomía, esto se logra construyéndolas de mayor tamaño. Todo parece indicar que la única manera de lidiar con esta problemática, es enfocarse en el ahorro de la energía. La amplia gama de interfaces y sensores inalámbricos, y la creciente popularidad de las aplicaciones que demandan energía, como las redes sociales, pueden reducir la duración de la batería de los dispositivos móviles a tan solo algunas horas de funcionamiento. Los investigadores y fabricantes de sistemas operativos y hardware, han encontrado optimizaciones y técnicas innovadoras para extender la vida de la batería de los dispositivos móviles [11]. Sin embargo, las últimas investigaciones sobre baterías de litio indican claramente que la eficiencia energética debe lograrse en conjunto, tanto a nivel de hardware, como de software [32].

Se han estudiado diversas maneras de atacar el problema, como sistemas operativos de bajo consumo energético, la gestión eficiente de los recursos, el impacto de los patrones de interacción de los usuarios con los dispositivos móviles y sus aplicaciones, administración de las diferentes interfaces inalámbricas y sensores, y finalmente los beneficios de la integración de los dispositivos móviles con servicios en "la nube", esto último conocido como Cloud Computing [32].Otro ejemplo, en este caso 100% de la industria, es Project Volta de Google, implementado en Android 5 [11].

Actualmente existen numerosas aplicaciones destinadas a la gestión de recursos en los dispositivos móviles, con la principal función de tratar de reducir el consumo energético, pero todas requieren de la intervención del usuario. Ninguna toma decisiones automáticas y personalizadas basándose en el uso que cada usuario le da a su dispositivo, aprendiendo a su vez de esto [17].

Por otro lado, se encuentra el hecho de que durante el proceso de monitoreo y gestión de

recursos pueda generarse un gasto extra de energía. En este punto la solución implementada, lejos de mejorar la situación, la empeora y agrava.

Por esta razón, y debido a la falta de administración de parte de los usuarios, se plantea la posibilidad de desarrollar una aplicación encargada de optimizar el consumo energético de cada dispositivo. La forma de llevarlo a cabo implica el monitoreo de la utilización de las características más significativas de los dispositivos, con la restricción de mantener un bajo nivel de procesamiento, creando modelos de uso para cada dispositivo, que permitan identificar patrones de comportamiento estrictamente relacionados con un consumo significativo de batería.

## 1.3 Propuesta y objetivos

Dadas las condiciones tecnológicas explicadas anteriormente, se plantea una posible solución desde la perspectiva del software. El objetivo en este trabajo, consiste en desarrollar un sistema de monitoreo del uso del dispositivo móvil. Este monitoreo tendrá la restricción de consumir la menor cantidad de batería posible, con el fin de no perjudicar el desempeño del dispositivo móvil. Para lograr este cometido se utilizaran las estrategias de diseño de software adecuadas a la problemática descripta, como por ejemplo un procesamiento *offloading* para el análisis de los datos recolectados, delegando la mayor carga de procesamiento a un servidor dedicado y liberando al dispositivo de esta carga de trabajo, ya que podría consumir demasiada energía [32].

Una vez recolectada la información correspondiente a diferentes usuarios, se plantea la posibilidad de detectar, a través de técnicas especificas de inteligencia artificial, posibles situaciones en las cuales los usuarios estén malgastando su recurso energético. Dicha investigación se basará en el uso de conocidas técnicas de minería de datos [20], como reglas de asociación, clasificadores, y la aplicación de filtros a los datos recolectados. Esta investigación se centrará, en analizar el uso de las diversas características de los dispositivos, como son los módulos WiFi, Bluetooth, GPS o 3G, que son ampliamente utilizadas por los usuarios y las que consumen mayor cantidad de energía. Para el análisis se tendrán en cuenta datos del entorno, como ubicación aproximada, horarios, actividad realizada por el usuario (Caminar, Conducir, Estático), día de la semana, fines de semana, rangos de carga de baterías (alto, medio, bajo), debido a que proveen información extra acerca del comportamiento del usuario. El objetivo final es detectar patrones de uso del dispositivo móvil como: el usuario tiene el módulo WiFi encendido los lunes, pero se encuentra en un sector donde no existe señal; el usuario utiliza el modulo Bluetooth y luego lo deja habilitado cuando termino la conexión; el usuario se encuentra conectado a una red WiFi y también mantiene el módulo 3G habilitado, los fines de semana. Estos patrones serán analizados con el fin de detectar comportamientos que desperdician

energía, por ejemplo “el WiFi está activo en un lugar sin red WiFi”.

En conclusión, en este trabajo se espera definir un sistema de monitoreo para dispositivos móviles, de bajo costo, desde el punto de vista energético. Así mismo se espera determinar que métodos son eficientes para la detección de patrones de comportamiento, a partir de los datos obtenidos por el monitoreo. El objetivo final es proveer un análisis que permita, en un futuro, la implementación de un sistema de recomendación inteligente, capaz de detectar patrones de uso energéticamente ineficientes y tomar acciones correctivas para disminuir el consumo de batería, sin afectar las prestaciones del dispositivo móvil.

## 1.4 Estructura del documento

Este informe está organizado de la siguiente manera:

En el Capítulo 2 2 se describen los trabajos relacionados. En el Capítulo 3 3 se realiza la propuesta y se describen las herramientas utilizadas para el desarrollo del sistema, así como también las que fueron descartadas por alguna razón. Se describe la arquitectura, el modelo de datos y los componentes de la aplicación. En el Capítulo 4 4 se explica cómo se desarrolló el sistema y su estructura/arquitectura final, también se explican algunos cambios que se realizaron a la planificación original. En el Capítulo 5 5 se muestran las funcionalidades y facilidades con las que cuenta el sistema y se explica a grandes rasgos el modo de uso. Por último, en el Capítulo 6 6 se presentan las conclusiones del trabajo junto con algunas propuestas para realizar extensiones al presente trabajo. Para finalizar se detalla la bibliografía utilizada.

Chapter 2

# Trabajos relacionados

Con el crecimiento de las redes y los servicios, los dispositivos móviles se han convertido en herramientas casi indispensables en la vida de las personas. Sin embargo, el corto tiempo de duración de sus baterías es uno de los factores que mayor inconvenientes ocasiona o reduce su utilidad. Por este motivo es necesario encontrar mecanismos que ayuden a prolongar la vida útil de las baterías y uno de estos mecanismos es reduciendo el consumo. A continuación se explican diferentes propuestas para abordar esta problemática. En la sección 2.1 se mencionan trabajos orientados a predecir el consumo energético basado en patrones de uso. En la sección 2.2 se mencionan trabajos que evalúan el potencial ahorro energético al delegar tareas a la "nube". En la sección 2.3 se describen trabajos que proponen reducir el consumo energético realizando un manejo eficiente de los métodos de localización. Por último en la sección 2.4, se mencionan trabajos relacionados con la obtención de patrones comportamentales, en sistemas sensibles al contexto.

## 2.1 Predicción del consumo de batería basado en patrones de uso

Según lo expuesto en [16] la clave para minimizar el consumo de batería esta en poder predecir, con exactitud, el consumo en cada uno de los diferentes estados en los que pueda operar el dispositivo. Las predicciones acerca del tiempo de vida de la batería, generalmente, se basan en estimaciones hechas por los fabricantes de baterías y dispositivos móviles, según ciertos experimentos que han realizado y mediante cálculos aritméticos del consumo de energía. Sin embargo, estos métodos son limitados, porque no tienen en cuenta patrones de uso. La cantidad de llamadas de voz realizadas, videollamadas, uso de internet, etc, es algo propio de cada usuario. Si se asume que los patrones de uso de cada usuario son diferentes y que cada uno de estos patrones se mantiene con una base lógica similar en el tiempo, se puede predecir el consumo de batería para cada

uno de estos patrones.

En el método propuesto por [16] se asume lo siguiente:

- el consumo de batería de los dispositivos se ve afectado por llamadas de voz, videollamadas, uso de datos, mensajes de texto, LCD, aplicaciones, música y llamadas en espera
- se pueden definir posibles estados en términos de las diferentes operaciones funcionales del dispositivo
- el consumo promedio de batería de cada estado es diferente
- la probabilidad de duración de cada estado es diferente, debido a los diferentes patrones de uso de cada usuario
- si el patrón de uso de un usuario se hace repetitivo y consistente a lo largo del tiempo, se puede utilizar *Zipf's law* para predecir el consumo de batería basándose en este patrón de uso

Desde el momento en el que se enciende el dispositivo se comienza a almacenar información, dejando registro en un log de la batería y el tiempo consumido desde que se ingresa, hasta que se sale de un cierto estado. Luego esta serie de datos es utilizada para poder medir consumos promedio de batería y patrones de uso. Estos patrones de uso no solo representa el patrones de uso en el pasado, sino también la probabilidad de insumir tiempo en ese estado en un futuro. Se pueden utilizar modelos de Holt-Winters [12, 37] para predecir patrones de uso basándose en la serie de datos recolectados. Por lo tanto se podría calcular el tiempo remanente de vida de la batería.

