Centro de Investigación y de Estudios
Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Mecanismo ubicuo de localización de dispositivos móviles

Tesis que presenta

Juan Carlos Pérez Pérez

para obtener el Grado de

Maestro en Ciencias de Computación

Director de la Tesis

Dr. José Guadalupe Rodríguez García

México, D.F. **Agosto 2011**

II

Resumen

La determinación de la ubicación física se ha vuelto un aspecto importante dentro del cómputo móvil, debido a que ayuda a ofrecer información más exacta al usuario, dependiendo en donde esté ubicado. Para las aplicaciones que son utilizadas en exteriores, el Sistema de Posicionamiento Global (GPS) es una solución muy recurrida, sin embargo, su funcionamiento no es conveniente en el interior de inmuebles o en áreas urbanas con densidad considerable de edificios altos, ya que es necesario que exista línea de vista entre el receptor y los satélites. Este trabajo trata sobre el diseño e implementación de un mecanismo de localización omnipresente que aprovecha las ventajas ofrecidas por los dispositivos móviles.

La propuesta determina la ubicación del usuario (de un dispositivo móvil) en cualquier momento, ya sea a través de las coordenadas GPS, de una dirección de Internet o del identificador de célula en las redes GSM. Estos mecanismos nos dan una aproximación de la ubicación actual del usuario.

Una vez que el dispositivo obtiene la información de su ubicación, la envía hacia un servidor a través de la primera red disponible, *i.e.* Internet o red celular. Preferentemente, el mecanismo establece la comunicación a través de una red inalámbrica (Wi-Fi), realizando la búsqueda de redes disponibles en la cercanía. Si no encuentra la red necesaria, se establece una conexión de datos GSM, para enviar la información de la ubicación del usuario.

Se hace uso del Protocolo de Inicio de Sesiones (Session Initiation Protocol) y de la comunicación oportunista, para mejorar la eficiencia en la comunicación entre cliente y servidor. Los resultados obtenidos muestran que se trata de un mecanismo factible para rastrear la ubicación de un dispositivo, mediante el uso de distintas tecnologías existentes como el cómputo móvil, los protocolos de Internet, las redes celulares y las aplicaciones Web.

Abstract

The determination of physical location is becoming an important aspect of mobile computing because it helps to bring more accurate information to the user, depending on where it is located. For applications used in outdoor areas the Global Positioning System (GPS) is a common choice. Nevertheless, its performance degrades in indoor areas and in high-dense building urban areas because it needs a clear line of sight to the satellites. This work deals with the design and implementation of an ubiquitous location mechanism and takes advantage of the facilities brought by mobile devices.

Our proposal can determine the user's location at any time, either by using GPS coordinates, an Internet address, or a Cell ID in GSM networks. All these mechanisms give us an approximation of the user's current location.

Once the device obtains information about its location, sends it to a server through the fist available network found i.e. Internet or GSM. Preferably, the mechanism establishes communication by means of a Wi-Fi network. First, it does a scanning search for nearby Wi-Fi available networks, if no Wi-Fi networks are found, a GSM data packet connection is established for sending the user's location information.

We make use of the Session Initiation Protocol (SIP) and the opportunistic communication mechanisms to improve the efficiency of the client-server communication. The obtained results show that it is a feasible mechanism to track a device's location, through the use of different existing technologies like mobile computing, Internet protocols, GSM networks and Web applications.

Agradecimientos

Doy gracias a Dios, por haberme permitido llegar hasta esta etapa de la vida, a lo largo de la cual, nunca me ha dejado de su mano y siempre ha sido la luz que guía mi camino.

A mi mamá Irma, por darme la existencia y ser un ejemplo de constante superación y tenacidad ante las dificultades. A mi tía Martha por haberme inculcado valores fundamentales que forjaron mi personalidad y carácter. A mi madrina Fiorela, por su invaluable ayuda para poder continuar con mi formación profesional, que la vida y el Creador recompensen con creces por todo lo que me ha dado.

A mis hermanos Rebeca, Horacio y Osiris, por su cariño y amor incondicional a lo largo de todos estos años. A mi primo Paco, por ser como un hermano más con quien compartir vivencias y aprendizaje.

A Linda, por ser mi compañera por tantos años y brindarme su amor, confianza y apoyo en todo momento. Por ser una motivación constante en la búsqueda continua del éxito en todos los ámbitos de la vida.

A mi asesor, el Dr. José Guadalupe, por su paciencia, comprensión y tutoría durante los años que pasé estudiando el posgrado. Gracias por el tiempo que dedicó a transmitir su experiencia y conocimientos académicos para construir un trabajo de calidad.

A los Dres. Jorge Buenabad Chávez y Manuel Romero Salcedo, por su orientación, aportaciones y consejos para dar forma a este trabajo de tesis y por aportar un punto de vista alternativo.

Al Consejo Nacional de Ciencia y Tecnología, por otorgarme la beca que me permitió estudiar la maestría y sin la cual este logro no hubiera sido posible. Del mismo modo, agradezco al Departamento de Computación del Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional por darme la oportunidad de formarme en sus aulas y ser un orgulloso egresado de tan prestigiosa institución.

A Sofía Reza, por su incansable ayuda para todos los que estudian y trabajan en el Departamento de Computación.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La tendencia tecnológica actual ha permitido el uso cada vez mayor de dispositivos personales, cuyas funcionalidades ponen al alcance de las personas una inmensidad de aplicaciones, que posibilitan diversos usos en la vida diaria. La principal característica de estos dispositivos es la movilidad, la cual permite el aprovechamiento de sus funciones en cualquier momento y lugar.

Existe una amplia variedad de dispositivos móviles que va desde los teléfonos celulares, que cuentan con acceso básico a Internet, hasta laptops, que poseen prácticamente la misma funcionalidad de una computadora de escritorio. Las diferencias entre estos dispositivos radican en el tamaño, las características, la funcionalidad y el costo.

La amplia propagación del cómputo móvil está cambiando la forma de desarrollar, implementar y acceder a los servicios de Internet. Los avances recientes en la conectividad inalámbrica, y el creciente mercado de dispositivos móviles, estimulan el ofrecimiento de servicios a un amplio conjunto de terminales cliente con recursos limitados y heterogéneos.

La creciente presencia del cómputo en la vida de las personas a través de los dispositivos móviles, hace posible diversificar su uso en aplicaciones prácticas para mejorar el bienestar y la comodidad personales. Asimismo, las redes de telecomunicaciones son capaces de transmitir una gran cantidad de información de forma ininterrumpida, permitiendo la alta disponibilidad de servicios y aplicaciones. Como ejemplos basta mencionar las redes 3G y WiMax.

Con el paso del tiempo se ha buscado diversificar el uso que se le puede dar a los dispositivos móviles, desde el entretenimiento hasta la salud y el cuidado personal. Recientemente han surgido aplicaciones que utilizan la ubicación física del dispositivo móvil para ofrecer servicios personalizados o que solo se encuentran disponibles en la localidad en la que reside el usuario durante un momento determinado.

El método más utilizado para determinar la ubicación de una entidad móvil se

conoce como GPS (*Global Positioning System*), este sistema ofrece posicionamiento continuo e información de tiempo en cualquier parte del mundo y se conforma de una constelación de satélites organizados en planos orbitales alrededor de la Tierra.

Los usuarios cuentan con un receptor GPS a través del cual se reciben señales que pueden ser utilizadas para determinar su posición, sin embargo, para que esto funcione debe de haber línea de vista entre el receptor y los satélites, lo cual excluye el uso de GPS en el interior de los edificios o en espacios cerrados.

La necesidad de línea de vista se presenta como una limitación en la localización de dispositivos móviles, por lo tanto es necesario incorporar una tecnología alternativa para la localización en interiores. El acceso a Internet de forma inalámbrica, desde un dispositivo móvil permite la creación de un mecanismo alternativo de localización basado en la red a la que se encuentre conectado el usuario.

El posicionamiento global se ha desarrollado como una tecnología que tiene gran utilidad en la navegación terrestre, sobre todo en áreas urbanas. Sin embargo, en la mayoría de las veces no es suficiente con conocer la posición en la que se encuentra una persona, sino también es importante notificar esta posición a otras entidades, ya sea con fines personales o de seguridad.

Por otro lado, la amplia proliferación de dispositivos de cómputo en el entorno, ha dado lugar a la computación ubicua, es decir, a la integración de estos dispositivos en cualquier ámbito de la vida diaria, a veces sin que el usuario se percate de su presencia, contando a la vez con un dispositivo que le ofrece conectividad y comunicación. Sin duda alguna, los dispositivos móviles han fomentado la generación de un ambiente de cómputo ubicuo, debido a su amplia propagación y a la movilidad que permite utilizarlos en cualquier lugar.

1.1 Planteamiento del problema

La gran disponibilidad de herramientas tecnológicas y el surgimiento de áreas de investigación como la computación ubicua nos permite proponer un mecanismo de rastreo o localización de dispositivos móviles que tienen acceso a la red de forma inalámbrica, tales como teléfonos celulares, PDAs (*Personal Digital Assistant*), *smartphones* y computadoras portátiles, a través de redes con diferentes características, *e.g.* las redes WiFi y las redes de telefonía celular.

Este mecanismo hará uso de las últimas tecnologías de acceso inalámbrico a Internet así como del protocolo de comunicación conocido como SIP *(Session Initiation Protocol)* [1], para el establecimiento y gestión de las sesiones de usuario.

Un mecanismo de localización de ésta índole es muy útil debido a que permite la ubicación del usuario cuando accede a Internet desde cualquier punto a su alcance. Las aplicaciones para este sistema son variadas y van desde los ya conocidos servicios dependientes de la ubicación hasta aplicaciones de seguridad en dispositivos móviles y personas que tienen acceso a Internet inalámbrico.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar los mecanismos que brinden un servicio de localización de alta disponibilidad enfocado a los dispositivos móviles, permitiendo el uso de redes con diferentes características.

1.2.2 Objetivos particulares

- Estudiar áreas emergentes del cómputo móvil y cómputo ubicuo como la comunicación oportunista y la omnipresencia (ubicuidad).
- Diseñar mecanismos que garanticen la comunicación permanente a través de redes con diferentes características.
- Desarrollo de un mecanismo de comunicación inalámbrico entre la tecnología WiFi y la celular.
- Desarrollo de la plataforma de comunicación.

Para lograr los objetivos planteados, se implementaron aplicaciones para dispositivos móviles, las cuales ofrecen un servicio automatizado de obtención y reporte de la ubicación, así como establecen la comunicación con un servidor central, a través de redes IP. Las aplicaciones son fáciles de utilizar y requieren el mínimo de interacción por parte del usuario, apegándose a los principios del cómputo ubicuo y el uso intuitivo e imperceptible de sus aplicaciones.

1.3 Metodología

La aplicación puede ser instalada en los dispositivos móviles más comunes en el mercado y en los que tienen un mayor potencial de crecimiento a mediano plazo. Esto implica que en el mecanismo propuesto interactúen distintas plataformas como son Linux, Java Mobile Edition y Android.

Tanto en el cliente como en el servidor converge la conectividad hacia redes IP con diferentes especificaciones: redes WiFi, Ethernet y celulares (3G o GRPS). Es tarea de la aplicación aprovechar estos medios de comunicación de forma oportunista para establecer la comunicación extremo a extremo.

