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: Ing. Juan Manuel Rodríguez

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

Tandil,

Resumen

Pequeño resumen.

Agradecimientos

Contents

List of Figures

List of Tables

Lista de algoritmos

Acrónimos

SEAS Simple Energy Aware Scheduler

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 con capacidades limitadas de almacenamiento y procesamiento, generalmente de propósito específico, como agendas electrónicas o teléfonos celulares, 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 [6, 5, 14]. Debido a sus capacidades los dispositivos móviles son ampliamente utilizados por las personas. En el año 2011ya 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 [17].

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 celulares, permitiendo ejecutar diversas aplicaciones, también aumentó 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 [14]. Por estas razones, uno de los grandes desafíos a los que se enfrentan los fabricantes de dispositivos móviles, y desarrolladores de aplicaciones [3] 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. 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. 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 potecial estrategia para reducir el consumo de batería, es optimizar el uso de este tipo de tecnologias [16, 17]. 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 [1, 2]. Existen resultados que demuestran que la ejecución remota de la aplicación puede ahorrar energía de manera significativa [10, 13, 15].

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 [11], 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. 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 [20].

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 [20].

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 [22][9].

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 pantea 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 [20].

Una vez recolectada la información correspondiente a diferentes usuarios, se plantea la posibilidad de detectar, a través de técnicas especificas de inteligencia artificia, 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 [12], 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 disposi-

tivos 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 **??** 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.

2.1 Predicción del consumo de bateria basado en patrones de uso [8]

La clave para minimizar el consumo de bateria 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 móbiles, 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, video llamadas, 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.

2.1.1 Suposiciones

Para este método propuesto 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, musica y llamadas en espera
- se pueden definir posibles estados en términos de las diferences operaciones funcionales del dispositivo
- el consumo promedio de bateria 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

2.1.2 Definición del problema

Se consideran los siguientes vectores:

- $B = (B1, ..., Bn)(1)$
- $p = (p1, ..., pn)(2)$
- $R = (R1, ..., Rn)(3)$

Donde $R_i = p_i \,.\, B_i(4)$

B_i: promedio del consumo de bateria en el estado n
p_i: tiempo insumido por el estado n, teniendo en cuenta que $\sum_{i=1}^{n} pi = 1$
R_i: proporción de bateria consumida en el estado i

Entonces se puede predecir el tiempo de duración de la bateria en el estado i de la siguiente manera:

$T = \frac{V}{\sum_{i=1}^{n} Ri} = \frac{V}{\sum_{i=1}^{n} pi.Bi}(5)$

Donde T es el tiempo de duración predecido y V es el total de bateria.

2.1.3 Solución propuesta

Desde el momento en el que se enciende el dispostitivo se comienza a almacenar información, dejando registro en un log de porción de 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 (B) y patrones de uso (p).

Cada entrada del log se puede definir mediante el vector $D_j = (D_j^1, D_j^2, D_j^3)$, donde D_j^i son los datos almacenados en la entrada j para el estado i. D^1es un estado del dispositivo, D^2es el nivel de bateria y D^3es el tiempo insumido. Entonces, utilizando la series de datos almacenados y partiendo de que consideramos n estados posibles y m entradas en el log, se puede representar con el vector $b = (b_1, \ldots, b_n)$al total de bateria consumida en el estado i, el cual se obtiene de sumar los D^2de cada entrada del log para dicho estado. De igual modo se puede representar el tiempo consumido por el vector $t = (t_1, \ldots, t_n)$sumando los D^3de cada entrada del log para el estado i.

Por lo tanto,

$$b_i = \sum_{j=1}^{m} D_j^2 \, (6)$$

$$t_i = \sum_{j=1}^{m} D_j^3 \, (7)$$

Donde $D_j^1 = i$. Entonces el consumo promedio y patrón de uso para cada usuario, utilizando las ecuaciones (6) y (7) están definidos por $B_i = \frac{b_i}{t_i}$,$p_i = \frac{t_i}{\sum_{i=1}^{n} t_i}$,donde $B_i$es el consumo promedio de bateria en el estado i y p_i es la porción de tiempo insumida en el estado i, teniendo en cuenta que $\sum_{i=1}^{n} p_i = 1$. Este patrón de uso no solo representa el patron 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 [4, 21] para predecir patrones de uso basándose en la serie de datos recolectados. Por lo tanto se puede calcular el tiempo remanente de vida de la batería (T) utilizando la ecuación (5) con B y p.