Otra ejemplo es [38], que propone la utilización de modelos estadísticos en línea (on-line) para predecir el tiempo de vida de la batería, combinando datos históricos de disipación de energía, con mediciones de referencia pre-computadas fuera de línea. Al incorporar mediciones en línea dinámicamente, este enfoque es capaz de hacer predicciones que tengan en cuenta las diferencias de cargas de trabajo, el "efecto de recuperación" que las baterías experimentan cuando se descargan, y el efecto del ciclo de carga que afecta el desempeño de las baterías cuando estas se recargan constantemente.

Predecir el consumo energético en base a patrones de uso, es importante, por ejemplo, para poder informar a los usuarios cuanto tiempo de vida le queda a la batería de sus dispositivos, basándose en el uso que le está dando. De esta manera el usuario puede por ejemplo, ante un caso de poca vida útil restante, desactivar conexiones, Sistema de Posicionamiento Global (GPS), bajar el brillo de la pantalla, etc, para tratar de reducir el consumo energético y alargar el tiempo de vida útil de la batería de su dispositivo, hasta la próxima carga.

## 2.2 Potencial ahorro de energía delegando tareas a la *"nube"*

El procesamiento en la *"nube"*, también conocido como *Cloud Computing*, se trata de un paradigma en el cual los recursos computacionales como el procesamiento, memoria y almacenamiento no están físicamente presentes en el lugar en el que se encuentra el usuario, sino que es un proveedor de servicios el que administra y provee estos recursos, a los cuales los usuarios acceden vía Internet [18].

Muchos investigadores creen que esta estrategia de delegar el procesamiento a la *nube*, es un excelente candidato para ayudar a reducir el consumo de batería en los dispositivos móviles, pero se debe tener en consideración que esta tarea involucra una comunicación entre los dispositivos y la *nube*. Esta comunicación no es gratis en términos de consumo energético y ancho de banda. Por este motivo es que se debe considerar y analizar la relación costo-beneficio entre la cantidad de energía insumida por dicha comunicación, contra la cantidad de energía ahorrada gracias al procesamiento en la *nube* [4].

Partiendo de la suposición de que la energía insumida en la comunicación de datos hacia la *nube* y la descarga de los resultados, es menor que la utilizada para el procesamiento de dichas tareas con la CPU local, se pueden distinguir dos posibles objetivos de optimización:

1. Minimizar el consumo de batería del dispositivo para extender el tiempo operativo del mismo

2. La energía consumida por el servidor en la nube para el procesamiento de tareas, debe ser también incluida en el cálculo del consumo

La clave está en obtener modelos que permitan predecir el consumo energético dado un tipo de tecnología de transmisión, como WiFi, 3G, 2G, la Proporción de Señal por Ruido (SNR) y la cantidad de datos a ser transmitidos. En conjunto, la predicción del consumo energético requerido para procesar una tarea de forma local y remota, constituyen la piedra fundamental para una nueva generación aplicaciones eficientes en cuanto al consumo energético.

Una vez obtenidos los modelos de consumo energético, pueden ser utilizados para decidir cuando conviene que una tarea sea procesada de forma remota o local. [29].

### 2.2.1 Ventajas y desafíos de delegar tareas a la *nube*

A continuación se enumeran las ventajas que se pueden obtener al delegar tareas a la *nube* [8][23]

1. Extender la vida útil de la batería. La técnica de delegar tareas a la *nube*, tiene como propósito migrar tareas costosas en cuanto a procesamiento, desde los dispositivos móviles de recursos limitados, a máquinas con mayor poder de procesamiento, como son los servidores en la *nube*. Esto evita insumir tiempo de procesamiento en los dispositivos móviles, lo cual se traduce directamente en un mayor consumo energético, proporcional al tiempo que demore dicho procesamiento. Por otro lado, muchas aplicaciones móviles aprovechan la ventaja de la migración de tareas y procesamiento remoto. Por ejemplo, delegando tareas de procesamiento de imágenes a un servidor remoto, se puede reducir en un 41% el consumo energético de un dispositivo móvil.

2. Mejorar la capacidad de almacenamiento de datos y el poder de procesamiento. La capacidad de almacenamiento también es restringida en los dispositivos móviles. La idea de delegar tareas a la *nube* permite a los usuarios acceder a, o almacenar, grandes cantidades de datos en la *nube* a través de redes inalámbricas. También ayuda a reducir el costo de procesamiento de tareas intensivas desde el punto de vista computacional, que demandan mucho tiempo y energía cuando se ejecutan en dispositivos con recursos limitados.

3. Mejorar la confiabilidad. Almacenar o ejecutar aplicaciones en la nube es una manera efectiva de mejorar la confiabilidad, porque los datos y aplicaciones son almacenados y respaldados en varias computadoras. Esto reduce las probabilidades de pérdida de datos en los dispositivos móviles.

Así también se plantea la siguiente pregunta: £delegar tareas a la *nube* es la solución definitiva al problema energético en los dispositivos móviles? La respuesta es no, al menos no del todo. Mas allá de su tremendo potencial para lograr un ahorro energético, se deben considerar varias cuestiones de gran importancia [18][8][23]:

- Privacidad y seguridad en la *nube*. Los datos y las aplicaciones ya no se almacenan en los dispositivos de los usuarios, sino que son almacenados y administrados en la *nube*, por lo que la seguridad y privacidad depende de la administración que se provea en la *nube*. Un error o brecha de seguridad en la nube puede resultar en una brecha de privacidad, dejando expuestos los datos de los usuarios sin su consentimiento. Los proveedores de servicios en la *nube*, muchas veces, trabajan con productos y/o servicios de terceros y no hay garantías de cómo estos terceros protegen los datos. Por otro lado, como los lugares físicos de almacenamiento son en general desconocidos por los usuarios, resulta difícil determinar que leyes aplican para proteger la privacidad de los datos. Por este motivo, hay cierto tipo de información que no debe almacenarse en la nube, sin considerar previamente las implicancias que esto tiene, en cuanto a privacidad y seguridad. Una posible solución puede ser

encriptar los datos antes de enviarlos, pero se requeriría un procesamiento extra en los dispositivos móviles, lo que produciría un consumo de energía adicional. Por esta razón que en ciertos casos, delegar tareas a la nube puede no resultar beneficioso en cuanto al ahorro energético, siendo mas eficiente realizar dichas tareas en el dispositivo mismo.

- Confiabilidad. Otro potencial problema con delegar tareas a la nube es la confiabilidad. La dependencia de las redes inalámbricas, implica que en algunos casos cuando la conectividad es limitada, delegar tareas a la nube no sea posible y mucho menos eficiente en cuanto al consumo energético.

- Datos en tiempo real. Algunas aplicaciones requieren datos en tiempo real. Por ejemplo una aplicación de ajedrez, para la cual delegar la tarea de procesamiento de cada jugada a la nube, puede ayudar a ahorrar energía, pero esto depende del tamaño de los datos enviados y recibidos. Si se trata de un gran volumen de datos, puede que convenga particionarlos, de modo de reducir la energía consumida en la transmisión de los datos desde y hacia la nube.

Se puede concluir entonces, que delegar tareas a la nube puede potencialmente producir un ahorro de energía en los dispositivos móviles. Sin embargo, no todas las aplicaciones se comportan de manera eficiente, desde el punto de vista del consumo energético, cuando se migran a la nube. Los servicios provistos por la nube para dispositivos móviles, deberían ser significativamente diferentes de aquellos provistos para computadoras de escritorio, porque deberían ofrecer un ahorro energético. Los servicios deben considerar, además, la energía extra que se necesita para mantener la privacidad, seguridad, confiabilidad y para la comunicación de los datos, antes de delegar tareas a la nube.

## 2.3 Manejo inteligente de métodos de localización, para optimizar el consumo energético

Una característica importante de un dispositivo móvil moderno, es que puede obtener su propia posición. No solo para poder utilizarla de forma local, sino también para aplicaciones remotas que requieren un seguimiento del dispositivo, para actualizar localmente información relevante y adaptar el contexto de este. Aunque la mayoría de los teléfonos inteligentes incluyen un receptor GPS, su uso es en general restringido debido al alto consumo de batería.

Para ayudar a reducir el consumo energético en la obtención de la localización, la mayoría de las propuestas parten del hecho de que:

1. Las aplicaciones que utilizan localización no siempre requieren la mas alta precisión, como la que se puede lograr mediante un GPS. La precisión requerida varía a medida que el usuario se mueve, por lo que se pueden aprovechar estas variaciones con el fin de ahorrar energía.

2. Un teléfono dispone de múltiples modalidades para medir la ubicación aparte del GPS: triangulación WiFi, triangulación mediante la antena celular, en un radio de Bluetooth, sensores audiovisuales, entre otros. La disponibilidad y exactitud de estas modalidades varían cuando el usuario se mueve, y las modalidades mas adecuadas puede ser seleccionadas para satisfacer eficientemente las necesidades de localización con menores costos de energía.

Un ejemplo de esto es [10], una de las APIs que provee Google para sistema operativo Android, que permite obtener la localización del dispositivo optimizando el consumo de la batería, utilizando la información proveniente de todos los sensores de comunicación en el dispositivo, incluyendo redes WiFi, GPS y redes celulares.