La metodología seguida para especificar e implementar esta propuesta de investigación consiste en los siguientes pasos:

1. Investigar los avances más recientes en el área así como las características de la tecnología actual, que permiten el desarrollo del proyecto. Se analizan los antecedentes y fundamentos de la propuesta para determinar el impacto, el alcance y la utilidad del mismo.
2. Identificar los requerimientos principales de las aplicaciones, con el fin de determinar las herramientas necesarias para el desarrollo de las mismas.
3. Aprovechar la infraestructura de comunicación existente, la cual incluye el PBX *Asterisk* para el uso de Voz sobre IP, y realizar pruebas de comunicación entre distintos dispositivos como *smartphones* y *softphones*.
4. Especificar los componentes e infraestructura del sistema final a partir de la planificación y diseño de la arquitectura.
5. Desarrollar aplicaciones tanto para el cliente (móviles) como el servidor, con la capacidad de comunicarse entre sí.
6. Realizar pruebas de funcionalidad y rendimiento, para verificar que se cumplan con los objetivos planteados anteriormente.
7. Reportar cada uno de los puntos descritos y las conclusiones del trabajo en el presente documento de tesis.

1.4 Requerimientos del sistema

A partir de los conceptos detallados, es posible realizar la abstracción de las funciones básicas que el sistema debe proporcionar, así como los objetivos del mismo.

El objetivo general de esta propuesta es desarrollar los mecanismos que brinden un servicio de localización a través de redes con diferentes características de

comunicación y transmisión de datos. Para lograr nuestro objetivo, es necesario utilizar dispositivos que sean capaces de conectarse a diferentes tipos de redes, es decir, dispositivos que cuenten con conectividad inalámbrica *WiFi* o GPRS.

Dado que la meta final de nuestra propuesta es el rastreo de dispositivos que están continuamente en movimiento, es necesario contar con sistema de posicionamiento global (GPS) que nos provea información en términos de coordenadas geográficas. Hasta hace unos años la única forma de hacer esto posible era mediante la comunicación por radio entre el dispositivo y un receptor GPS, sin embargo, actualmente una cantidad creciente de teléfonos móviles cuentan con GPS integrado.

Al contar con un método de posicionamiento y de comunicación inalámbrica, el dispositivo móvil enviará de forma periódica a un servidor, la información que haga posible su rastreo y aproximación de posición actual, o en su defecto la posición más reciente. Es muy importante que esta actividad cíclica no obstaculice el funcionamiento normal del dispositivo portátil, ni que interrumpa al usuario en el desarrollo de sus actividades normales. Asimismo, se debe minimizar el procesamiento de información ya que los dispositivos móviles se caracterizan por tener recursos limitados de energía y memoria.

Se debe tener en cuenta que un usuario puede utilizar distintos dispositivos a lo largo de su actividad diaria, por lo tanto es importante contar con un método de administración de sesiones por medio del cual un usuario indique el comienzo o el final de su sesión en uno u otro dispositivo. Para enfrentar este escenario, se utilizará el protocolo SIP, el cual se encarga del establecimiento y administración de las sesiones de usuario.

Aunque la mayor parte de la información proviene del lado del cliente, es decir, de los dispositivos móviles que integran el receptor GPS y poseen conectividad inalámbrica, es necesario que el servidor reciba la información de posicionamiento enviada desde los móviles y que a la vez almacene y administre los registros de usuario y lleve un historial de sus ubicaciones, con la finalidad de que esta información sea consultada posteriormente y sirva para conocer la ubicación del dispositivo en un momento determinado.

Por último, para prevenir la falta de conectividad ocasionada por la ausencia de transmisión de datos por vía celular, o por las continuas desconexiones de las redes *WiFi* en la cercanía, se planean utilizar los fundamentos de la comunicación oportunista para buscar y aprovechar la presencia de cualquier red inalámbrica, con la finalidad de transmitir los datos de la ubicación por cualquier medio de comunicación, favoreciendo la conexión vía WiFi, para reducir costos e incrementar la fiabilidad de la comunicación.

1.5 Organización de la tesis

La tesis se encuentra organizada en seis capítulos con la siguiente estructura: en el capítulo 2 se presenta el estado de la tecnología existente al día de hoy, así como los trabajos relacionados con nuestra propuesta. Los conceptos expuestos sirven de antecedente y justificación de nuestro trabajo.

En el capítulo 3 se especifica la arquitectura de la solución, apegándonos al método conocido como *Ingeniería de Software*. Se detallan los módulos que componen a cada entidad del sistema, los servicios que ofrecen y sus casos de uso.

En el capítulo 4 se detalla a mayor profundidad la forma en que se implementaron las aplicaciones cliente y servidor, las interfaces de programación utilizadas y la plataforma en la cual se instaló y configuró el sistema final.

En el capítulo 5 se plantean las pruebas a las que se sometieron cada una de las aplicaciones y sus componentes. Se reportan los resultados obtenidos y algunas observaciones que consideramos importantes.

Por último, en el capítulo 6 se obtienen las conclusiones generales de este trabajo de investigación y se describe el posible trabajo futuro a partir de nuestros resultados.

Capítulo 2

Estado del arte y trabajos relacionados

La tendencia tecnológica actual ha permitido el uso cada vez mayor de dispositivos personales, cuyas funcionalidades ponen al alcance de las personas una inmensidad de aplicaciones, que posibilitan diversos usos en la vida diaria. La principal característica de estos dispositivos es la movilidad, la cual permite el aprovechamiento de sus funciones en cualquier momento y lugar.

En este capítulo se describen los conceptos fundamentales y se hace una revisión de algunos trabajos relacionados con esta propuesta de tesis, partiendo de las definiciones y describiendo la evolución que han tenido los dispositivos móviles durante el avance tecnológico reciente. La propuesta del tesista junto con el asesor se fundamenta principalmente en el cómputo ubicuo, el cual es conocido como el paradigma que surge de los componentes y servicios brindados por el cómputo móvil.

Comúnmente los términos móvil e inalámbrico se utilizan de forma indistinta, sin embargo son diferentes. Un dispositivo móvil es aquel que puede ser utilizado en movimiento por una persona e incluye desde celulares hasta laptops; se caracterizan por tener recursos limitados y conectividad intermitente. En cambio, un dispositivo inalámbrico es el que utiliza una red inalámbrica para enviar y recibir datos (a través de señales de radio) y cuya ubicación puede ser fija o móvil.

Existen aplicaciones que no se consideran móviles, porque utilizan redes inalámbricas fijas que proveen acceso a Internet en un ambiente determinado (hotspots).

Las aplicaciones móviles que no tienen acceso inalámbrico se encuentran presentes en laptops y PDAs (*Personal Digital Assistant*), por lo tanto los datos deben ser sincronizados y almacenados en el dispositivo para su posterior uso [2].

Se describirán asimismo otros conceptos básicos de la propuesta, como es la comunicación oportunista utilizada para la transmisión de datos desde los

dispositivos móviles, ya sea utilizando una red inalámbrica Wi-Fi o la red celular. Por último se cubrirá el concepto de alta disponibilidad, propiedad que se busca tenga el mecanismo de localización.

2.1 Comunicaciones inalámbricas

Convencionalmente la tecnología de sistemas de comunicación se enfocaba únicamente a la transmisión, recepción y difusión de la información. Originalmente, toda comunicación era análoga, luego, la tecnología digital permitió la comunicación entre las computadoras y en nuestros días, incluso las aplicaciones convencionales como el radio y la televisión se están volviendo digitales, debido a la necesidad de mejor calidad y fiabilidad.

Cuando el emisor y el receptor no están conectados por un cable, se le llama comunicación inalámbrica. Esto es básicamente la comunicación por radio y el espectro de frecuencia en cuestión va desde la onda corta hasta las frecuencias de microondas.

La tecnología celular de comunicaciones por radio fue un avance considerable en las comunicaciones inalámbricas digitales, específicamente para la telefonía móvil. Entretanto, el desarrollo del Protocolo de Internet (IP) fue paralelo, permitiendo el surgimiento de los "servicios de datos" celulares, como el correo electrónico y la navegación móvil, que combinan las capacidades de estas innovaciones para ofrecer verdadera movilidad en términos de comunicación y cómputo [3].

El Internet cubre cuatro necesidades primarias de la sociedad: comunicación, intercambio de conocimiento, comercio y entretenimiento [4]. Esta convergencia se conoce como TIC (Tecnologías de la Información y Comunicación). A través de las TIC estamos avanzando hacia la sociedad basada en la información y cubriendo la necesidad de acceder a datos, información y conocimiento desde cualquier lugar y en cualquier momento.

Al igual que las computadoras, la evolución de la tecnología inalámbrica se ha definido por generaciones [4]. La primera generación (1G) utilizó la tecnología analógica conocida como FDMA (*Frequency Division Multiple Access*) para la modulación. La segunda generación (2G) utilizó tecnología digitalizada, es decir, una combinación de TDMA (*Time Division Multiple Access*) y FDMA, como ejemplo tenemos a GSM (*Global System for Mobile Communications*). En la primera y segunda generaciones, la información es gestionada a través de circuitos, de forma similar a lo que ocurre en una conexión *dial-up*, a esto se le conoce como CSD (*Circuit Switched Data*).

La siguiente fase en la evolución es 2.5G, en donde la voz también es

digitalizada, a través de un circuito y los datos se envían en paquetes, como ejemplo de esta tecnología está GPRS (*General Packet Radio Service*). La tecnología inalámbrica de tercera generación o 3G realiza un gran cambio desde el punto de vista tecnológico, debido a que utiliza técnicas de difusión de espectro para el acceso y codificación del medio, es decir, tanto la voz como los datos se transmiten por medio de paquetes. Ejemplos de esta tecnología son UMTS (*Universal Mobile Telecommunications System*) y CDMA2000 (*Code Division Multiple Access*).

Mientras que 1G, 2G y 3G dejan huella entre las MANs (*Metropolitan Area Network*), la tecnología inalámbrica se está volviendo popular en las redes LAN (*Local Area Network*) y PAN (*Personal Area Network*), debido a que ofrecen comodidad y flexibilidad. A causa del éxito de la telefonía inalámbrica y los servicios de mensajería de texto, la comunicación inalámbrica está comenzando a ser aplicada en el campo del cómputo personal y de negocios, dentro del dominio de las redes LAN [4]. Las redes LAN inalámbricas o WLAN (*Wireless* LAN) se han desplegado tanto en casas, universidades y establecimientos comerciales, como en medios de transporte y vehículos comerciales.

El dominio de las redes inalámbricas de datos hoy comprende las redes PAN inalámbricas (Bluetooth e infrarrojo), WLAN (familia IEEE 802.11) y WAN (*Wide Area Network*) inalámbricas (GSM, GPRS y 3G).

Como se puede observar, existen muchas redes inalámbricas en la actualidad, lo que ha permitido la transmisión de voz y datos de cualquier origen a cualquier destino. Las telecomunicaciones multimedia también son una realidad: incluyen texto, audio e incluso imágenes siendo transmitidas entre teléfonos móviles [3]. El acceso a Internet desde los teléfonos móviles es soportado por el protocolo WAP (*Wireless Access Protocol*) [2].