2.2 Potencial ahorro de energía delegando tareas a la *"nube" [18]*

La idea principal de esta propuesta es delegar el procesamiento de tareas computacionalmente costosas a la *"nube"*, en vez de hacerlo localmente. Se realiza la siguiente suposición: la energía insumida en la transferencia de estas tareas hacia la *nube* y la descarga de los resultados, es menor que la utilizada para el procesamiento de dichas tareas con la CPU local.

En este contexto se pueden distinguir dos posibles objetivos de optimización:

1. Minimizar el consumo de bateria 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 incluída en el cálculo del consumo

Independientemente de las optimizaciones antes mencionadas, también se deben obtener los pros y contras entre procesar las tareas de forma local y la energía consumida por la tranmisión de forma inalámbrica de los datos hacia y desde la nube.

La clave está en obtener modelos que permitan predecir el consumo energético dado un tipo de tecnología de transmición, como Wifi, 3G, 2G, la proporción de señar 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.

2.2.1 Entorno de medición y metodología

El primer paso para realizar mediciones de energía es configurar y validar un entorno de medición. En cuanto a la metodología de medición, debido a la falta de funcionalidad del sistema operativo Android (utilizado para la prueba) la unica posibilidad es conectado un dispositivo de medición física al teléfono inteligente. La manera más común es mediante la conexión de un voltímetro y un medidor de amperios entre la batería y el teléfono Cuando se utilizan las API de Android, en cambio, la única métrica que se puede recoger es el porcentaje de batería consumido.

Se utiliza entonces un voltimetro especial, que permite conectarlo a una computadora para grabar los resultados obtenidos. Básicamente, se pueden emplear dos estrategias de muestreo. La primera es donde un valor instantáneo de la muestra se mide, mientras que con la segunda estrategia se registra el promedio a lo largo todo el periodo de muestreo. El primer método es el mas utilizado en el contexto del problema, ya que las tecnologías y protocolos subyacentes operan en algunos órdenes de magnitud más rápidos que el intervalo de muestreo. Por lo tanto, una enorme cantidad de repeticiones serían necesarias con el fin de obtener resultados significativos. El segundo método, que calcula la potencia media durante el intervalo de muestreo, es el mas adecuado, ya que el principal interés reside en conocer el consumo total de energía, que es a su vez completamente reflejado por las muestras promediadas.

Con el fin de realizar mediciones reproducibles, se automatiza el proceso de prueba para que no se requiera ninguna interacción con el teléfono durante el curso de las mediciones. Para este motivo se utiliza una aplicación para Android que permite definir una campaña

de medición en un formato de texto simple. La aplicación requiere permisos especiales para evitar ser interrumpida cuando el teléfono cambia al modo de espera.

Para definir una campaña de medición se especifican una serie de archivos a ser descargados o cargados, y tiempos de espera entre cada transferencia. Estos tiempos de espera son importantes especialmente en mediciones que involucren redes 3G, donde el dispositivo puede continuar en un estado de alto consumo energético aún luego de que la transferencia del archivo se haya completado. Este puede ser el caso en el que el dispositivo mantiene una conexión activa, aunque en reposo, con el proveedor, produciendo un consumo de energía adicional el cual debe ser tenido en cuenta en las mediciones.

De esta manera, realizando pruebas con diferentes tamaños de archivo, 10MB, 20MB y 50MB, y diversas tecnologías de transmisión, 2G, 3G y WiFi se construye un modelo de consumo energético que permite obtener un indicativo de la energía consumida para cargar o descargar una cierta cantidad de datos utilizando alguna de las tecnologías de comunicación inalámbricas disponibles. Dado que existe una relación directa entre la cantidad de datos enviados y el consumo energético, el modelo se puede representar de la siguiente manera:

$[J] = \beta\, X[MB]$

utilizando ambas, regresión lineal (para obtener β_{avg}, el consumo medio de energía por MB) y la regresión cuantílica (para obtener β_{10} y β_{90}, el percentil 10 y 90) para ajustar los parámetros. La ventaja de la regresión cuantílica por sobre solo proporcionar la desviación estándar, por ejemplo, es que considera que el 80% de los resultados caen en el intervalo dado.

Luego, utilizando este modelo, se puede comparar el consumo energético necesario para procesar una tarea de forma remota, con la cantidad de energía consumida para llevar a cabo la misma tarea de forma local y tomar la decisión de si conviene que la tarea se procesada de forma remota o local.