Otro ejemplo es [15], que propone un servicio para obtener la localización privilegiando un bajo consumo energético, determinando automáticamente la precisión requerida para las aplicaciones basadas en búsquedas móviles. Mientras el usuario se mueve, tanto los requisitos de precisión como los errores en los sensores de ubicación cambian. Esta propuesta monitorea constantemente el gasto de energía para satisfacer los distintos requisitos de precisión, utilizando los sensores disponibles, con el fin de no sólo proporcionar un importante ahorro energético, sino también mejorar la precisión alcanzada (dado que se utilizan múltiples sensores).

También podemos mencionar el trabajo [22], donde se proponen estrategias de alto nivel para lograr un ahorro energético en la obtención de la localización, utilizando inteligentemente, no utilizando o intercambiando entre funciones y operaciones provistas por el sistema operativo, sin pensar en optimizaciones a nivel hardware. En particular se basa en evitar el uso del GPS siempre que sea posible, utilizando monitoreo de regiones (una técnica que utiliza funciones de transferencia del teléfono para redes inalámbricas y antenas celulares, para detectar un posible movimiento) o posicionamiento basado en redes WiFi o redes celulares cuando el dispositivo no se está moviendo, por ejemplo durante la noche mientras el usuario duerme o mientras está en la oficina trabajando.

De esta manera se trata de liberar a las aplicaciones de lidiar con los errores de localización y el manejo del consumo energético, haciendo un uso eficiente de la batería.

## 2.4 Identificación de patrones comportamentales en sistemas sensibles al contexto

Los sistemas para dispositivos móviles sensibles al contexto, han ganado mucha popularidad en los últimos años. Los dispositivos móviles están equipados con una variedad de sensores que los hace muy potentes desde un punto de vista computacional, permitiendo el procesamiento en tiempo real de los datos obtenidos por dichos sensores. Sin embargo, muchas aplicaciones sensibles al contexto se basan en información estática, con arquitecturas centralizadas y aquellas que son diseñadas para dispositivos móviles se focalizan principalmente en recursos limitados en términos de CPU y memoria, lo cual hoy en día no es un gran problema [6]. Si bien siempre se busca que las aplicaciones móviles sensibles al contexto sean lo más sencillas posibles, también es deseable que sean lo suficientemente potentes para no solo resolver tareas simples de identificación de contexto, sino también realizar procesamientos más complejos, como aprender sobre el comportamiento de los usuarios, identificando patrones de uso, para luego extraer conclusiones o predecir acciones. Aunque esto último puede resultar en sistemas con un alto consumo energético.

Un ejemplo de esto es la solución propuesta en [6] que tiene como principal objetivo minimizar el consumo energético del sistema, aprendiendo hábitos de uso de los diferentes sensores provistos. Incorpora la idea de un dispositivo móvil como una entidad autónoma sensible al contexto, equipada con una capa inteligente de intercambio (middleware) y otra capa de inferencia basada en el contexto. La capa inteligente de intercambio actúa como un intermediario entre la información del contexto y la capa de inferencia. Esta última tiene la capacidad de aprender patrones de uso de los sensores del dispositivo y ajustar los tiempos de muestreo para mejorar el consumo energético del sistema. La arquitectura general del sistema se puede describir de la siguiente manera (ver Figura 2.1):

1. capa de sensores: responsable de obtener información de los diferentes sensores del dispositivo (GPS, Acelerómetro, Bluetooth) y realizar un procesamiento inicial de esta información

2. capa de inferencia: responsable del razonamiento basado en el contexto y de administrar el conocimiento adquirido

3. capa inteligente de intercambio: actúa como un intermediario inteligente entre la capa de sensores y la capa de inferencia.

Por último utilizando un algoritmo de regresión logística se crea un modelo para aprender sobre el uso de un sensor, basándose en datos históricos. Esto permite ajustar los

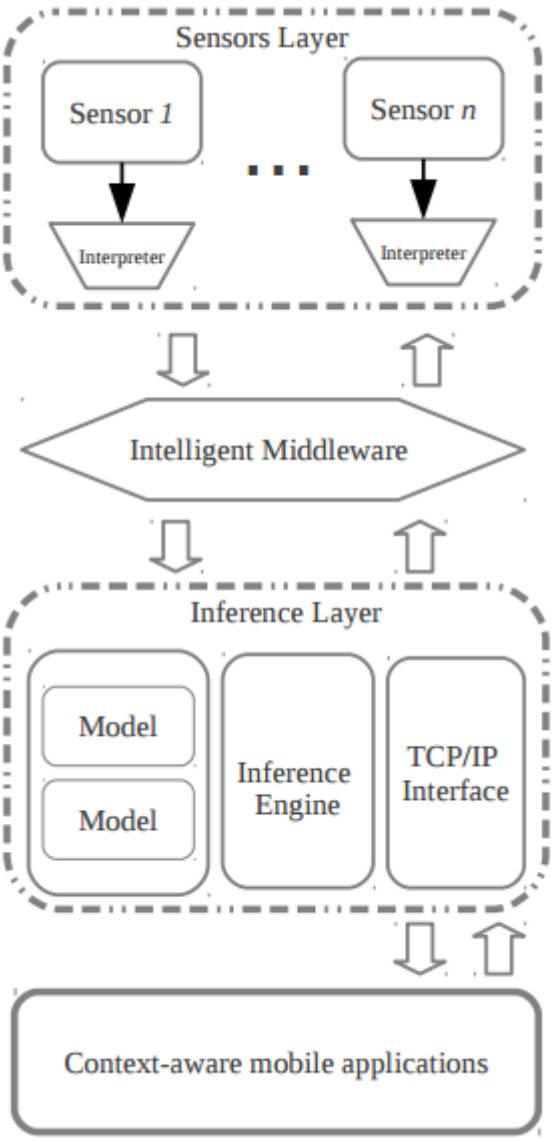



**Figure 2.1:** Arquitectura del sistema sensible al contexto

tiempos de muestreo de los sensores, de acuerdo a la probabilidad de uso. Las pruebas en dispositivos reales muestran que se puede ganar un 50% de ahorro energético usando este algoritmo.

Otra ejemplo en la detección de patrones en sistemas sensibles al contexto es [31], pero sin considerar el ahorro energético como principal objetivo. Esta propuesta se basa en identificar patrones comportamentales para cada usuario, para tratar de predecir a donde irá el usuario y que aplicación va a usar en los próximos diez minutos, haciendo uso de la información contextual obtenida desde los diversos sensores que posee un teléfono inteligente, tales como localización, registros de llamadas, proximidad con otros dispositivos, hora y aplicaciones utilizadas. Utiliza un conjunto de datos reales obtenidos del uso que diversos usuarios dieron a sus dispositivos, proveyendo información tal como uso de GPS, Bluetooth, puntos de acceso WiFi, acelerómetro, aplicaciones utilizadas y registros de llamadas. Luego se realiza un pre-procesamiento de los datos de localización obtenidos mediante el GPS y los puntos de acceso WiFi, para obtener regiones en las que los usuarios se mantienen por al menos diez minutos, con un radio de cien metros para hacer frente a la existencia de ruido en la obtención de los datos de localización y con ayuda de los usuarios se le asignan etiquetas para identificar cada una de las regiones. Se toma un máximo de ocho regiones por usuario. Cinco de estas regiones corresponden a las mas visitadas y las tres restantes son elegidas al azar de entre el 10% de las menos frecuentadas. Dado que los usuarios usan un número limitado de aplicaciones de su dispositivo, solo se consideran para las predicciones aquellas más relevantes, utilizadas al menos una vez a la semana. Luego, utilizando modelos estadísticos se puede predecir hacia donde se va a dirigir un usuario y que aplicación va a utilizar, basándose en el estado actual y la probabilidad de moverse hacia otro estado.

En la propuesta [5] se presenta un sistema capaz de integrar información proveniente de diversas redes sociales y sensores del dispositivo móvil, para generar datos de salida sensibles al contexto no solo para individuos sino también para grupos. La arquitectura del sistema está organizada en múltiples etapas y clases de datos recolectados de los diversos sensores que posee el dispositivo, además de datos obtenidos de redes sociales como Facebook, Twitter, LinkedIn, MySpace, etc. En una primera etapa estos datos son organizados, procesados y filtrados. En la segunda etapa, se utiliza un conjunto de clasificadores que operan sobre los datos, para tratar de extraer información contextual relevante. Por ejemplo, un clasificador puede ser utilizado para encontrar el estilo de música preferido por un individuo, analizando sus patrones comportamentales, mientras que otro clasificador puede buscar como medir una característica asociada a un grupo de individuos, como puede ser que tan cercanos son unos con otros en relación a la amistad. Basándose en estas inferencias a alto nivel, como también en los datos sin procesar, en una tercer etapa un motor de recomendaciones utiliza técnicas de *data mining*, para generar como resultado acciones sensibles al contexto. Estas son generadas tanto para

individuos como para grupos, dependiendo de la circunstancia.