2.2 Cómputo móvil

El término cómputo móvil es considerado como el siguiente paso del cómputo distribuido, en el cual la ubicación física de los elementos involucrados puede cambiar durante la ejecución de una aplicación. Los dispositivos que realizan este tipo de cómputo son vistos como pequeñas computadoras que contienen y ejecutan aplicaciones, accediendo a código y datos almacenados de forma local y remota [5].

Inicialmente el enfoque de la comunicación móvil iba hacia la transmisión de voz, para luego incluir el envío y recepción de datos, permitiendo así que actualmente los dispositivos móviles ofrezcan muchos servicios, entre los que se encuentran el correo electrónico, conectividad a Internet, mensajería instantánea,

agenda electrónica, juegos y demás aplicaciones.

Los servicios de comunicación personal y celular han revolucionado el campo de las telecomunicaciones porque, a pesar de que en un inicio su avance fue lento, se han desarrollado a nivel mundial. Asimismo, han tenido una penetración en el mercado por más tiempo y con mayor demanda que otras tecnologías. El número de usuarios de la comunicación inalámbrica se incrementa año con año [6].

Un sistema de cómputo móvil, como cualquier otro sistema de cómputo, puede estar conectado a una red, sin embargo, esto no es un requisito para que sea considerado móvil. Lo que sí es necesario es que la conectividad entre los dispositivos móviles no dependa de un medio con ubicación fija, sino que éste permita la comunicación al mismo tiempo que la movilidad.

2.2.1 Funciones del cómputo móvil

Un ambiente de cómputo puede definirse como móvil si soporta una o más de las siguientes características [4]:

- **Movilidad del usuario**: El usuario debe ser capaz de moverse de una ubicación física a otra y usar el mismo servicio.
- **Movilidad de red**: El usuario puede desplazarse de una red a otra y utilizar el mismo servicio.
- **Movilidad del portador**: Debe ser posible moverse de un portador a otro y utilizar el mismo servicio. Por ejemplo, si la red 3G no está disponible, se debe tener conectividad con otro portador, como GPRS.
- **Movilidad del dispositivo**: El usuario puede cambiar de un dispositivo a otro y seguir utilizando el mismo servicio.
- **Movilidad de sesión**: Una sesión de usuario debería ser capaz de desplazarse de un entorno a otro.
- **Movilidad de huésped** (host): El dispositivo de usuario puede ser cliente o servidor.

Las funciones del cómputo móvil pueden ser divididas en los siguientes grandes segmentos [4]:

1. **Usuario con dispositivo**: El dispositivo puede ser desde una computadora portátil hasta un teléfono móvil.
2. **Red**: Cuando el usuario es móvil, utiliza diferentes redes en diferentes lugares a horas distintas.
3. **Puerta de enlace**: Esta es requerida como interfaz entre los distintos

soportes de transporte. Por ejemplo, con las teclas de un teléfono fijo generan señales DTMF (*Dual Tone Multi Frequency*). Estas señales analógicas se convierten a digitales a través de la puerta de enlace IVR (*Interactive Voice Response*), actuando como interfaz con una aplicación de computadora.

4. **Middleware**: Es más una función que un elemento tangible, ya que se encarga de la presentación y reproducción del contenido en un dispositivo móvil en particular. También se encarga de la seguridad y personalizar la aplicación para diferentes usuarios.
5. **Contenido**: Puede tratarse de una aplicación, sistema o incluso un agregado de sistemas. El contenido puede ser personal, corporativo o mercado masivo.

2.2.2 Cómputo móvil y redes inalámbricas

El cómputo móvil hace uso de distintos tipos de redes inalámbricas: GSM, GPRS, CDMA, WiFi, Bluetooth, etc. Es importante destacar que los dispositivos móviles también hacen uso de la red de telefonía fija. Aunque esta red incluye satélites y redes de microondas, se le denomina fija debido a que está diseñada sobre conductores tangibles. Forman parte de la telefonía fija las redes de banda ancha sobre DSL (*Digital Suscriber Loop*) o cable, así como el *backbone* de Internet [4].

A través de la conectividad inalámbrica, los dispositivos móviles tienen la posibilidad de conectarse con otros dispositivos en la red. Incluso esto ha propiciado la confusión entre los términos "cómputo móvil" y "comunicación inalámbrica". La comunicación inalámbrica es sólo un tipo de comunicación que se distingue por utilizar el espacio como canal de comunicación. Los sistemas de cómputo móvil simplemente utilizan este tipo de comunicación para obtener conectividad de red.

Recientemente, las redes de computadoras han evolucionado a pasos agigantados, haciendo que al día de hoy sea difícil concebir a la computación sin conectividad de red. La red y los dispositivos de cómputo se han mezclado hasta el punto de que no se puede imaginar uno sin el otro. Las redes y el cómputo distribuido son dos de los segmentos más importantes en los que se enfoca la investigación en cómputo [7].

En la actualidad, la demanda de acceso inalámbrico a Internet a través de los teléfonos móviles está en aumento, con un mayor énfasis en mayores velocidades y mejor desempeño. El buen desempeño del acceso a Internet se ve mermado en cuanto se considera la interferencia y los problemas relacionados con el ambiente inalámbrico.

La tendencia al día de hoy es hacia las redes 3G, puestas en marcha por muchos proveedores en todo el mundo. Bluetooth y 802.11b se han vuelto los estándares alternativos *de facto* para el acceso a Internet y datos de forma inalámbrica, cada uno con sus propias fortalezas y debilidades [3].

La amplia propagación del cómputo móvil está cambiando la forma de desarrollar, implementar y acceder a los servicios de Internet. Los avances recientes en la conectividad inalámbrica, y el creciente mercado de dispositivos móviles, estimulan el ofrecimiento de servicios a un amplio conjunto de terminales cliente con recursos limitados y heterogéneos. La introducción de estándares con mayor capacidad y eficiencia de servicios (*e.g.* 3G) ha impactado el diseño de los dispositivos móviles, los cuales son equipados con múltiples interfaces de comunicación, como son los dispositivos con conectividad Wi-Fi (IEEE 802.11) y Bluetooth [8].

El entorno móvil posee ciertas propiedades y requerimientos que lo distinguen:

1. Conciencia de la ubicación.
2. Calidad de servicio en la conectividad de red.
3. Capacidades limitadas de los dispositivos (en particular el almacenamiento y poder de cómputo).
4. Fuente de alimentación limitada.
5. Soporte para una amplia variedad de interfaces de usuario.
6. Proliferación de plataformas.
7. Transacciones activas.

Este conjunto de características debe ser la herramienta principal que guíe la arquitectura, diseño e implementación de la aplicación móvil. Sin embargo, al tratar de cumplir con los mencionados requisitos, las posibles soluciones pueden tener conflicto entre sí. Como en cualquier otro sistema, al diseñar aplicaciones móviles es necesario encontrar el balance entre las ventajas y desventajas de cada solución planteada.

Para el caso de nuestra propuesta, nos enfocaremos principalmente en el aspecto de la ubicación, sin que esto signifique que ignoremos las demás cualidades en el momento del diseño e implementación del sistema final. La posibilidad de conocer la ubicación es una de las características que distingue a los sistemas móviles de los sistemas estacionarios y sirve como una herramienta para automatizar tareas [7]. Como ejemplo se puede mencionar una aplicación móvil que utiliza la información de la ubicación para registrar las distintas rutas que toma un usuario para llegar de la casa al trabajo, sin necesidad de que el usuario vaya registrando la ubicación de forma manual. Con la información recabada se puede determinar la ruta más corta para trasladarse y mostrarla al

usuario.

Con el objetivo de ofrecer la mayor funcionalidad por parte de la aplicación móvil, es importante tomar en cuenta la ubicación del cliente, no solo para ofrecerle determinados resultados sino para personalizar los servicios. Esto puede depender de la ubicación del usuario, las características del dispositivo y también del estado actual de los recursos necesarios, ya sea dentro de la localidad del cliente o a nivel global [5].

Una red inalámbrica debe permitir que los objetos móviles se desplacen de una ubicación a otra, manteniendo la comunicación. Estos objetos móviles pueden recibir muchos servicios de varios proveedores en su ruta de tránsito. Para ofrecer estos servicios, es esencial que el proveedor tenga en cuenta la ubicación exacta del objeto móvil en cualquier momento [3].

Desde el inicio de la investigación en cómputo móvil, la ubicación ha sido reconocida como una propiedad crucial, que debe considerarse con el objetivo de desarrollar aplicaciones móviles efectivas y eficientes. Además, en ocasiones es necesario que la información de la ubicación sea propagada hacia los componentes de capas superiores, donde se pueda manejar la complejidad del procesamiento de la ubicación y se brinde una abstracción simplificada y fácil de utilizar a las aplicaciones móviles [5].

2.3 Arquitecturas para el desarrollo de aplicaciones móviles

El primer paso para construir una aplicación de software, después de haber recabado los requerimientos, es establecer a grandes rasgos un plan de cómo será la aplicación una vez que se haya terminado su implementación. Las aplicaciones móviles, como cualquier otra aplicación de software, necesitan de este plan, mejor conocido como "arquitectura de software". La arquitectura de software es una abstracción de alto nivel del sistema y una especificación de cómo colaboran sus componentes [7]; las decisiones que se tomen sobre la arquitectura son las más importantes durante todo el ciclo de vida de un sistema de software y sólo pueden mejorarse con la experiencia.

Muchos factores contribuyen al éxito o fracaso de una solución móvil, entre los cuales se encuentra el dispositivo, la conectividad de red inalámbrica y sobre todo la arquitectura de la aplicación. Existen muchos modelos de aplicación que pueden ser aprovechados en el desarrollo móvil, cada uno con un conjunto distinto de características, que lo vuelven apropiado para algunas aplicaciones e inapropiado para otras [2].

Es necesario evaluar el usuario objetivo, el tipo de dispositivo en el que residirá la aplicación, el consumo de energía del dispositivo (pila) y los requerimientos de seguridad, todos estos factores nos permiten seleccionar la arquitectura más adecuada para la aplicación móvil.

Sin embargo, cuando se trata de aplicaciones móviles, es difícil que un sólo modelo de arquitectura se ajuste a las necesidades de la aplicación, por lo tanto es necesario llegar una combinación óptima entre ventajas y desventajas, con el objetivo de beneficiar al usuario final tanto en facilidad de uso como en rango de dispositivos. En cuanto a la parte del desarrollador de la aplicación, se debe encontrar la correcta combinación entre complejidad y capacidades de la aplicación, con la finalidad de apegarse a los requerimientos y hacer que el desarrollo sea factible [2].

Existen tres tipos arquitecturas de aplicaciones móviles:

- Aplicaciones con conectividad inalámbrica.
- Aplicaciones *cliente inteligente.*
- Aplicaciones con conectividad alámbrica.

Las aplicaciones con conectividad (alámbrica e inalámbrica) tienen la misma arquitectura, ya que los componentes del sistema son esencialmente los mismos, la única diferencia es el medio de transmisión de la información hacia el usuario final.