2.3 Manejo inteligente de métodos de localización, para optimizar el consumo energético [7]

Las aplicaciones móviles a menudo necesitan datos de localización, para actualizar localmente información relevante y adaptar el contexto de dispositivo. Aunque la mayoría de los teléfonos inteligentes incluyen un receptor GPS, su uso es en general restringido debido de alto consumo de la batería. Esta propuesta presenta un servicio para obtener la localización, que ayuda a reducir el consumo de batería. El diseño se basa en el hecho de que precisión de la ubicación requerida varía según el lugar, por lo tanto métodos de

localización con menor costo energético y menos precisión, como aquellos basados en WiFi y triangulación de antenas celulares, a veces pueden ser utilizados. Este método determina automáticamente la precisión requerida para aplicaciones basadas en búsquedas móviles. Mienstras 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).

2.3.1 Observaciones generales

Se tienen en consideración las siguientes observaciones:

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 energia.

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.

2.3.2 Solución propuesta

El flujo general del sistema se puede describir de la siguiente manera:

- Entrada

- Cálculo de la precisión dinámica requerida: para aquellas aplicaciones basadas en búsquedas móviles se calcula en base a las entidades buscadas, pero puede darse el caso de que la aplicación especifique directamente la precisión requerida

- Modelo energético de cada sensor: representa la cantidad de energía utilizada por cada sensor para obtener la localización del dispositivo.

- Modelo de precisión dinámica de cada sensor: representa la calidad de la información sobre la localización que cada uno ofrece. Un desafío clave en este punto, es

que la disponibilidad y exactitud de los sensores de localización varía con la ubicación. Por ejemplo, el GPS puede no funcionar en interiores o en áreas con vista de satélite obstruido, la triangulación WiFi puede funcionar mejor cuando el número de puntos de acceso es alta, etc. Este modelo tiene en cuenta tales efectos.

- Algoritmo de selección de sensor: determina el sensor a ser utilizado en cada caso. Se utilizan un modelo de estimación Bayesiana para combinar los datos del sensor con la localización predecida y asi proveer una estimación de máxima verosimilitud.
- Después de que se ha seleccionado un sensor de localización, se gasta energía para utilizar ese sensor y los datos producidos por dicho sensor se utilizarán para generar una estimación de la localización. La estimación de la localización se envía a la aplicación cliente. Además, los datos de localización y sensores pueden ser usados para mejorar el modelo de la precisión del sensor.

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

2.4 Predicción de la próxima ubicación del usuario y que aplicación utilizará [19]

Esta propuesta, aunque no está ligada al ahorro energético, si esta relacionada con nuestro trabajo, dado que 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 próximo 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.

Este enfoque permite modelar la interacción entre las variables predichas, para estudiar las relaciones entre el lugar donde un usuario está y la posibilidad de que éste realice una llamada telefónica, utilice la cámara, etc. Por otra parte, otras fuentes de información, como la lista de dispositivos Bluetooth cercanos o información del sistema, también se puede explotar con el fin de enriquecer el contexto del usuario y mejorar los modelos de predicción.

2.4.1 Solución propuesta

Se utiliza un set 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, elegiendo entre 22 mutalmente excluyente etiquetas, incluyendo una etiqueta "Otros" para aquellos casos en el que ninguna de las 21 restantes sea apropidada. 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.

Para reducir el número de etiquetas, se agrupan en 8 grupos mas generales:

- Casa
- Trabajo
- Casa-de-amigo: casa de un amigo o familiar
- Trabjo-de-amigo: lugar de trabajo o escuela de un amigo o familiar
- Restaurante
- Estacion-Transporte: lugar estático como una estación de subte, parada de colectivo, etc.
- Shopping
- Entretenimiento

Aquellas regiones sin etiquetar, o que no caigan dentro de ninguna de estas 8 etiquetas, serán consideradas dentro de la etiqueta "Otros". Adicionalmente, se tienen dos categorías especiales:

1. Transición: para un contexto de viaje
2. Sin localización: para estados de localización desconocida debido a errores o falta de información.

Mientras que "Transición" está considerado dentro de las etiquetas a predecir, el caso de "Sin localización" no es considerado para las predicciones.

Dado que los usuarios usan un número limitado de aplicaciones de su dispositivo, solo se consideran para las predicciones aquellas mas relevantes, utilizadas al menos una vez a la semana.