Otra propuesta muy interesante es la descripta en[34][33][35]. Se trata de una aplicación para dispositivos Android, la cual recolecta estadísticas de uso en segundo plano mientras se utiliza el dispositivo. Toda información sensible es ofuscada, para mantener la privacidad de los usuarios. Periódicamente los datos almacenados son enviados a un servidor, donde se almacenan junto con los datos provenientes de otros usuarios y se sacan conclusiones a partir de los patrones que emerjan. Algunos de los datos recolectados son:

- cuando el dispositivo se enciende o carga
- cuando se realizan llamadas o se envían mensajes de texto
- qué aplicaciones son utilizadas
- dispositivos cercanos vía WiFi y Bluetooth
- localización basada en la red

El sistema completo se puede dividir conceptualmente en seis partes:

1. *Medición*. Los datos son recolectados en base a mediciones que va realizando la aplicación, la cual corre en dispositivos con sistema operativo Android. Los datos son almacenados como pares clave-valor, ambos son texto plano, y pueden contener datos de longitud variable. Un simple punto de datos puede contener una pequeña cantidad de información, por ejemplo acerca del nivel de señal de la antena WiFi, así como gran cantidad de información, por ejemplo todas las fechas completas de todas las imágenes almacenadas en la galería de fotos del dispositivo. Por otro lado, las claves están organizadas en una estructura jerárquica, la cual permite encontrar coincidencias entre prefijos durante la fase de análisis.

2. *Procesamiento en el dispositivo*. Con el fin de proveer de información y estadísticas de alto nivel al usuario, acerca del uso de su dispositivo, la aplicación procesa datos de forma local. Estas estadísticas incluyen la duración de llamadas telefónicas, números de mensajes de textos enviados y recibidos, nivel histórico de batería y muchos mas. En esta etapa también se remueven datos personales que puedan llevar a identificar al usuario y el resto de la información sensible se ofusca.

3. Recolección de datos. La aplicación está diseñada para almacenar datos hasta que estos puedan ser enviados al servidor y se confirme su recepción. Si se llega a un cierto máximo de datos recolectados, se suspende la recolección hasta que sea posible su envío.

4. *Almacenamiento*. La tarea principal del proceso del servidor es recibir y registrar, de forma fiable, los datos recibidos desde los dispositivos. Lotes válidos de datos se añaden a un archivo plano para el dispositivo en cuestión. Los datos duplicados producidos por transmisiones repetidas desde el cliente se descartan en este punto . Cuando el archivo anterior de cada dispositivo llega a 10MB, se crea uno nuevo. Los archivos antiguos se comprimen y se trasladan a otra ubicación para su almacenamiento permanente.

5. *Software de análisis en tiempo real del lado del Servidor*. Se provee información sobre el número actual y total de participantes a todos los usuarios y el sitio web muestra un mapa dinámico de todo el mundo, donde se pueden visualizar las subidas de archivos a medida que ocurren. Estas estadísticas se calculan como filtros simples que se ejecutan a medida que los datos se reciben.

6. *Procesamiento fuera de línea del lado del servidor*. Durante esta fase se procesan todas los archivos recibidos de un dispositivo dado en orden y se generan las tuplas de datos a un grafo de estados de procesamiento. Cada uno de estos grafos expone su estado a otros para que lo aprovechan. Por ejemplo, el grafo relacionado a estados de pantalla observa eventos como "pantalla encendida" y "pantalla apagada", con el fin de reportar el estado de la pantalla del dispositivo en cualquier punto del tiempo a otros grafos que lo tienen como dependencia, por ejemplo cuando se miden los datos transferidos mientras la pantalla estaba encendida.

Además, los datos recolectados de mas de 17 mil dispositivos de todo el mundo, proveen una inmensa base de datos la cual está disponible de forma gratuita para los investigadores académicos e industriales. La única restricción es prohibir identificar directamente a cualquier individuo dentro del conjunto de datos. Como ejemplo de la utilización de estos datos se puede mencionar el trabajo [7], donde se presenta un estudio de medición y modelización del impacto de la intensidad de la señal inalámbrica, 3G y WiFi, en el consumo de energía de un teléfono inteligente. A partir de análisis de los datos se demuestra que los usuarios de teléfonos inteligentes experimentan, rutinariamente, variaciones significativas de señal 3G y WiFi en su uso diario. Además, que el total de usuarios estudiados realizan en promedio 43% y el 21% de sus transferencias de datos en primer plano con una señal pobre de 3G y WiFi, respectivamente. Esto motiva la necesidad de diagnosticar los servicios de redes celulares, para mejorar la experiencia del usuario. Adicionalmente se presentan experimentos realizados, de forma controlada, para cuantificar el impacto energético que tiene la mala intensidad de señal en la transferencia de datos y se señalan implicaciones en el diseño de aplicaciones energéticamente eficientes. Por último, en base al modelo energético obtenido, se muestra que simplemente retrasando el tráfico en segundo plano, se puede reducir el consumo total

de energía en la comunicación de datos hasta en un 23,7% y 21,5% utilizando WiFi y 3G, respectivamente, suponiendo un retraso máximo de 12 horas.

En el siguiente capítulo se describe el sistema propuesto y se detallan los diferentes aspectos tenidos en cuenta para su desarrollo, así como las restricciones de calidad en las cuales se basa el diseño y estrategias generales utilizadas en cada componente del mismo.

Chapter 3

# Propuesta

Como se menciona en capítulos anteriores, el consumo energético, hoy en día, es uno de los problemas más importantes a enfrentar en los dispositivos móviles. Por esta razón nuestro trabajo tiene el objetivo de optimizar el rendimiento de la batería. Actualmente, con el auge de la información y la comunicación se encuentran disponibles una gran cantidad de datos y funciones que pueden ser muy útiles a la hora de estudiar el comportamiento de los diferentes usuarios, ya sea que estén utilizando, o no, su dispositivos. Información como la localización, o la red inalámbrica a la cual un aparato se encuentra conectado, permiten estudiar en que contextos se malgasta el potencial energético. Eventos relacionados con cambios de estados en el dispositivo, nos brindan información acerca de cuáles son los horarios pico en los cuales se utiliza un aparato y cuál es la posible relación entre estas variaciones de estado.

En este trabajo se propone LoguinUse, una aplicación destinada a recopilar información relacionada al estado y el contexto en el cual un dispositivo es utilizado, para luego, en base a estos datos, realizar una investigación con el propósito de encontrar patrones comportamentales en los cuales se detecte un mal uso de la energía. LoguinUse es una aplicación para dispositivos móviles con sistema operativo Android. La plataforma se eligió debido a que cuenta con una gran popularidad, por lo que cubre un gran número de usuarios. La aplicación recopilará diversa información sobre el uso que los usuarios le dan a sus dispositivos móviles, por ejemplo lugar del uso, conexiones WiFi, conexiones Bluetooth, etc. La información recopilada por la aplicación será comprimida y enviada a un servidor dedicado, desarrollado sobre la plataforma .NET que provee un WebService SOAP y una posee una base de datos relacional SQL Server. Si bien el objetivo final sería que el servidor procese los datos, como prueba de concepto se piensa utilizar la herramienta WEKA para probar distintas técnicas de aprendizaje de maquina para detectar patrones en los datos recopilados. Para realizar las pruebas de concepto, se desarrollo una aplicación de escritorio sobre .NET con la capacidad de filtrar y exportar la informa-

ción desde la base de datos a archivos en formato ARFF para su posterior estudio con la herramienta WEKA.

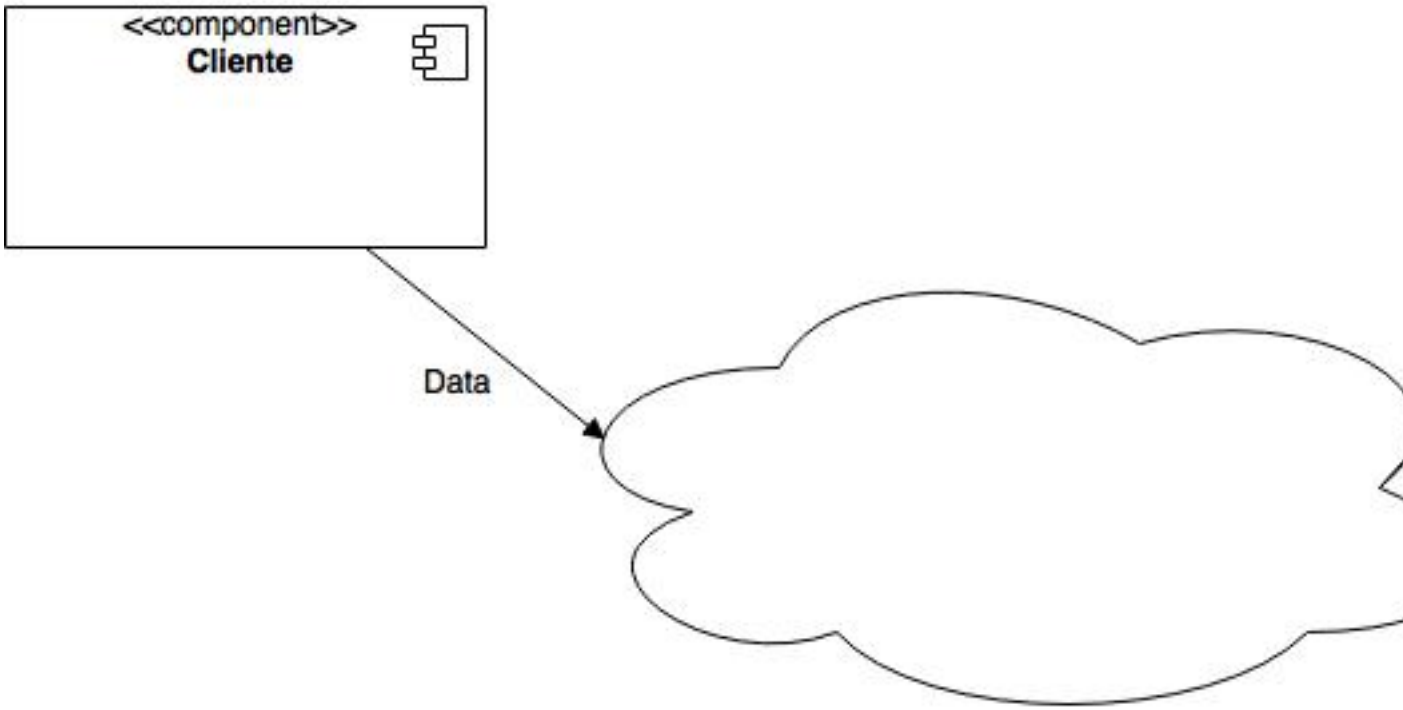
En este capítulo se detallaran los diferentes aspectos tenidos en cuenta para el desarrollo del sistema, así como las restricciones de calidad en las cuales se base el diseño y estrategias generales utilizadas en cada componente del sistema.