En la arquitectura con conectividad inalámbrica, la lógica e información del sistema son almacenados en un servidor y del lado del cliente está el dispositivo móvil con capacidad de navegar por la red a través de un navegador o *micro-browser*. No es necesario algún otro software en el dispositivo móvil, por lo tanto a este tipo de aplicaciones también se les conoce como *cliente ligero.*

Las ventajas de la arquitectura con conectividad inalámbrica son: el mínimo desarrollo de software, la interfaz gráfica es familiar al usuario (similar a un navegador de PC), amplio rango de dispositivos compatibles (la mayoría integran su propio navegador), no es necesario actualizar la aplicación y brinda un alto nivel de seguridad ya que la información sensible reside en el servidor y no en el cliente. Las desventajas son: la dependencia de una conexión inalámbrica, el desempeño pobre de la aplicación (debido a la latencia de red), el bajo control que se tiene sobre el comportamiento de la aplicación, la disponibilidad del sistema (centralizada) y el costo de la comunicación (tarifas de conectividad celular).

Las aplicaciones de cliente inteligente son una alternativa poderosa ya que se diseña un software capaz de almacenar datos de forma persistente y de tener su propia lógica de funcionamiento. Este tipo de aplicaciones puede ser ejecutado en

cualquier momento incluso cuando no hay una conexión disponible. La información se sincroniza cuando el cliente se comunica con el servidor.

Las ventajas de las aplicaciones de cliente inteligente son: la alta disponibilidad de los datos (no necesita de conexión para obtenerla), interfaz de usuario rica en componentes, rendimiento (depende de las capacidades del dispositivo), cómputo distribuido (no sólo lo realiza el servidor), seguridad en la comunicación punto a punto y reducción de costos. Entre las desventajas se encuentra el incremento en la complejidad de desarrollo y las múltiples fases de implementación debido a las diferentes plataformas y sistemas operativos de los dispositivos móviles.

Las aplicaciones alámbricas o también conocidas como aplicaciones de mensajería utilizan medios como las alertas por correo electrónico y los mensajes de notificación entre aplicaciones.

Existen varias tecnologías de mensajería disponibles, pero las tres de mayor interés son: la mensajería de usuario a usuario (e-mail, mensajes de texto, mensajes multimedia, etc.), las notificaciones y alertas (WAP *Push*) y la mensajería de aplicación a aplicación (sincronización independiente del usuario). Las dos primeras tecnologías no necesitan el diseño de software adicional, sin embargo la mensajería entre aplicaciones requiere que el dispositivo sea capaz de comunicarse a través de un flujo de datos con el servidor.

Adicionalmente, las mensajería entre aplicaciones ofrece la posibilidad de almacenar información para su posterior envío (*store-and-forward*), de esta forma se resuelve el problema de la conectividad intermitente, conexiones rechazadas y falta de cobertura. Estos problemas son abordados por la comunicación oportunista, detallada más adelante.

Las ventajas de las aplicaciones de mensajería son: la capacidad de enviar mensajes *Push*, el almacenamiento y posterior envío de información y la entrega de datos personalizada (según horario o preferencias de usuario). Entre las pocas desventajas de esta tecnología se encuentra la dependencia con el proveedor de servicio de mensajería, ya sea de texto o multimedia.

Como se mencionó al inicio de esta sección, se buscó un equilibrio entre las distintas arquitecturas de aplicaciones móviles, tomando las características que mejor se adaptan a las necesidades de nuestra solución. En el siguiente capítulo se detalla la arquitectura tanto del mecanismo general como de la aplicación móvil propuestas.

2.3.1 Usuario móvil

Los datos y la información a través de servicios de cómputo móvil, son requeridos por muchas personas, sin importar si están en movimiento o no.

Así como los sistemas móviles son distintos a los sistemas estacionarios, los usuarios de cada uno también difieren. El usuario móvil se caracteriza por:

1. Estar en movimiento, al menos de forma ocasional, entre distintas ubicaciones.
2. No está enfocado en la tarea de cómputo.
3. Frecuentemente requiere de altos grados de disponibilidad y capacidad de respuesta por parte del sistema.
4. Cambiar de tarea frecuentemente o de forma abrupta.
5. Puede requerir de acceso al sistema en cualquier parte y cualquier momento (cómputo ubicuo).

Puede parecer trivial decir que el usuario móvil está frecuentemente en movimiento, sin embargo, este desplazamiento tiene una consecuencia muy importante: se puede utilizar la información de la ubicación física para sacar conclusiones sobre el contexto en el cual el usuario utiliza la aplicación [7].

2.3.2 Aplicaciones del cómputo móvil

Las posibles aplicaciones del cómputo móvil son casi ilimitadas tanto en los campos socio-económicos, industriales como comerciales. El cómputo móvil proporcionará el impulso necesario al comercio electrónico, a través de una serie de aplicaciones de comercio móvil (*m-commerce*).

En el contexto de las aplicaciones socio-económicas, la tecnología del cómputo móvil sería muy eficaz y valiosa en el entorno rural, donde la infraestructura básica puede ser carente o poco fiable para soportar el cómputo convencional y la tecnología de la comunicación.

El comercio depende críticamente de la velocidad de ejecución del ciclo de comercio en una transacción. Los ciclos implican actividades como catalogar, mostrar, revisar, seleccionar y finalmente ordenar y pagar. Todos estos pasos pueden acelerarse para que sean completados en un corto periodo de tiempo, incluso cuando el cliente esté viajando, con el uso apropiado de la tecnología móvil [3]. Las tendencias tecnológicas actuales del cómputo móvil ofrecen oportunidades hasta ahora no previstas en la implementación efectiva del comercio móvil.

En cuanto a aplicaciones empresariales de cómputo móvil, el *framework* MVC (Modelo, Vista, Controlador) se vuelve parte central para hacer frente a los retos que implica desarrollar una solución flexible, segura, adaptable, modular y portátil.

El contenido de una aplicación móvil dependerá principalmente del estilo de vida del usuario. Desde esta perspectiva, las aplicaciones móviles pueden agruparse en las siguientes categorías [4]:

- **Personal**: Pertenece al usuario (*e.g.* cartera, registros médicos, diario)
- **Efímera**: Es sensible al tiempo y su relevancia pasa rápidamente (*e.g.* noticias, clima, deportes, negocios, cotizaciones de bolsa).
- **Orientada a transacciones**: Las transacciones deben ser cerradas (*e.g.* transacciones bancarias, pago de servicios, tienda en línea).
- **Ubicación específica**: Información relacionada con la ubicación geográfica actual (*e.g.* mapa con dirección de calles, guía de restaurantes).
- **Corporativa**: Información corporativa de negocios (*e.g.* correo, inventario, directorio, alertas de negocio, recordatorios).
- **Entretenimiento**: Aplicaciones para diversión y entretenimiento (*e.g.* juegos, chat, televisión).

2.3.3 Seguridad en el cómputo móvil

Las cuestiones de seguridad en el entorno del cómputo móvil plantean un reto especial, debido a que se ofrecen servicios "por el aire", utilizando redes sobre las cuales no se tiene ningún control. Si tomamos en cuenta que toda la infraestructura y tecnología diseñada por GSM y demás foros es para incrementar los ingresos de los operadores de red, esto vuelve a la tecnología compleja y altamente dependiente de los operadores. Como ejemplo, la tecnología SMS (*Short Message Service*) se centra en el operador; WAP requiere de una puerta de enlace instalada en la red del operador y administrada por él mismo. La política de seguridad implementada por el operador de red depende de su prioridad y del potencial para generar ingresos mas no de las necesidades del proveedor de contenido.

En el ambiente móvil el usuario puede desplazarse entre una red y otra o de un dispositivo a otro. Por lo tanto, las implementaciones de seguridad, en teoría, necesitan ser independientes del dispositivo y de la red. El requisito es llegar a un modelo de seguridad que pueda ofrecer seguridad homogénea de extremo a extremo [4].

2.3.4 Lenguajes de programación móvil

Cada una de las plataformas disponibles para desarrollar aplicaciones móviles utiliza un lenguaje de programación distinto: Java, C++, Python, etc. implicando de este modo ventajas y desventajas desde el punto de vista del desarrollador y del usuario. El reporte *Developer Economics 2010* [9] nos ofrece datos relevantes para llevar a cabo la selección de una plataforma de desarrollo móvil, tomando en cuenta tanto el punto de vista técnico del programador como el aspecto económico y de mercado.

Para el caso de nuestra propuesta el único enfoque que tomaremos en cuenta será el técnico, que involucra aspectos como la rapidez de aprendizaje y codificación (Ver Figura 2.1), la documentación existente y el entorno integrado de desarrollo (IDE y emulador).

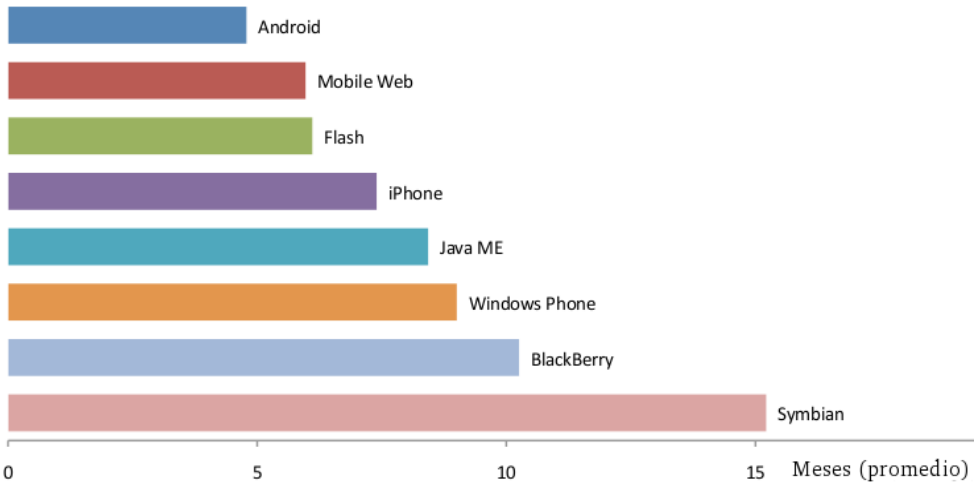


Figura 2.1: Tiempo que lleva dominar una plataforma

La plataforma Symbian ofrece grandes ventajas debido a que, al utilizar el lenguaje C++ para desarrollar, permite el acceso a una gran cantidad de funcionalidades en el dispositivo móvil. Sin embargo, el aprendizaje de este lenguaje así como la codificación de aplicaciones básicas conlleva un mayor lapso de tiempo que con otras plataformas disponibles (Ver Figura 2.2).

Otro punto en contra de Symbian es la poca portabilidad que ofrece, ya que además de que el código en C++ sólo se ejecutaría en un teléfono de esta plataforma, es necesario adaptarlo para las distintas versiones de este sistema. El hecho de que C++ ofrezca rapidez de ejecución y bajo consumo de memoria, deja de ser relevante con la aparición de *smartphones* con mejores capacidades [10].

Para favorecer el impacto de nuestra aplicación, vale la pena tomar en cuenta la base de dispositivos que utiliza la mayor parte de los usuarios. El reporte consultado nos muestra una gráfica de móviles por plataforma hacia la mitad del año 2010, la cual se muestra en la Figura 2.3.