Toda la información antes mencionada se agrupa en intervalos de tiempo de 10 minutos de duración y se clasifica en seis categorías diferentes:

1. Localización
2. Aplicaciones
3. Tiempo
4. Registro de llamadas
5. Proximidad Bluetooth
6. Estado interno

Estas categorías juntas describen el uso general del dispositivo para cada usuario. Luego, utiliando 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.

Chapter 3

Propuesta

Para enfrentar la problemática explicada en capítulos anteriores se propone LoguinUse, una aplicación para dispositivos móviles con sistema operativo Android versión 2.3.3 o superior, que debido a su gran popularidad nos provee un número aceptable de usuarios. La información obtenida por dicha aplicación será comprimida y enviada a un servidor dedicado, desarrollado sobre la plataforma .NET Framework 3.5 (FW 3.5) que provee un WebService SOAP y una posee una base de datos relacional SQL Server 2005 Express. También se crea una aplicación de escritorio sobre FW 3.5 con la capacidad de filtrar y exportar la infromación desde la base de datos a archivos en formato ARF para su posterior estudio, mediante métodos de Datamining con la herramienta WEKA.

En este capítulo se detallaran los diferentes aspectos tenidos en cuenta para el desarrollo del sistema, asi 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 un rendimiento óptimo a la hora de obtener y almacernar 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

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 mantener alojados en memoria objetos que de otra manera no hubieran sido necesarios. Si bien esta estrategia requiere de un mayor uso de memoria RAM, también simplifica el acceso a los estados almacenados por la aplicación, evitando la creación de nuevos objetos para la lectura de estados en cada evento monitoreado del sistema.

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. Ademas 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 practidicad, debido a que la plataforma .NET nos 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*. Este permite la definición de un servicio a partir del código, por lo que no es necesario tener ningun conocimiento sobre WSDL. Se desarrollan los métodos de la interface del servicio para luego generar el servicio, utilizando las herramientas provistas por el fremawork .NET 3.5.

El servidor soporta la mayor carga de procesamiento del sistema, ya que se deben parsear 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 ARF para su posterior estudio.

3.5 Estrategia Base de Datos

La información se almacena en una base de datos relacional SQL Server Express 2005, la cual consta de un motor muy poderoso y optimizado con el que se pueden realizar consultas rápidas, lo cual es un beneficio a la hora de extraer los datos para su estudio.

La base de datos contiene grandes volúmenes de información, los cuales podrán ser consultados mediante la utilización de filtros de la manera más adecuada a la hora de exportar los archivos ARF.

Con el objetivo de minimizar el espacio utilizado por la base la datos relacional, se decidió compartir datos entre usuarios, de esta manera se evita 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.

Chapter 4

Implementación

4.1 Aplicación Android

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 analizarlos datos.

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

4.1.1.1 Type

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: *Type*. 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.

4.1.1.2 Property

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 mas 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.

4.1.1.3 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 *Value* como el valor almacenado en un momento dado, para una *Property* específica, incluida en el tipo de log monitoreado. Cabe destacar que una *Property* pude tener únicamente un *Value* en un momento dado.

A continuación se muestra un ejemplo de un evento monitoreado, con su estructura y valores.

[09:15:11]&[WIFI]&[STATE::1 | SSID::"E1200P" | MAC::C0:65:99:A7:BC:E0 | ACCESS::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. A continuación se describen cada uno de ellos.

4.1.2.1 Servicio

Este componente es el encargado de monitorear el dispositivo constantemente, haciendo uso de los *Receivers* (ver4.1.2.2) del sistema. Se instancia al encenderse el aparato, inicializa los *Receivers* y comienza el monitoreo del dispositivo. El servicio es el componente escencial de la aplicación, ya que conserva en memoria los diferentes objetos involucrados en el monitoreo y captura de la información del dispositivo. Para asegurar un monitoreo contstante, este servicio se incializa en lo que se llama modo *foreground*, lo que hace que siempre este en ejecución. De esta manera el sistema operativo, en caso de necesitar recursos, no considerará matar el servicio. Solo se puede detener de forma manual. Es por

esta razón es que al iniciar el servicio, y siempre que el mismo se encuentre en ejecución, se muestra en la barra de estado del dispositivo una notificación de que dicho servicio se encuentra corriendo.

4.1.2.2 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 *Log-Type*. Cada receiver registra un único Log-Type y durante su ejecución, al momento de detectar el evento correspondiente, obtiene un listado de pares Property-Value 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 Log-Types y su correspondiente Receiver responsable del monitoreo.