## 3.1 Restricciones de calidad

Como primera medida se desea que la ejecución de la aplicación no interfiera en las medidas obtenidas relacionadas al rendimiento de la batería, por lo tanto se pone especial énfasis en un rendimiento óptimo a la hora de obtener y almacenar la información del dispositivo. Por este motivo es que se debe disminuir lo máximo posible el procesamiento realizado por el dispositivo durante la captura de los datos, y reducir, además, el espacio necesario para almacenar dichos datos. También, teniendo en cuenta el rápido avance tecnológico, en lo que a dispositivos móviles respecta, y las cada vez más numerosas características de estos, se busca un bajo nivel de cohesividad entre componentes y gran flexibilidad para cambios futuros, de manera tal que la aplicación pueda crecer sin mayores inconvenientes en caso de necesitar, por ejemplo, obtener información adicional referente a una nueva funcionalidad del dispositivo.

## 3.2 Estrategia general

Dada la problemática planteada, se toman en consideración algunas restricciones de calidad con respecto al diseño del sistema que apuntan a un buen desempeño, sobre todo de la aplicación móvil, pero sin desatender la aplicación servidor. Debido a esto se propone un sistema cliente-servidor en el cual la mayor carga de procesamiento se encuentra del lado del servidor. De esta manera se limita a la aplicación cliente a la obtención de datos del usuario y evitando toda lógica adicional de procesamiento de datos. El servidor será capaz de almacenar y procesar todos los datos obtenidos.




## 3.3 Estrategias aplicación móvil

Las estrategias utilizadas en la aplicación móvil están orientadas a minimizar el tiempo de procesamiento y almacenamiento con el objetivo de minimizar el consumo de energía y evitar que el registro de la información afecte indirectamente los datos obtenidos ya que de otra manera la aplicación podría ocasionar variaciones en el consumo de la batería haciendo que el monitoreo de los datos sea erróneo.

Teniendo en cuenta los costos de procesamiento utilizados por Android, a la hora de crear objetos en memoria dentro de una aplicación, se optó por diseñar una estructura en memoria en la cual se centraliza la información relevante del estado actual del dispositivo y a la que se puede acceder desde los diferentes componentes de la aplicación. Esta estructura se actualiza automáticamente en cada evento del sistema por lo tanto siempre se podrá encontrar en ella el estado actual del sistema. Si bien esta estrategia requiere de un mayor uso de memoria RAM este no es un recurso prioritario, y al simplificar considerablemente el acceso a los estados almacenados, evita una lógica adicional a la hora de buscar información relacionada al estado del dispositivo ahorrando tiempo de

procesamiento.

Como método de almacenamiento se decidió persistir la información en archivos de texto plano, evitando el procesamiento de estructuras mas complejas como XML, que si bien son más deseables a la hora de trabajar con ellas, conllevan un mayor costo de procesamiento para leer y escribir en ellas, comparados con el necesario para leer y escribir en un archivo plano. Además utilizan un mayor espacio de almacenamiento. Aunque esta decisión implica un desarrollo más complejo del lado del servidor, el cual debe ser capaz de interpretar los datos recibidos, el foco está puesto en minimizar el procesamiento en la aplicación móvil y no en el servidor.

## 3.4 Estrategia servidor

Se decidió utilizar SOAP WebService para realizar la comunicación de datos simplemente por una cuestión de practicidad, debido a que la plataforma .NET provee una interfaz amigable a la hora de crear este tipo de servicios. El método empleado para la creación del servicio SOAP es *code-first* [30]. Este permite la definición de un servicio a partir del código, por lo que no es necesario tener ningún conocimiento sobre WSDL [9]. Se desarrollan los métodos de la interfase del servicio para luego generar el servicio, utilizando las herramientas provistas por el .NET.

El servidor soporta la mayor carga de procesamiento del sistema, ya que se deben leer e interpretar los archivos enviados por cada usuario, para luego almacenarlos de manera organizada.

A su vez el WebService proveerá los métodos capaces de exportar la información desde la base de datos, a archivos en formato ARFF para su posterior estudio.

## 3.5 Estrategia Base de Datos

Una vez puesto en marcha el sistema, este se encargara de recolectar una gran cantidad de información relacionada a varios usuarios. Esta información consistirá en un elevado volumen da datos, los cuales deben se almacenados de una manera ordenada y de fácil acceso, es por este motivo que se decidió utilizar una base de datos relacional [27].

La estrategia a la hora de diseñar una estructura de bases de datos, estará orientada a minimizar el espacio físico utilizado, normalizando ([21]) las tablas de la manera más conveniente en términos de practicidad, debido a que el sistema almacenara una abundante cantidad de información diaria, esta estrategia nos evita un crecimiento exponencial del espacio de almacenamiento.

Dado que el volumen de información al que se tiene acceso mediante el uso de los dispositivos es muy grande, para este trabajo, se estudiara un subconjunto reducido de los datos disponibles. Con el objetivo de encontrar patrones comportamentales, los cuales definen de que manera es utilizado el dispositivo por cada usuario, se almacenara en la base de datos, información relacionada con el dispositivo, el usuario y el contexto en el cual el aparato es utilizado.

Si bien, el estudio se realizara con una serie de datos específicos seleccionados cuidadosamente para este trabajo, el diseño de la base de datos soportara la incorporación de nuevos valores, permitiendo la flexibilidad de, en un futuro, sumar información que se considere relevante. Por ejemplo, llegado el caso de que un dispositivo cuente con un termómetro incorporado, la información de la temperatura del ambiente se podrá incluir en la base de datos sin modificar su estructura.

Con el objetivo de minimizar el espacio utilizado por la base la datos relacional, se decidió compartir datos entre usuarios, de esta manera se evitara la generación duplicada de información, para que ante un caso, por ejemplo, de que dos usuarios utilicen la misma red Wifi, la información de esta se almacene una única vez y sera referenciada por cada usuario particular.

Chapter 4

# Implementación

En este capítulo se explicara de manera detallada, cuáles fueron las decisiones tomadas a la hora de llevar a cabo las estrategias explicadas en el capítulo anterior. En particular se pondrá énfasis en las plataformas utilizadas y las decisiones correspondientes al diseño de software en los diferentes componentes del sistema. Adicionalmente se explicaran algunos conceptos del sistema para una mejor comprensión.

## 4.1 Aplicación Android

Haciendo uso de la gran variedad de funciones del sistema Android (Intents, IntentFilter, Services, Receivers, etc), esta aplicación, será capaz de obtener una gran cantidad de datos, relacionados con el estado del dispositivo y el contexto en que es utilizado, para luego enviar estos datos a través de la red, hacia un servidor en el cual serán almacenados. A continuación se explicaran los componentes de la aplicación así como también los métodos utilizados y el diseño del sistema.

### 4.1.1 Conceptos básicos

Dentro del sistema se hace uso de tres conceptos básicos, a la hora de almacenar los diferentes tipos de indicadores de actividades del usuario. Estos conceptos simplifican la lectura de los datos recolectados y permiten cierta flexibilidad en caso de que sea necesario agregar información. A su vez permiten minimizar el espacio de almacenamiento requerido y recursos de CPU al momento de analizar los datos.

El formato utilizado para almacenar cada línea de log es el siguiente:

[HORA]&[TIPO]&[PROP1::VALOR1 | PROP2::VALOR2 | ... | PROP_N::VALOR_N]

A continuación se explicaran dichos conceptos básicos, así como también se mostrarán ejemplos prácticos para su mejor comprensión.