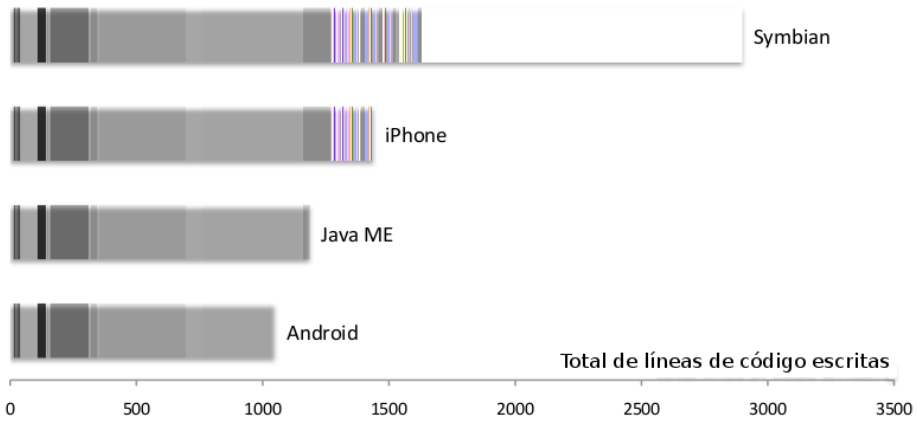


Figura 2.2: Líneas de código por plataforma

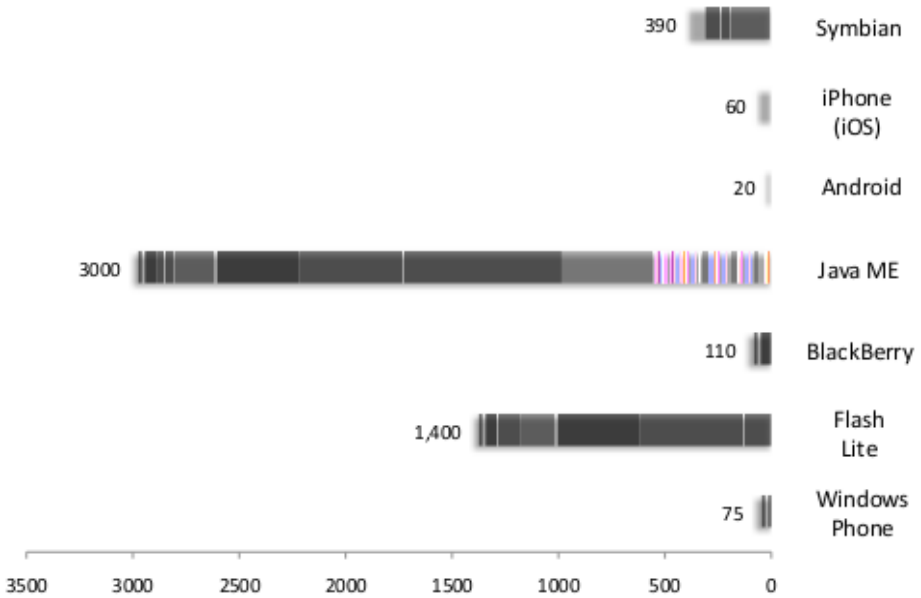


Figura 2.3: Millones de dispositivos móviles por plataforma hacia la mitad de 2010

2.4 Cómputo ubicuo

El término "ubicuo", que significa que aparece o existe en todas partes,

combinado con el cómputo, forman el concepto de cómputo ubicuo (UbiComp), utilizado para describir los sistemas ICT (*Information and Communication Technology*), que permiten que la información y las tareas estén disponibles en todas partes y que soporte un uso intuitivo y aparentemente invisible al usuario [11]. Mark Weiser acuñó el término "cómputo ubicuo" en 1988 durante su trabajo en Xerox PARC (*Palo Alto Research Center*) [12].

Es posible considerar al cómputo ubicuo como un ambiente de cómputo que está en todos lados y desaparece a la vez (porque nadie nota su presencia) [4].

En este modelo de interacción hombre-máquina el procesamiento de información ha sido integrado minuciosamente en los objetos y actividades de la vida diaria. El individuo que utiliza el cómputo ubicuo hace uso de diversos dispositivos y sistemas computacionales de forma simultánea, incluso sin darse cuenta.

Todos los modelos de cómputo ubicuo comparten la visión de dispositivos pequeños, económicos y con procesamiento robusto de red, que se encuentran distribuidos en todos los entornos y generalmente van enfocados hacia fines comunes al entorno real.

Cuando la tecnología del cómputo está inmersa en esos dispositivos, la interacción humano-computadora (*Human-Computer Interaction*) desempeña un rol crítico en la eficacia, eficiencia y experiencia del usuario. Esto debido a que la información móvil y los dispositivos de comunicación cada vez reducen más su tamaño y están restringidos con respecto a la presentación de la información, entrada de datos y control de diálogo.

Los retos de las interfaces humano-computadora son [4]:

1. La interacción debe ser coherente de un dispositivo a otro.
2. La interacción debe ser adecuada al dispositivo en particular y al entorno en el cual el sistema es utilizado.

De este modo, el cómputo ubicuo presenta retos a las ciencias de la computación, debido a que los modelos contemporáneos de HCI (línea de comandos, interfaz gráfica, etc.) son inapropiados y no se adecúan a los requerimientos del cómputo ubicuo, por lo tanto, es necesario un paradigma de interacción que sea "natural" para la computación ubicua. Los dispositivos móviles junto con los reproductores multimedia, las etiquetas RFID (*Radio Frequency Identification*), los GPS (*Global Positioning System*) y los pizarrones interactivos ofrecen cierto soporte a la interacción natural con el cómputo.

2.5 Información de la ubicación

Se conoce así a los datos que nos ayudan a identificar la ubicación de un usuario o dispositivo. Esto se puede lograr de dos formas: desde el dispositivo o desde la red.

Desde el dispositivo, la mejor forma de obtener la ubicación es a través de GPS. Los sistemas basados en esta tecnología ofrecen precisión con un radio de 3 metros. En algunas redes como GSM, se puede aproximar la posición del usuario a partir de la ubicación de la estación base a la cual está asociado el dispositivo; esta última ubicación se puede determinar a partir del identificador de célula (*CellID*).

Desde la red, la ubicación de un dispositivo puede ser determinada por medio de tecnología de sincronización avanzada. La posición obtenida a partir de la estación base es precisa en un radio de 100 metros [4].

2.5.1 GPS

GPS es un sistema de navegación basado en satélites que fue desarrollado por el departamento de defensa de los Estados Unidos a principios de los años 70's [13]. Este sistema ofrece posicionamiento continuo en cualquier parte del mundo. Debido a que sirve a un número ilimitado de usuarios y es utilizado con propósitos de seguridad, el GPS es un sistema pasivo, *i.e.* los usuarios únicamente reciben las señales del satélite.

GPS consiste de una constelación de satélites organizados en planos orbitales que comenzó a ofrecerse a los usuarios civiles en julio de 1995. Consiste de tres segmentos: de espacio, de control y de usuario. El segmento de espacio consiste de la constelación de satélites antes mencionada; cada satélite transmite una señal que contiene la distancia entre el receptor y los satélites GPS y las coordenadas de los satélites en función del tiempo, entre otras cosas. Las señales son controladas por relojes atómicos de alta precisión, incluidos en cada satélite; esta precisión es necesaria debido a que una variación de una milésima de segundo en los cálculos puede resultar en una variación de 300 km [13].

El segmento de control consiste de una red mundial de estaciones de rastreo. Su tarea principal es rastrear los satélites GPS con el objetivo de determinar la ubicación, la integridad del sistema, el comportamiento de los relojes atómicos y la información atmosférica entre otras consideraciones; esta información se envía a los satélites.

El segmento de usuario corresponde a las personas que cuentan con un receptor

conectado a una antena GPS a través de la cual se reciben señales que pueden ser utilizadas para determinar su posición en cualquier parte del mundo [13]. Sin embargo, para que esto funcione debe de haber *línea de vista* entre el receptor y los satélites, lo cual excluye el uso de GPS en el interior de los edificios. Para que un dispositivo móvil haga uso de este sistema debe incluir un receptor GPS dentro de su arquitectura.

La idea básica de GPS es muy simple: si se conocen las distancias de un punto en la Tierra hacia tres satélites así como la ubicación de los satélites, entonces la ubicación del punto puede ser determinada por triangulación. Una vez que el receptor obtiene las medidas de posicionamiento puede calcular las coordenadas de ubicación directamente en el dispositivo o enviar las medidas de vuelta al servidor de la red para que las procese. GPS brinda información en tres dimensiones: latitud, longitud y elevación [2].

GPS produce resultados precisos que se ubican dentro del rango de 5 a 40 metros de la ubicación real, de aquí que ha revolucionado los campos de la topografía y la navegación en tierra, aire y mar. Sin embargo, la necesidad de *línea de vista* se presenta como una limitación en el campo de los móviles, haciendo necesaria la incorporación de otra tecnología de posicionamiento típicamente basada en red. Otra limitante es el tiempo requerido para obtener la información de la ubicación.

2.5.2 Identificador de célula

Es un método que se basa en la red celular para determinar la posición de un dispositivo, mediante la evaluación del identificador de celda a la cual está conectado. Debido a que se utiliza la posición de la estación base (*Base Station Subsystem*) que da servicio al teléfono celular para determinar la ubicación del mismo, la técnica de localización es considerada burda y hasta cierto punto inexacta.

Un método más avanzado de posicionamiento de este tipo, determina la ubicación de un móvil evaluando primero el centro de gravedad del sector celular; para lograr esto es necesario determinar las distancias entre la estación base a la que está conectado y las estaciones vecinas [14].

El posicionamiento en sistemas celulares tiene varias ventajas entre las que se encuentra el corto tiempo de obtención de la primera posición, buena cobertura y bajo costo, debido al hecho de que la información proviene de la misma red celular. Sin embargo, también existen desventajas, como la precisión limitada (100 metros), que además depende de la densidad de las estaciones base, *e.g.* en zonas rurales la precisión decrece.

2.5.3 Ubicación a partir de una red IP

Este tipo de posicionamiento que puede realizarse en redes inalámbricas de área local (*Wireless Local Area Network*) ha surgido como una alternativa viable debido a que no depende de los nodos de la red celular y tampoco de comunicación con satélites, lo cual requiere línea de vista además de un mayor tiempo para obtener la primera posición.

Las WLANs tienen diferentes variantes especificadas en la serie de estándares IEEE 802.11. Una red de este tipo es hasta cierto punto similar a la red celular que comprende varias células, cada una atendida por una estación base. En la terminología de IEEE 802.11 la estación base se conoce como punto de acceso (*Access Point*) y el área que cubre se llama área de servicio básico (*Basic Service Area*). Las redes inalámbricas de esta familia se distinguen por el rango, la frecuencia, modulación, los tipos de señales, las velocidades de datos y el ancho de banda que utilicen. Las más usadas al día de hoy se basan en la especificación 802.11b y 802.11g y operan en el rango de frecuencia libre de 2.4 GHz, con un alcance que va de decenas a unos cuantos cientos de metros, dependiendo del entorno [15].

Debido a que las redes inalámbricas de este tipo están disponibles en una amplia cantidad de lugares públicos y edificios, y a que los fabricantes de dispositivos móviles han incluido la capacidad de conectarse de forma inalámbrica en los mismos, las WLANs se han vuelto muy atractivas para servir como base para determinar la ubicación de las personas.

Casi todos los sistemas de posicionamiento inalámbrico dependen de mediciones de la intensidad de la señal recibida (*Received Signal Strength*), la proporción entre señal recibida y ruido (*Signal-to-Noise Ratio*) o de detección de proximidad. Las indicaciones de RSS y SNR se basan en los llamados *beacons* emitidos durante el envío o recepción de datos. Al recibir los *beacons*, el receptor mide la RSS y SNR y la pone a disposición de las aplicaciones correspondientes dentro del dispositivo WLAN [15].