Los *Receivers* utilizan el comportamiento provisto por las clases *BroadcastReceiver* y los *IntentFilter*, propias del sistema operativo Android. Un *BroadcastReceiver* es un proceso que corre en segundo plano y puede ser activado por llamadas a través de *Intents*. Para lograr esto se utilizan *IntentFilters* con los cuales se filtran las llamadas del sistema, para solo atender a las llamadas que correspondan y/o sean de interés.

El sistema Android lanza eventos, en forma de *Intents*, cuando ocurre un acontecimiento importante, un cambio de estado en el dispositivo. Cada *Receiver*, durante su creación, se suscribe al evento correspondiente. De esta manera se evita la existencia de uno o varios procesos activos, encargado de monitorear constantemente el estado del dispositivo cada intervalos fijos de tiempo, ahorrando una gran cantidad de energía.

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.

4.1.2.3 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.

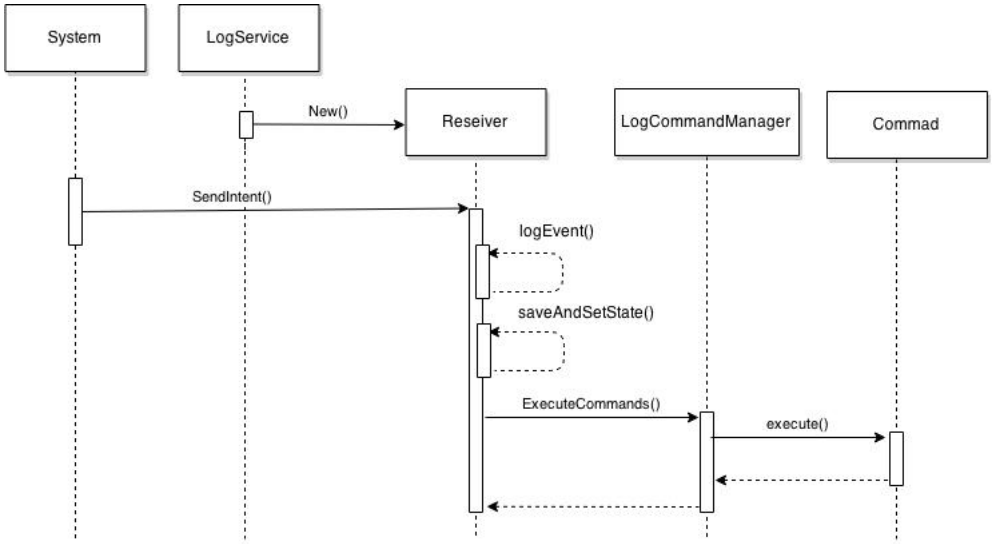

Figure 4.1: Diagrama de secuencia Log-Event se puede observar el mecanismo de monitoreo del sistema