**Tipo**

Para una buena organización de la información obtenida es necesario definir una estructura simple y ordenada. De esta manera surge la noción de tipos de log. Cada tipo de log engloba un conjunto de datos propios de la característica que monitorea. Algunos ejemplos de tipos de log pueden ser: WIFI, BLUETHOOT, BATTERY, ACTIVITY, CONNECTION, LOCATION, etc. Cada uno de estos se corresponde con una característica específica del dispositivo y contiene un número variable de propiedades relacionadas a esta. Cada línea generada en el log estará encabezada por el horario y el tipo al cual pertenece el cambio de estado detectado.

**Propiedad**

Una vez definidos los tipos de logs, es necesario definir cuál es el conjunto de datos de interés que se quieren almacenar. El conjunto de propiedades definidas para cada tipo de log, identifica cuales son los valores más relevantes, relacionados con un evento específico dentro de la aplicación, que deben ser almacenados. Un ejemplo de propiedades, dentro del tipo de log WIFI, son la dirección ip y el nombre de la red, definidos como *IP* y *NAME* respectivamente.

**Value**

Haciendo uso de los conceptos explicados hasta el momento, podemos definir la estructura base para un evento específico. Un evento monitoreado por la aplicación, almacenara un conjunto de datos definidos por el tipo del log al que pertenece y el conjunto de propiedades que lo conforman. Definimos Valor como el valor almacenado en un momento dado, para una Propiedad específica, incluida en el tipo de log monitoreado. Cabe destacar que una Propiedad pude tener únicamente un Valor en un momento dado.

[09:15:11]&[WIFI]&[STATE::1 | SSID::EI200P | MAC::C0:65:99:A7:BC:E0 | IACCESS::1]

Como se puede observar en el ejemplo, adicionalmente a la estructura explicada con anterioridad, se almacena el momento exacto en el cual fue capturado el evento, con el fin de ordenar cronológicamente la información de ser necesario.

### 4.1.2 Componentes

La aplicación móvil está compuesta por tres grupos de componentes, los cuales son los encargados de realizar las funcionalidades más importantes de la misma. Tanto la funcionalidad como el diseño de estos componentes utilizan mecanismos provistos por la plataforma Android como *Service*, *BroadcastReceiver* e *Intents*.

• *Intent*: Se utiliza para comunicar componentes en una aplicación Android

• *Intent-Filter*: nos permite filtrar los diferentes *Intents* del sistema y realizar diferentes tareas dependiendo del tipo de Intent esperado.

• *Activity*: Es un componente el cual posee una interface gráfica con la cual la aplicación interactúa con el usuario.

• *BroadcastReceiver*: responde a la emisión de un Intent particular haciendo uso de *Intent-Filters*. Estos componentes permanecen inactivos y son ejecutado al momento de recibir un Intent esperado.

• *Service*: es un componente que corre, en todo momento, en segundo plano sin interacción con el usuario y sin una interface gráfica, el cual se comunica con *Activities* atreves de Intents

#### 4.1.2.1 Receivers

Heredando de la clase abstracta *GeneralLoggingReceiver*, la cual provee la funcionalidad de escritura en el log, estos componentes, permiten el monitoreo de un determinado Tipo de Log. Cada receiver registra un único Tipo de Log y durante su ejecución, al momento de detectar el evento correspondiente, obtiene un listado de pares Propiedad-Valor del sistema, que luego serán almacenados en un archivo. Dado que los dispositivos incluyen nuevas características con el rápido avance de la tecnología, esta herencia permite la inclusión de nuevos Tipo de Log y su correspondiente Receiver responsable del monitoreo.

Los Receivers utilizan el comportamiento provisto por las clases *BroadcastReceiver* y los *Intent-Filter*, propias del sistema operativo Android. Un *Receiver* es activado por llamadas a través de *Intents*. Para lograr esto se utilizan *Intent-Filters* con los cuales se filtran las llamadas del sistema, para solo atender a las llamadas que correspondan y/o sean de interés.

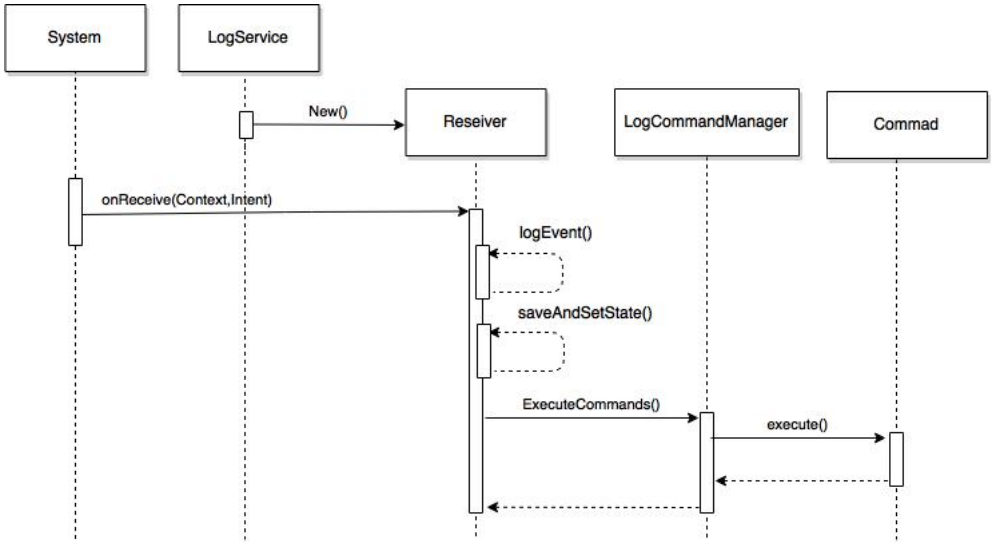



**Figure 4.1:** Diagrama de secuencia Log-Event se puede observar el mecanismo de monitoreo del sistema. Nótese en la imagen los componentes Receiver y Command se refieran a instancias específicas de las clases GeneralLogReceiver y LogCommand respectivamente.

El sistema Android lanza eventos, en forma de *Intents*, cuando ocurre un acontecimiento importante, como un cambio de estado en el dispositivo. Cada *Receiver* es activado solo por los *Intents* definidos en su *Intent-Filter*, de esta manera, por ejemplo, en caso de un cambio de estado de la Batería, solo se ejecutara el *Receiver* encargado de monitorear la Batería. Este mecanismo es muy eficiente ya que nos permite ejecutar tareas atómicas vinculadas únicamente a un cambio de estado específico en el dispositivo.4.14.2

Cada *Intent* utilizado por los *Receivers* debe ser incluido en el archivo *AndriodManifest.xml* de la aplicación. De esta manera al momento de la instalación se obtienen los permisos de sistema operativo para acceder la información específica de cada característica que se desea monitorear.

#### 4.1.2.2 Servicio

Este componente es el encargado de monitorear el dispositivo constantemente, haciendo uso de los *Receivers* (ver4.1.2.1) del sistema. Este componente se activa en el momento en que se enciende el aparato, durante su proceso de creación el servicio se encarga de inicializar la totalidad de los Receivers que serán utilizados en el monitoreo. El servicio es el componente esencial de la aplicación, ya que conserva activos en memoria los diferentes objetos involucrados en el monitoreo y captura de la información del dispositivo.

En la plataforma Android, un servicio puede ser ejecutado en modo Foreground o en modo Background.

- Foreground: El sistema reconoce al servicio como un proceso del cual el usuario

no es consciente de manera activa, de esta manera en caso de necesitar recursos, el sistema intentara no eliminar este proceso.

- Background: Es el modo de ejecución por defecto en la plataforma Android. En este caso el sistema considerara que el proceso puede ser eliminado sin problemas en caso de necesitar recursos.

Ya sea en modo Foreground así como en Background, el sistema puede eliminar el proceso de ser necesario, por ejemplo en caso de necesitar memoria RAM para ejecutar un proceso con mas prioridad.

En esta aplicación se decidió utilizar el modo Foreground ya que el monitoreo del dispositivo no es un proceso del cual el usuario deba estar pendiente, y adicionalmente le da una mayor prioridad al servicio con el objetivo de mantenerse la aplicación en funcionamiento el mayor tiempo posible. En caso de que el sistema decida eliminar el proceso, durante su destrucción, el servicio intentara ejecutarse a sí mismo. Este mecanismo permite que la aplicación se inicialice nuevamente, permitiendo el logueo ininterrumpido de datos. En caso de que esta estrategia no funcione y el servicio sea eliminado, se deberá inicializar manualmente. Durante la ejecución de este componente, se muestra en la barra de estado del dispositivo una notificación de que dicho servicio se encuentra corriendo.

#### 4.1.2.3 Commands y CommandManager

Dentro de la aplicación existe el componente *CommandManager*, en el cual se mantiene un registro del estado actual del dispositivo. Esto permite saber en cualquier momento cual es el último estado en el que se encontraba el sistema. Este estado está representado por un vector de valores en un objeto del tipo *HashTable*. Cuando un nuevo evento es detectado por los *Receivers* del sistema, los datos son persistidos en el archivo de log, actualizando además dicho vector de estados.