2.6 Servicios basados en la ubicación

Se denomina como LBS (*Location-based services*) a la entrega de datos y servicios de información cuyo contenido se ajusta a la ubicación actual o inmediata del usuario móvil [16]. Estos servicios conforman un sector de tecnología nuevo y de rápido crecimiento que incorpora sistemas de información geográfica, tecnología inalámbrica, sistemas de posicionamiento e interacción

móvil entre humano y computadora. Generalmente se considera que la información de la ubicación revela el conocimiento básico de la circunstancia y el ambiente del usuario que accede a los servicios.

Los servicios basados en la ubicación han proliferado debido a la infraestructura de comunicación inalámbrica y a las crecientes capacidades de los dispositivos móviles. Esto permite la creación de aplicaciones y servicios que van desde sistemas de guía y rastreo hasta publicidad específica en una ubicación. Sin embargo, mientras que los servicios basados en la ubicación que funcionan en exteriores han crecido a partir de la disponibilidad del GPS en dispositivos portátiles, los servicios ofrecidos en interiores no han podido mantener el mismo ritmo de avance. La razón para esto reside en la falta de una buena solución para el posicionamiento en interiores, en la cual sea posible lograr una alta precisión en la ubicación [17].

La triangulación de señales de radio, para determinar la ubicación en interiores, se ve afectada por la fluctuación de las señales ocasionada por el desplazamiento del usuario. Por lo tanto, se están explorando alternativas como el rastreo por medio de dispositivos de bajo costo basados en sensores, *e.g.* acelerómetros, giroscopios y magnetómetros. Estos dispositivos de tamaño reducido pueden medir distancias a partir de información sobre la aceleración, el ángulo de rotación y el cambio de dirección.

Recientemente, los requerimientos de la Internet han dado un nuevo alcance a los servicios basados en la ubicación. La enorme cantidad y rango de contenidos disponibles en la red, puede asimilarse sólo si existen mecanismos para clasificar la información de acuerdo a las necesidades del usuario. Esta ha sido la motivación para la evolución de los motores de búsqueda en Internet en los últimos años [18].

Así es como la creación de un mecanismo automático para determinar la ubicación del usuario, es necesaria para optimizar el acceso a ciertos servicios en Internet y evitar que el usuario tenga que especificar manualmente en dónde está ubicado. Los estándares más importantes de telefonía celular y las redes 3G definen una infraestructura de servicios de ubicación. Estas especificaciones han sido implementadas de forma universal, y la mayoría de las redes celulares poseen la habilidad de determinar la ubicación de teléfonos celulares de forma automática y entregar esta información a las aplicaciones.

2.7 Voz en IP y SIP

VoIP (*Voice over IP*) es una familia de tecnologías que permiten que aplicaciones de voz y telefonía sean sustentadas por una red IP (*Internet*

Protocol), como es el caso de Internet. Estas tecnologías incluyen protocolos, estándares de hardware y software y programas de computadora. También son conocidas con el término "Telefonía IP".

En su forma más simple, la telefonía IP permite realizar llamadas telefónicas a través de Internet, en lugar de utilizar la línea telefónica tradicional, lo cual evita la imposición de tarifas y regulaciones sobre la comunicación a larga distancia. Debido a la existencia de software para comunicación por Internet, *e.g.* Gizmo5 y Skype, es posible realizar llamadas a cualquier parte del mundo desde cualquier computadora.

La telefonía IP se está desarrollando como una tecnología emergente proponiéndose como el reemplazo de los sistemas telefónicos existentes basados en la conmutación de circuitos TDM (*Time Division Multiplex*). Asimismo, el Internet ha reemplazado las tecnologías basadas en TDM orientada a conexión, *e.g.* la Red Digital de Servicios Integrados (*Integrated Services Digital Network*) [1].

De este modo, las comunicaciones alámbricas e inalámbricas están migrando hacia los estándares de Internet desarrollados por la IETF (*Internet Engineering Task Force*). A pesar de que las redes de telecomunicación, que han llegado hasta nuestros días, son aún dominantes, ya se tiene noción del tiempo en el que serán reemplazadas por las redes IP (*Internet Protocol*).

Asterisk es un PBX (*Private Branch Exchange*) [19] de código abierto que permite crear un sistema telefónico personalizado. Un PBX puede considerarse como un tablero de conmutadores de telefonía privada que permite la conexión entre los teléfonos internos (extensiones) de una organización y entre la red telefónica externa y los teléfonos de la organización. Esto permite disminuir costos debido a que no es necesario utilizar una línea telefónica independiente para cada teléfono dentro de la organización [20].

El impacto de Internet ha provocado que las compañías de telecomunicaciones se interesen por nuevos modelos de negocio, que saquen ventaja de las tecnologías y protocolos de Internet, entre los cuales se encuentra el protocolo SIP (*Session Initiation Protocol*). Como ejemplos de estos modelos de negocio se encuentran los servicios de mensajería instantánea con voz sobre IP, PSTN sobre IP y telefonía por cable [1].

2.7.1 Protocolo de Inicio de Sesiones

El protocolo SIP forma parte de la segunda ola de protocolos de aplicación de Internet, la cual introduce la noción de comunicación interactiva de humano a humano, permitiendo la integración con cualquier tipo de medio, sin limitarse

únicamente a la voz. Es un protocolo simple y genérico para iniciar, modificar y terminar sesiones interactivas de comunicación a través de Internet [1].

La premisa de SIP es que cada extremo de una conexión es un punto o entidad y su propósito es únicamente ayudar a que estos dos puntos se comuniquen entre sí. Es un protocolo relativamente simple, con una sintaxis similar a la de otros protocolos familiares como HTTP (*Hypertext Transfer Protocol*) y SMTP *(Simple Mail Transfer Protocol)* [21].

El protocolo SIP permite el establecimiento de sesiones de comunicación a través de redes IP (*Internet Protocol*), detallando sus usos en la transmisión de VoIP (*Voice over IP*) y por consiguiente en la telefonía móvil IP. SIP puede ser transportado mediante los protocolos de transporte UDP (*User Datagram Protocol*) o TCP (*Transmission Control Protocol*).

SIP ha sido adoptado prácticamente por todos los proveedores de telefonía tanto para redes alámbricas como inalámbricas. A pesar de que H.323 era el principal competidor de SIP, pronto será reemplazado por éste último, ya que es utilizado para implementar sistemas punto a punto, servicios de telefonía residencial, sistemas de reemplazo de centrales telefónicas (PBX) y redes de próxima generación a gran escala [22].

Las ventajas del protocolo SIP radican en su amplia aceptación y flexibilidad de arquitectura. El uso de SIP para establecer sesiones de voz, video y datos coloca a la telefonía como una aplicación más en Internet, utilizando direccionamiento, tipos de datos, software, protocolos y seguridad similares. Por lo tanto, las redes independientes para voz ya no son necesarias.

SIP puede soportar movilidad a nivel de aplicación a través de diferentes redes. De esta forma, es posible que los usuarios se comuniquen mientras están en movimiento y mantengan la transferencia de archivos sin interrupción o la navegación constante por Internet, lo cual no es posible con direcciones IP que cambian [1].

El protocolo SIP redefine las comunicaciones y ha impactado la industria de la telecomunicación, de acuerdo con los proveedores de servicio de telecomunicaciones y los distribuidores de tecnologías de la información. Asimismo, este protocolo es utilizado cada vez más en servicios, aplicaciones de usuario y dispositivos como teléfonos, PCs, laptops y móviles [1]. La siguiente generación de teléfonos móviles utilizará SIP como tecnología de señalización [22].

2.7.2 Funcionamiento del protocolo SIP

Los dispositivos que incluyen soporte SIP son capaces de comunicarse entre sí de forma directa, sin embargo, también es posible implementar servidores SIP que

se encarguen de ofrecer una variedad de servicios adicionales y desempeñar funciones como el registro de usuarios SIP, autenticación, autorización y contabilidad (AAA), soporte de movilidad de usuario entre redes y dispositivos, etc.

Para que un agente de usuario se comunique mediante SIP, primero debe registrarse con un servidor SIP. Esto se logra mediante la petición `REGISTER`, en donde un usuario especifica los identificadores o URIs (*Uniform Resource Identifier*) para los cuales desea recibir notificaciones. Las comunicaciones IP utilizan estos identificadores para realizar un direccionamiento similar al del correo electrónico, y tienen la forma: *usuario@dominio*.

La petición posee encabezados del tipo `Contact` para incluir las URIs a registrar, de este modo SIP cuenta con soporte integrado de movilidad, considerado como uno de sus principales beneficios, ya que el usuario notifica regularmente sus identificadores y las peticiones hacia las mismas son dirigidas a la dirección IP del dispositivo SIP desde el cual se haya realizado la petición [1].

Un agente de usuario puede configurarse para que se registre automáticamente, a intervalos preestablecidos o cuando el usuario así lo desee.

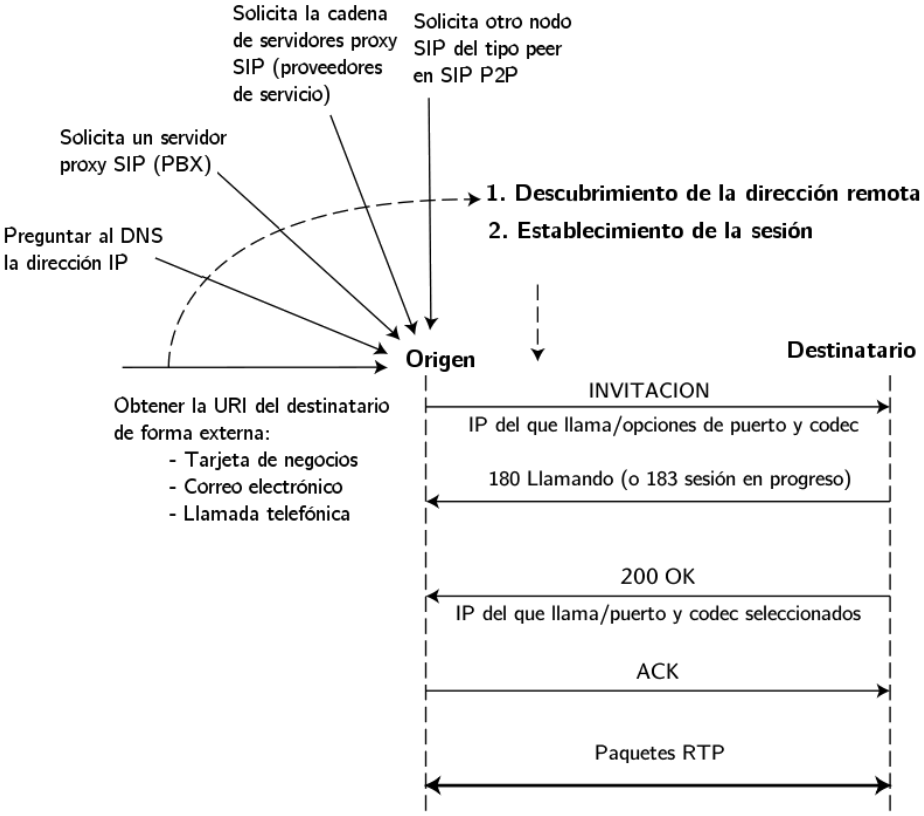


Figura 2.4: Funcionamiento básico de SIP

Una vez que el usuario está registrado, puede realizar el intercambio de mensajes SIP con el servidor o con otro cliente. El método `MESSAGE` simplemente transporta el cuerpo de un mensaje hacia una URI determinada. A continuación se muestra un ejemplo de paquete enviado mediante SIP, en donde el usuario denominado 'usuarioa' y que reside en el dominio 'aqui.com', envía a 'usuariob' del dominio 'alla.com' el mensaje 'Hola qué tal'. De este modo se observa el manejo de encabezados que realiza el protocolo, *e.g.* los campos `Via`, `To`, `From`, `Content-Type` y `Content-Length`.

```
MESSAGE sip:usuariob@alla.com SIP/2.0
Via: SIP/2.0/UDP pc.aqui.com;branch=z9hG4bK343g
To: User B <sip:usuariob@alla.com>
From: User A <sip:usuarioa@aqui.com>;tag=4541232ds
Max-Forwards: 70
Call-ID: a532431277432513
CSeq: 15 MESSAGE
Content-Type: text/plain
Content-Length: 15
Hola qué tal
```

De este modo se cuenta con una forma sencilla de enviar información desde los usuarios hacia el servidor SIP.