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.

```
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

El ejemplo anterior muestra que solo en el caso de que la variable *"change"* sea verdadera se registra la información. Este mecanismo provee flexibilidad para agregar nuevos Log-Types e incrementar el número de *Properties* dentro de un Log-Type, sin alterar el funcionamiento descripto.

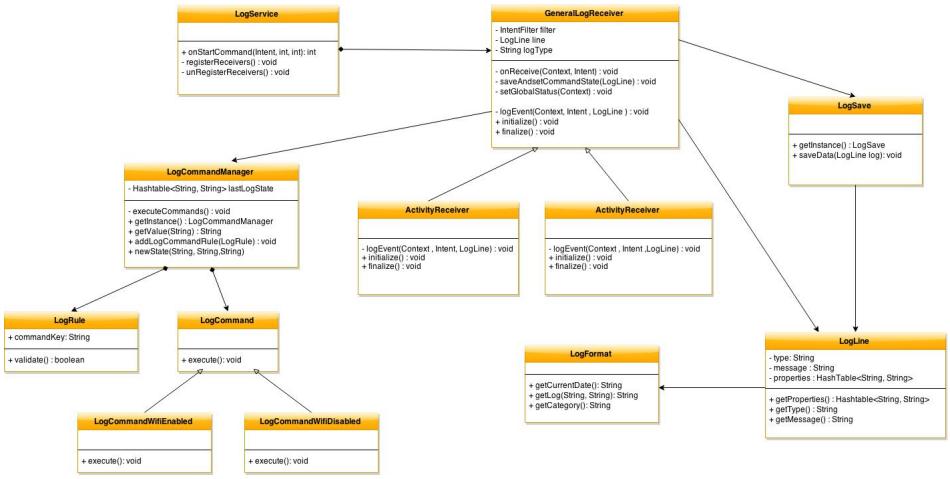

Figure 4.2: Diagrama de Clases: Aplicación Android

Dentro del *CommandManager* también existen *Commands*. Cada *Command* hereda la interface 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.

Es posible configurar cada *Command* mediante el uso de las reglas, *Rules*. Cada *Rule* contiene un sub-vector compuesto por pares Property-Value y una Key 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 con anterioridad. 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 influir en el consumo de enegía del dispositivo, a la hora de enviar los datos recolectados al servidor, la aplicación tiene configurado, por defecto, enviar los datos solo si el aparato se encuentra cargando la batería y tienen acceso a internet mediante Wifi. Así no solo no existe impacto en el consumo de batería durante el envío de datos, sino que además al evitar el uso de redes 3G, que son en general mucho mas lentas y menos estables que las redes Wifi, no existen riesgos de ocasionar gastos extras

al usuario.

Para evitar particionar los datos recolectados, se decidió no enviar el archivo correspondiente al día en curso, debido a que es probable que continúe siendo utilizado. 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 en la sobre la plataforma .Net con una aplicación Android. Este WebService nos 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 constituído 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.

4.2.1.1 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 ACK 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

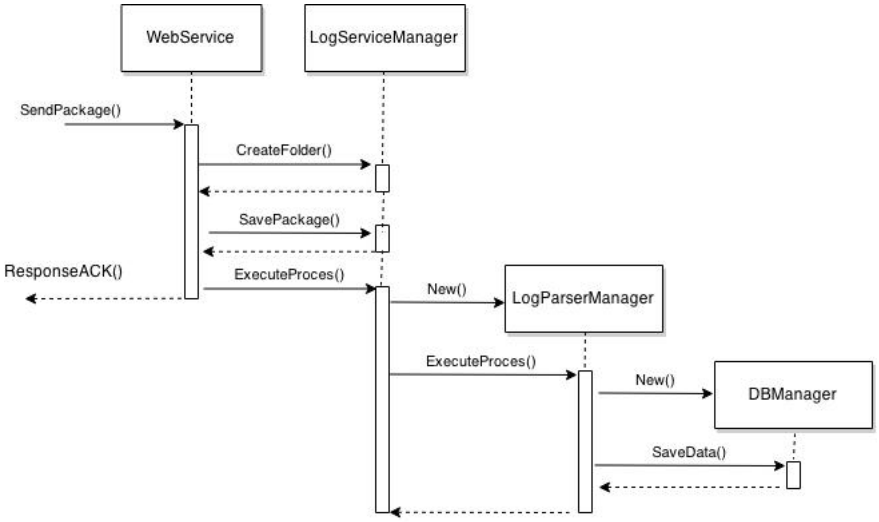

Figure 4.3: Diagrama de Secuencia, procesamiento del paquete de datos

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 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 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.

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

Almacdena 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.

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.

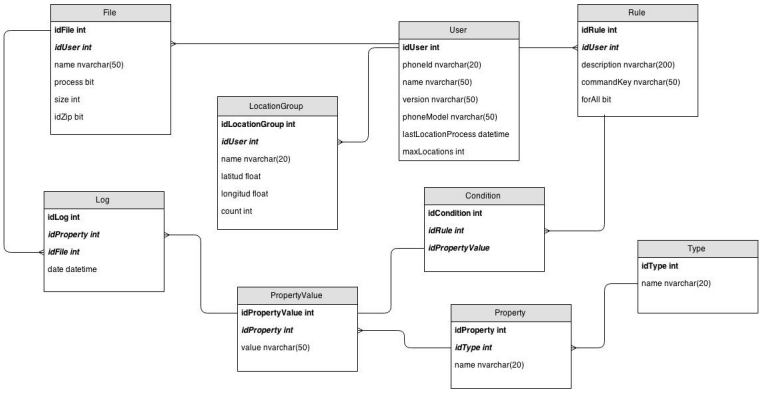

Figure 4.4: Estructura Base de Datos

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] *Energy efficiency of mobile clients in cloud computing*, 2010.

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

[3] 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.

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

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

[6] 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.

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

[8] 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.

[9] 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.

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

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

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

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

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

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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

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

[22] M.A. ; Nassr R. Zahid, I. ; Ali. Android smartphone: Battery saving service. *Research and Innovation in Information Systems (ICRIIS), 2011 International Conference on*, pages 1–4, 2011.