Este componente también nos permite disminuir significativamente el espacio requerido por los archivos de logs. Los datos registrados en el log en cada evento específico, están formados por un sub-vector del vector de estados. De esta manera, solo se permite el logueo de la información si el nuevo sub-vector difiere del sub-vector correspondiente al último estado almacenado.4.1

El listado de código 4.1 muestra que solo en el caso de que la variable *"change"* sea verdadera se registra la información. Este mecanismo provee flexibilidad para agregar nuevos tipos de logs e incrementar el número de *Propiedades* dentro de un tipo de log, sin alterar el funcionamiento descripto.

```
Hashtable<String, String> properties = l.getProperties();
Enumeration<String> enumKey = properties.keys();
While (enumKey.hasMoreElements() {
    property = enumKey.nextElement;
    value = properties.get (property);
    change |= LogCommandManager.getInstance().
        newState(l.getType(),property,value);
}
If (change)
    LogSave.getInstance().saveData(l);
```

**Algoritmo 4.1:** Ejemplo Logueo

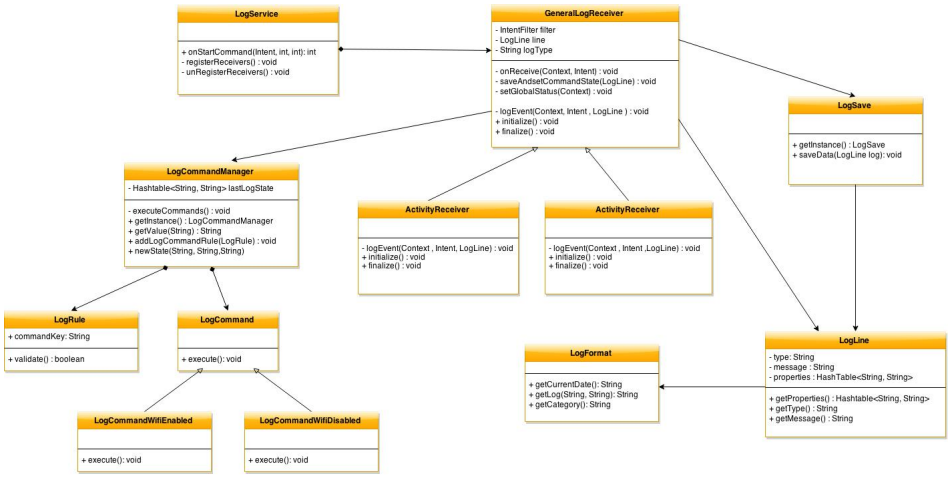



**Figure 4.2:** Diagrama de Clases: Aplicación Android

Dentro del *CommandManager* también existen *Commands*. Cada *Command* hereda la interfase y funcionalidad de un *Command* abstracto y es capaz de realizar una tarea específica, mediante el uso de su método *execute*. Estos objetos son utilizados para realizar tareas programadas dentro de la aplicación, como por ejemplo el envió de datos al servidor.4.24.1

Es posible configurar cada *Command* mediante el uso de las reglas, *Rules*. Cada *Rule* contiene un sub-vector compuesto por pares Propiedad-Valor y una clave que indica el tipo de *Command* a ejecutar, de esta manera, cuando el sub-vector configurado coincide con el estado actual del dispositivo, el *CommandManager* ejecuta el *Command* correspondiente.

#### 4.1.2.4 Comunicación de Datos

Toda la información generada por la aplicación, será almacenada en archivos de texto plano, siguiendo el formato especifico explicado en la sección4.1.1. Con el objetivo de almacenar la información de manera organizada, se decidió generar archivo por día dentro de un directorio creado por la aplicación. El nombre de cada archivo contiene la fecha en la cual fue generado y la información correspondiente. De esta manera también logramos reducir el espacio requerido, ya que no es necesario registrar la fecha en cada línea del archivo.

Con el objetivo de no producir consumo de energía de la batería del dispositivo a la hora de enviar los datos recolectados al servidor, la aplicación, por defecto, envía los datos solo si el aparato se encuentra cargando la batería. Adicionalmente, para evitar incurrir en gastos por el uso de redes de telefonía celular, la aplicación también verifica antes de enviar los datos que se cuente con conexión a Internet mediante una red WIFI. Otra ventaja de las redes WIFI sobre las redes 3G es que, en general, proveen una velocidad de acceso a Internet superior

Para evitar fragmentar los datos recolectados, se decidió no enviar el archivo correspondiente al día en curso, debido a que es probable que continúe siendo utilizado. De esta manera en el momento de enviar la información al servidor se obtendrán todos los archivos pendientes exceptuando el correspondiente al día actual. Una vez que se obtiene la lista de archivos a enviar, se genera un paquete comprimido y este es enviado al servidor.

## 4.2 WebService

Para realizar la comunicación de datos entre el dispositivo móvil y el servidor se utiliza un WebService SOAP, el cual debido a su protocolo estandarizado nos permite comunicar sin problemas diferentes tecnologías, como son en este caso, un servidor desarrollado sobre la plataforma .Net, con una aplicación Android. Este WebService provee los métodos para almacenar los datos recolectados por los usuarios en una base de datos relacional, que corre sobre un motor SQLServer Express.

Como se explicó anteriormente , es el servidor el encargado de realizar las tareas mas costosas en cuanto a recursos.

### 4.2.1 Componentes

El WebService está constituido por un punto de acceso público, a través del cual los usuarios pueden enviar la información, y por diferentes componentes encargados de

realizar las tareas de procesamiento y almacenamiento de los datos generados por los usuarios. A continuación se explicaran en detalle cada uno de estos componentes.4.3

#### 4.2.1.1 Recepción de Datos

Utilizando el punto de acceso el usuario envía un archivo comprimido. Dentro de este paquete de datos pueden existir una cantidad variable de documentos, uno por cada día de logueo, dependiendo de cuantos días han transcurrido desde la última conexión. Una vez almacenado, en un directorio local, el WebService envía una confirmación al cliente indicando que el archivo fue recibido correctamente.

#### 4.2.1.2 Procesamiento de Datos

Una vez finalizada la comunicación y una vez descomprimidos los documentos, el servidor comienza el procesamiento de los datos recibidos en el paquete. El procesamiento de los datos se realiza de manera secuencial por orden cronológico. El sistema lee cada documento realizando un parseo de sus datos, obteniendo los Log-Types, sus Properties y Values, para luego almacenar los datos en la base de datos relacional. Cuando finaliza el procesamiento de los datos, los documentos son eliminados conservando un backup del paquete recibido.

#### 4.2.1.3 Almacenamiento de Datos

Los paquetes recibidos por el WebService son almacenados en un directorio para cada usuario específico, creado por el mismo servicio. En el caso de existir un error en el procesamiento de un documento, este no será eliminado del directorio. De esta manara es posible verificar fácilmente los datos procesados incorrectamente. Los datos procesados correctamente son almacenados en la base de datos, en la cual se registran la información relacionada a sus documentos y el usuario de origen. De esta manera es posible localizar para cada dato almacenado, cual fue su paquete de origen, así como su documento dentro de este y a que usuario pertenece.

## 4.3 Base de Datos

Debido a la gran cantidad de información obtenida por la aplicación y la necesidad de consultar de diferentes formas los datos obtenidos para la búsqueda de patrones, es indispensable la correcta organización de los datos y la utilización de una estrategia que

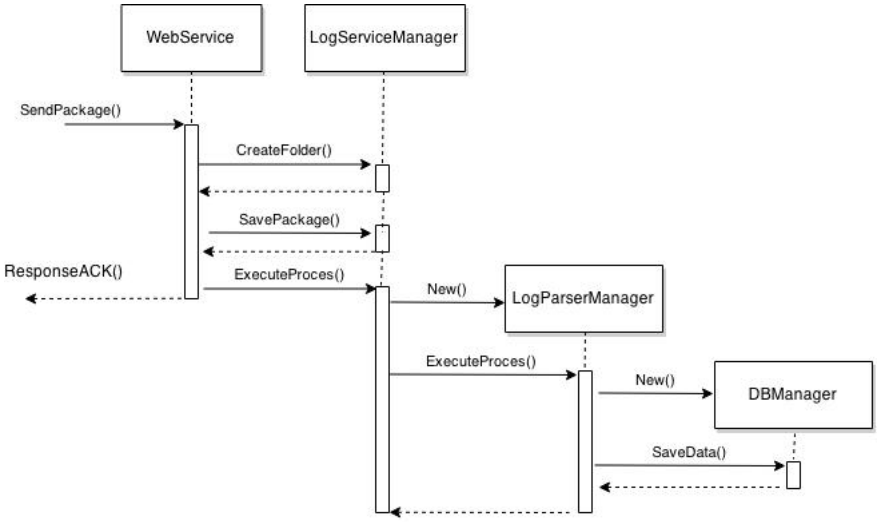



**Figure 4.3:** Diagrama de Secuencia, procesamiento del paquete de datos

nos permita obtener información de manera eficaz. Es por este motivo que se utiliza una base de datos relacional, controlada por el motor de bases de datos SQLServer Express.

### 4.3.1 Estructura

Dentro de la base de datos, la información se encuentra organizada en diferentes tablas de manera tal que nos permita optimizar el espacio utilizado. A continuación se describen cada una de las tablas utilizadas y de qué manera se utiliza su información. 4.4

**User**

Almacena la información de los usuarios registrados en el sistema. En base a los datos de esta tabla se puede obtener la fecha de la última conexión, la versión y modelo de dispositivo utilizado y un identificador único del aparato.