2.8 Comunicación oportunista

La movilidad de un sistema puede traducirse en pérdida de la conectividad de red debido a que el desplazamiento de una ubicación a otra implica barreras físicas que provocan que en algún momento habrá desconexión de la red. En especial en el caso de la conectividad inalámbrica, las condiciones físicas pueden afectar significativamente la calidad de servicio de la red [7]. Las aplicaciones móviles deben saber cómo seguirán operando incluso cuando están desconectadas de la red o mientras realizan el cambio entre una red y otra (*handover*), es decir, la aplicación debe saber cómo lidiar con la falta de una conexión confiable.

La creciente presencia del acceso a Internet de forma inalámbrica, a través de redes como Wi-Fi o 3G, permite un mayor desarrollo del paradigma del cómputo móvil. En este ambiente, la información y los servicios inteligentes estarán inmersos en el medio que nos rodea y grandes cantidades de información circularán con el objetivo de crear un ambiente inteligente y proactivo que mejorará significativamente la experiencia de las personas tanto en el trabajo como en el tiempo libre [23].

La disponibilidad de un conjunto de tecnologías inalámbricas ha motivado la investigación enfocada hacia la integración de redes heterogéneas, que permiten al usuario tener siempre la mejor conectividad en cualquier momento, lugar y con cualquier dispositivo [12].

La proliferación de dispositivos portátiles con mayor capacidad de procesamiento ha creado un nuevo entorno para las redes. Al contrario de las comunicaciones convencionales que dependen de infraestructura, estos dispositivos pueden utilizar el envío oportunista de datos a través de saltos entre cada dispositivo [24]. Este tipo de envío es un algoritmo distribuido tolerante a fallos particularmente útil en las redes *ad hoc* deficientes, en donde es difícil obtener información de la topología global debido a interrupciones frecuentes, mediante el cual los datos son transmitidos de forma aleatoria a un vecino, con base en la información de la red local [25].

En el ambiente oportunista, un dispositivo debe decidir si transmitirá o no un mensaje al momento que encuentra otro dispositivo con conectividad. El modo de seleccionar el siguiente salto hacia el destinatario final, de tal forma que se minimice el retraso y se maximice la tasa de éxito, aún es tema de discusión [24].

Las redes oportunistas son redes *ad hoc* móviles deficientes (multi-hop) caracterizadas por desconexiones prolongadas, separaciones y topologías impredecibles e inestables [26].

Las redes oportunistas móviles están caracterizadas por un corto diámetro, un dispositivo destinatario es alcanzable utilizando un pequeño número de relevos, bajo restricciones estrictas de retraso. Estas redes pueden verse como una clase de redes tolerantes a retrasos [24].

Los protocolos de transmisión de datos también deben explotar de forma oportunista cualquier contacto entre los nodos para acercar los datos a los posibles usuarios interesados. La movilidad de los usuarios debe ser aprovechada para superar las brechas de conectividad y transmitir los datos [25].

Es importante considerar que al utilizar la comunicación oportunista, habrá costos adicionales en el sistema, como las copias adicionales de mensajes y el incremento en el consumo de energía en los dispositivos móviles. Cuando el número de puntos de acceso en el sistema es pequeño, el uso de la comunicación oportunista es muy importante para el desempeño del sistema en términos de rendimiento [27].

En el ambiente móvil, para incrementar el uso de todo el sistema, es muy importante permitir el trabajo continuo a pesar de que exista o no conectividad. Por lo tanto se requiere sobre todo la disponibilidad de los datos antes que la consistencia de los mismos.

2.9 Alta disponibilidad

Conforme las redes de telecomunicaciones y el Internet se han integrado en la vida diaria, las personas se han vuelto más dependientes de ellos. Ya que dependemos de estas redes, es necesario que sean altamente confiables, es decir, se necesita que las redes funcionen todo el tiempo y sean permanentemente accesibles [28].

Este concepto también es conocido como resistencia a fallos o RAS (*reliability, availability, serviceability*) y se refiere a un sistema de multiprocesamiento que puede recuperarse rápidamente de un fallo. Pueden transcurrir uno o dos minutos en inactividad mientras se cambia de un sistema a otro, pero el procesamiento continúa.

Anteriormente, los componentes de hardware eran la principal fuente de fallos e interrupciones en el servicio. Sin embargo, en la actualidad estos se deben a fallos en la operación y el software. A mayor tamaño de un sistema, su disponibilidad es más crítica y por lo tanto es más difícil hacerlo altamente disponible; los grandes sistemas son la tecnología base de la actualidad.

La alta disponibilidad es distinta que la tolerancia a fallos, en donde los componentes redundantes son diseñados para el procesamiento continuo sin saltar un solo *heartbeat*.

El control de procesos y producción y las aplicaciones de procesamiento de transacciones son los principales consumidores de los sistemas altamente disponibles, debido a que no pueden detenerse por una interrupción en el sistema de cómputo. Entre los servicios más conocidos que poseen alta disponibilidad se encuentran:

- Tiempo en Internet
- Sistemas bancarios y bursátiles
- Portales de comercio electrónico
- Proveedores de VoIP
- Sistemas de control aéreo
- Sistemas industriales

2.10 Discusión

En este capítulo se describieron los conceptos tecnológicos a partir de los cuales se construyó la propuesta: la evolución de las comunicaciones inalámbricas, el

cómputo móvil y el cómputo ubicuo, los servicios basados en la ubicación, la comunicación oportunista y el protocolo de inicio de sesiones.

También se cubrieron temas muy importantes desde el punto de vista del desarrollador y que se deben considerar en el diseño e implementación de la propuesta, como son: los lenguajes de programación, las plataformas existentes en los dispositivos móviles y la funcionalidad de SIP.

Estos temas sirven como fundamento de nuestra propuesta, con la finalidad de justificar su viabilidad y detallar las características del mecanismo de localización propuesto. Por lo tanto, el siguiente paso es especificar la forma en que se construirá el mecanismo, utilizando la tecnología existente y los medios disponibles para llevar a cabo su desarrollo. En el capítulo 3 se cubre la especificación y diseño del sistema final.

Capítulo 3

Arquitectura de la propuesta

Antes de desarrollar cualquier solución de software, como es el caso de nuestra propuesta, es necesario diseñar todos y cada uno de los componentes que conformarán el sistema, con la finalidad de tomar decisiones con antelación en lo que respecta al tipo de arquitectura en la cual se basará la solución, la funcionalidad que debe ofrecer, así como los resultados esperados del software.

La especificación de la arquitectura se basa en el proceso de la ingeniería de software, que es una metodología que provee un modelo bien fundamentado de las mejores prácticas para desarrollar soluciones basadas en sistemas computacionales. Formalmente puede definirse a la ingeniería de software como una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, como es el caso de este capítulo, hasta el mantenimiento del software después de que se utiliza.

A partir de este capítulo, cada apartado de la tesis corresponderá a una fase del proceso de software, el cual está compuesto por cuatro actividades fundamentales: [29]

1. **Especificación**.- Se define tanto la funcionalidad del software como las restricciones que tiene su operación.
2. **Desarrollo**.- Consiste en implementar la solución que cumpla con las especificaciones de la propuesta.
3. **Validación**.- El software desarrollado debe validarse para asegurar que haga lo necesario.
4. **Evolución**.- El software debe evolucionar para cumplir con los cambios que se requieran.

3.1 Arquitectura del sistema

Como parte del diseño del sistema final, es necesario modelar su arquitectura como un conjunto de entidades y sus relaciones. Por lo general, los sistemas son diseñados por medio de una jerarquía, en el sentido de que incluyen otros sistemas interconectados, mejor conocidos como subsistemas, que pueden operar por sí solos de forma independiente.

Un diseño de software es una descripción de la estructura que se va a implementar y los datos que son parte del sistema, las interfaces entre los componentes del mismo y, algunas veces, los algoritmos utilizados [29].

En la fase de diseño se deben identificar los subsistemas y especificar su funcionalidad para posteriormente implementarlos en paralelo y por último conjuntarlos en el sistema final. Entre lo que se busca diseñar en esta etapa es la arquitectura del sistema y las estructuras de datos que se utilizarán durante la fase de implementación.

En el capítulo 2 se mencionó que la arquitectura de la aplicación móvil puede basarse en un cliente ligero, cliente inteligente o mensajería. Para el caso de esta propuesta, la arquitectura que mejor se apega a los requerimientos es la de cliente inteligente, debido a que la obtención, validación y envío de la información de la ubicación se realizará desde el dispositivo móvil. La misión del servidor será recibir esta información y almacenarla para su posterior uso. Asimismo, deberá aprovechar estos datos para localizar al dispositivo cuando así sea necesario.

De este modo, el productor de la información será el dispositivo móvil y el consumidor o receptor será el servidor. La comunicación conforma un sistema cliente-servidor, en el cual existen muchas entidades que enviarán simultáneamente información a un punto central de control y almacenamiento.

El estilo particular y la estructura elegida para una aplicación pueden depender de los requerimientos no funcionales a los que se les dé mayor prioridad [29]. Para nuestro caso, se le da mayor importancia al desempeño de la aplicación, por lo tanto las operaciones críticas deben quedar dentro de un número reducido de subsistemas con mínima comunicación entre sí (aplicación móvil). Otro aspecto importante es la seguridad del sistema, para lo cual se debe contar con mecanismos de autenticación en el servidor del sistema.

A partir del análisis de los requerimientos y de los distintos modelos mencionados en este capítulo, es posible abstraer la arquitectura general del sistema de localización. En la Figura 3.1, se muestra la arquitectura principal del sistema, con los dos subsistemas que conforman el mecanismo de localización: la aplicación móvil y los servidores del sistema. Cada uno de estos subsistemas se conforma de módulos, es decir, de componentes que suministran uno o más servicios entre sí y sin embargo no funcionan de forma independiente.