**File**

Almacena los datos correspondientes a los documentos procesados por el servicio, se pueden obtener datos como el tamaño, nombre y a que usuario pertenece.

**LocationGroup**

Almacena los *LocationGroups* pertenecientes a cada usuario, ubicación del *LocationGroup* y la cantidad de localizaciones que contiene.

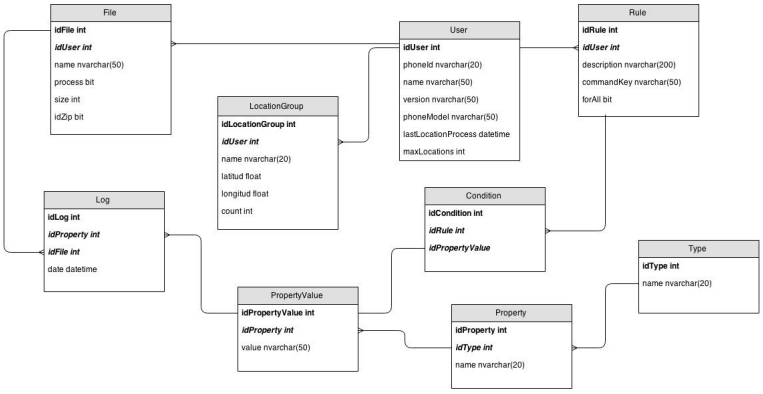



**Figure 4.4:** Estructura Base de Datos

### Type

Contiene el listado de LogTypes utilizado por la aplicación a la hora de registrar eventos en el dispositivo.

### Property

Almacena un listado de cada posible *LogProperty* relacionada con su *LogType* correspondiente.

### PropertyValue

Se encuentran los posibles valores para cada *LogProperty*, los cuales pueden ser compartidos por diferentes usuarios. Un ejemplo recurrente es el caso en el que dos usuarios se conectan a una misma red Wifi.

### Rule

Contiene las reglas especificadas para cada usuario del sistema, así como un código que indica que comando se deberá ejecutar cuando esta se cumpla.

### Condition

Almacena las condiciones de una regla específica. Cada una de estas indica un *LogProperty* y su *PropertyValue* necesarios para que la condición pueda ser validada.

Chapter 5

# Experimentos

No hay necesidad de explicar!!

Chapter 6

# Conclusiones

Explicar los resultados de los experimento y que se desprende. Por ejemplo lo que nos contaste de los 500 y 600 trabajos que uno falla 5 y en el otro 30. Limitaciones de tu trabajo y posibles extensiones.

# Bibliography


[1] M. Ettus A. Smailagic. System design and power optimization for mobile computers. *IEEE Computer Society Annual Symposium on VLSI*, pages 10–14, 2002.

[2] Alejandro Zunino Ana Rodriguez, Cristian Mateos. Mobile devices-aware refactorings for scientific computational kernels. In *Proceedings of 13th Argentine Symposium on Technology, 41th JAIIO*, 2012.

[3] Jukka K. Nurminen Antti P. Miettinen. Energy efficiency of mobile clients in cloud computing. pages 4–4, 2010.

[4] Sokol; Mei-Alessandro; Stefa Julinda Barbera, Marco V.; Kosta. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. *INFOCOM, Proceedings IEEE*, pages 1285–1293, 2013.

[5] Aaron Beach, Mike Gartrell, Xinyu Xing, Richard Han, Qin Lv, Shivakant Mishra, and Karim Seada. Fusing mobile, sensor, and social data to fully enable context-aware computing. In *In HotMobile Workshop*, pages 60–65, 2010.

[6] Krzysztof; J. Nalepa-Grzegorz Bobek, Szymon; Porzycki. Learning sensors usage patterns in mobile context-aware systems. *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, page Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, 2013.

[7] Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y. Charlie Hu, and Andrew Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. *SIGMETRICS Perform. Eval. Rev.*, 41(1):29–40, June 2013.

[8] C.; Niyato-D.; Wang P. Dinh, H. T; Lee. A survey of mobile cloud computing: architecture, applications, and approaches. *Wirel. Commun. Mob. Comput. 2013*, page 1587âĂŞ1611, 2013.

[9] Greg Meredith-Sanjiva Weerawarana Erik Christensen, Francisco Curbera. Web services description language (wsdl) 1.1. *W3C*, 2001.

[10] Google. Fused location. `https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi`.

[11] Google. Proyecto volta. `https://developer.android.com/intl/es/about/versions/android-5.0.html#Power`.

[12] C.C Holt. *Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages*. Carnegie Institute of Technology, 1957.

[13] Alejandro Zunino Juan Manuel Rodriguez, Cristian Mateos. Are smartphones really useful for scientific computing? *Lecture Notes In Computer Science*, 7547:38–47, 2012.

[14] Marcelo Campo Juan Manuel Rodriguez, Alejandro Zunino. Introducing mobile devices into grid systems: a survey. *International Journal of Web and Grid Services 2011*, 7:1–40, 2011.

[15] Dimitrios Lymberopoulos Kaisen Lin, Aman Kansal and Feng Zhao. Energy-accuracy trade-off for continuous mobile device location. In *ACM Mobisys*, 2010.

[16] Chang Keun; Seo Sin Seok; Choi Mi Jung; Hong James Won Ki Kang, Joon Myung; Park. *Challenges for Next Generation Network Operations and Service Management*, chapter User-Centric Prediction for Battery Lifetime of Mobile Devices, pages 531–534. Springer Berlin Heidelberg, 2008.

[17] A.R. ; Kitey S. Karmore, S.P. ; Mahajan. Battery monitoring and analysis for android based system. *Advanced Computing Technologies (ICACT), 2013 15th International Conference on*, pages 1 – 6, 2013.

[18] Yung-Hsiang Lu Karthik Kumar. Cloud computing for mobile users: can offloading computation save energy? *Computer*, 43:51–56, 2010.

[19] Lisa Mahapatra. Android vs. ios: WhatâĂŹs the most popular mobile operating system in your country?, November 2013.

[20] Geoffrey Holmes Bernhard Pfahringer Peter Reutemann Ian H. Witten Mark Hall, Eibe Frank. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[21] Microsoft. Fundamentos de la normalizaciÃşn de bases de datos. `https://support.microsoft.com/es-es/kb/283878`.

[22] Wolfgang Narz. High-level energy saving strategies for mobile location-based services on android devices. *ICWMC 2013 : The Ninth International Conference on Wireless and Mobile Communications*, pages 238–243, 2013.

[23] Wenny Rahayu Niroshinie Fernando, Seng W. Loke. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29:84–106, 2013.

[24] Joseph A. Paradiso and Thad Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing, IEEE Computer Society*, 4:18–27, 2005.

[25] J. Parkkila, J. ; Porras. Improving battery life and performance of mobile devices with cyber foraging. *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, pages 91–95, 2011.

[26] F.H.P. ; Sasso G. ; Kellerer W. ; Widmer J. Perrucci, G.P. ; Fitzek. On the impact of 2g and 3g network usage for mobile phones' battery life. *Wireless Conference, 2009. EW 2009. European*, pages 255–259, 2009.

[27] Raquel Zambrano Ramirez. Bases de datos relacionales. pages 1–6, 2008.

[28] Andrew Rice and Simon Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, 6:593–606, 2010.

[29] B. ; Sommer C. ; Dressler F. Segata, M.; Bloessl. Towards energy efficient smart phone applications: Energy models for offloading tasks into the cloud. *IEEE International Conference on Communications (ICC)*, pages 2394 – 2399, 2014.

[30] Dennis Sosnoski. "code first" web services reconsidered. `http://www.infoq.com/articles/sosnoski-code-first`.

[31] Daniel Gatica-Pereza Trinh Minh Tri Doa. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12:79âĂŞ91, 2013.

[32] J. Vallina-Rodriguez, N. ; Crowcroft. Energy management techniques in modern mobile handsets. *Communications Surveys & Tutorials, IEEE*, 15:179–198, 2013.

[33] Daniel T. Wagner, Andrew Rice, and Alastair R. Beresford. Device analyzer: Large-scale mobile data collection. *SIGMETRICS Perform. Eval. Rev.*, 41(4):53–56, April 2014.

[34] DanielT. Wagner, Andrew Rice, and AlastairR. Beresford. Device analyzer: Understanding smartphone usage. In Ivan Stojmenovic, Zixue Cheng, and Song Guo, editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 131 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 195–208. Springer International Publishing, 2014.

[35] DanielT. Wagner, Andrew Rice, and AlastairR. Beresford. Device analyzer: Understanding smartphone usage. In Ivan Stojmenovic, Zixue Cheng, and Song Guo, editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 131 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 195–208. Springer International Publishing, 2014.

[36] Lee W McKnight William Lehr. Wireless internet access: 3g vs. wifi? *Telecommunications Policy*, Volume 27:351–370, JuneâĂŞJuly 2003.

[37] P.R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6:324âĂŞ342, 1960.

[38] Rich Wolski Ye Wen and Chandra Krintz. *Lecture Notes in Computer Science*, chapter Online Prediction of Battery Lifetime for Embedded and Mobile Devices, pages 57–72. Springer Berlin Heidelberg, 2005.