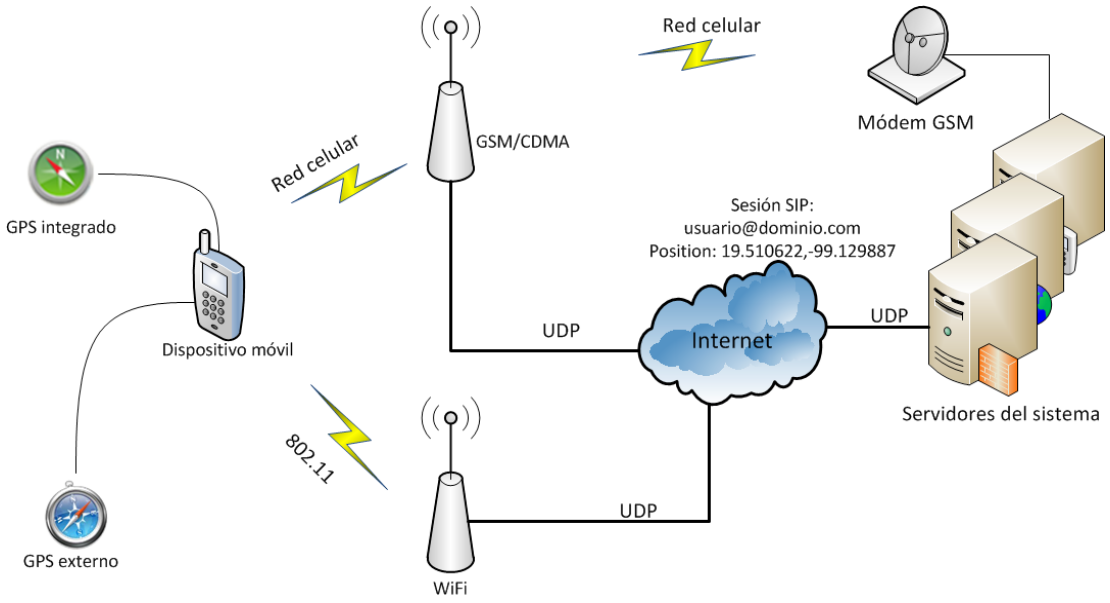


Figura 3.1: Arquitectura de la propuesta

3.1.1 Modelo cliente-servidor

El modelo arquitectónico cliente-servidor es un modelo de sistemas distribuido que muestra cómo los datos y el procesamiento se distribuyen a lo largo de varios procesadores. Los componentes principales de este modelo son:

1. Uno o varios servidores independientes que ofrecen servicios a otros subsistemas.
2. Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores. Por lo general, éstos son subsistemas y existen varias instancias de un programa cliente que se ejecuta de forma concurrente [29].
3. Una red que permite a los clientes acceder a estos servicios. En el caso de nuestra propuesta, esta es la red IP.

Comúnmente en este modelo de arquitectura, los clientes son los que conocen a los servidores y los servidores no requieren conocer la identidad de los clientes, sin embargo, para el caso en que se necesite localizar a un dispositivo móvil, es necesario que el servidor averigüe su ubicación por medio de los datos de "identidad" que le han sido proporcionados con anterioridad.

La ventaja más importante del modelo cliente-servidor es que es una arquitectura distribuida. Un sistema distribuido es aquel donde el procesamiento de la información está repartida entre varias computadoras más que confinada a

una sola máquina. Es fácil agregar un nuevo servidor e integrarlo con el resto del sistema y cada servidor debe tener responsabilidad de las actividades de administración de datos como la de respaldo y de recuperación [29].

3.2 Módulos del sistema

Después de diseñar la arquitectura, se debe descomponer cada subsistema en módulos. Durante la implementación se decidirá si estos módulos se ejecutarán en secuencia o en paralelo. La Figura 3.2 muestra los módulos que componen la aplicación móvil para nuestra propuesta.

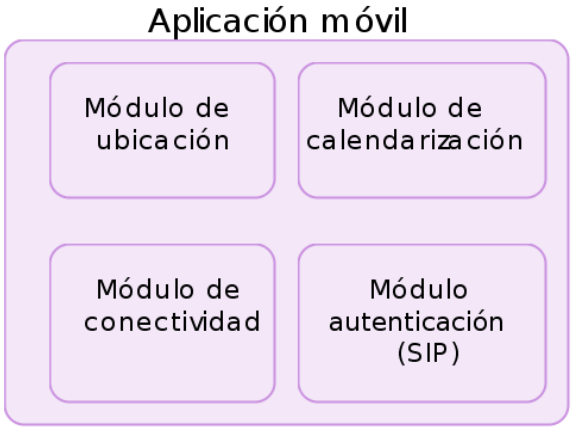


Figura 3.2: Módulos del cliente

A grandes rasgos, el módulo de ubicación se encarga de obtener la coordenada geográfica del dispositivo móvil, el módulo de calendarización contiene las preferencias del usuario en cuanto a la frecuencia del envío de reportes hacia el servidor. El módulo de conectividad se encarga de establecer la comunicación con el servidor, a través de redes con distintas características y el módulo SIP administra la identidad del usuario y su respectiva autenticación con el servidor, antes de enviarle cualquier información.

El subsistema servidor también se puede segmentar en módulos que ofrecerán distintos servicios al mecanismo global y a los clientes. Estos módulos se detallan en la Figura 3.3.

El módulo de localización trata de contactar al dispositivo móvil en alguna de las ubicaciones (direcciones IP de una red WiFi o celular) desde donde se ha reportado. El módulo SIP se encarga de la autenticación y el establecimiento de una sesión de comunicación entre el cliente y el servidor. El módulo de

autenticación y autorización permite que sólo los usuarios autorizados puedan solicitar que se localice a un dispositivo en específico o que accedan al historial de ubicaciones del mismo. Por último, el módulo de usuarios y dispositivos vincula a un usuario con los dispositivos desde los cuales ha reportado su ubicación.

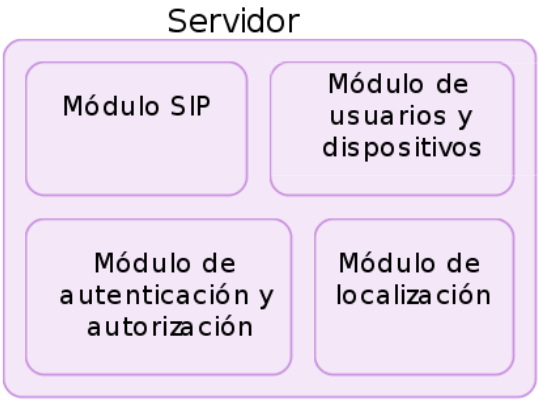


Figura 3.3: Módulos del servidor

3.2.1 Diagrama de clases

Por lo general, los sistemas distribuidos se desarrollan utilizando un enfoque orientado a objetos, debido a que están compuestos de partes independientes poco integradas, cada una de las cuales interactúa directamente con los usuarios o con las otras partes del sistema.

Un modelo orientado a objetos de una arquitectura de sistemas estructura el sistema en un conjunto de objetos débilmente acoplados con interfaces bien definidas. Los objetos llaman a los servicios ofrecidos por otros objetos y el hecho de estar débilmente acoplados permite que en la implementación se puedan modificar sin afectar a otros objetos [29].

La Figura 3.4 muestra el diagrama de clases básicas para la aplicación móvil. El usuario, que cuenta con un identificador o nombre y su contraseña para llevar a cabo la autenticación con el servidor, establece las preferencias con respecto a la ejecución de la aplicación. La información básica y más importante que se manipulará será la ubicación física del dispositivo móvil.

En el caso del servidor, sólo interactúan dos tipos de usuarios: los que reportan su ubicación desde un dispositivo móvil y los que acceden al mecanismo para rastrear algún dispositivo. En la Figura 3.5 se muestra la jerarquía de clases básicas del servidor. En la sección 3.6 se muestran más clases utilizadas por el servidor.

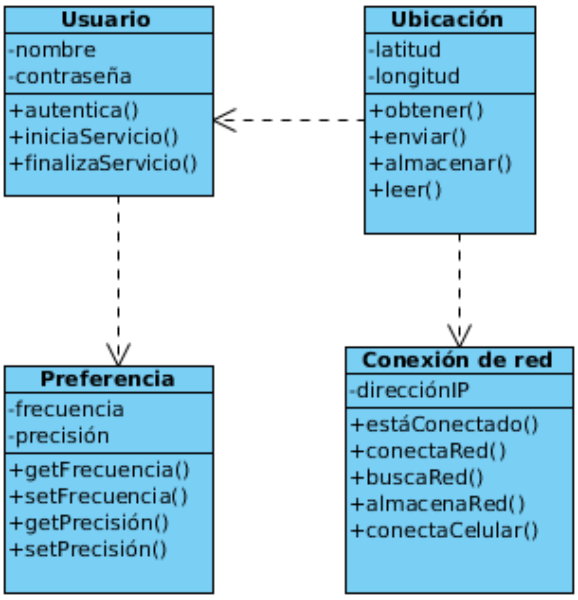


Figura 3.4: Diagrama de clases del cliente

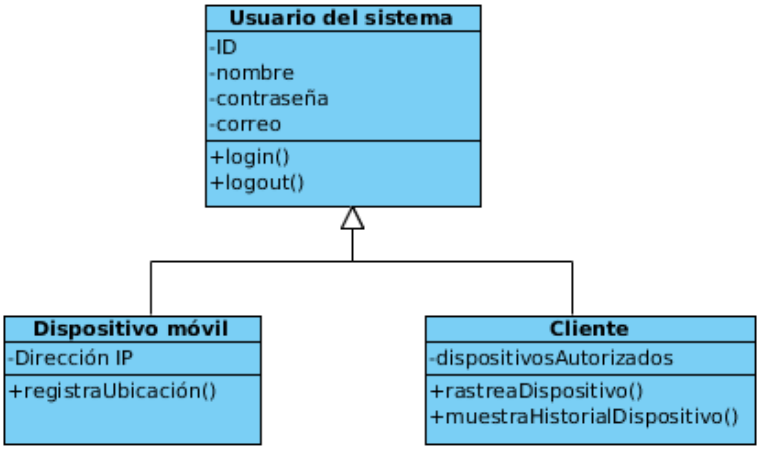


Figura 3.5: Jerarquía de clases en el servidor

3.3 Casos de uso

Entre las técnicas existentes para obtener y analizar los requerimientos de un sistema como el que se propone en este trabajo de tesis, se encuentran las orientadas a escenarios, es decir, que describen ejemplos de cómo se interactúa con el sistema deseado.

El enfoque más común para realizar la especificación de un sistema se basa en UML (Lenguaje Modelado Unificado), en el cual se utilizan modelos gráficos

como sistema de especificación. Los casos de uso se han convertido en una característica fundamental de la notación de UML, que se utiliza para describir modelos de sistemas orientados a objetos. En su forma más simple, un caso de uso identifica a los actores involucrados en una interacción y nombra el tipo de ésta [29].

En la Figura 3.6 se muestran los casos de uso para el usuario de un dispositivo móvil al que es posible localizar. Se denota como **Aplicación móvil** a la parte del mecanismo de localización que reside en el dispositivo utilizado por el usuario. Esta aplicación ofrecerá cuatro servicios a los que será posible acceder de forma manual o automática: posicionamiento, comunicación, calendarización y conectividad, en la siguiente sección se detallan cada uno de estos servicios.

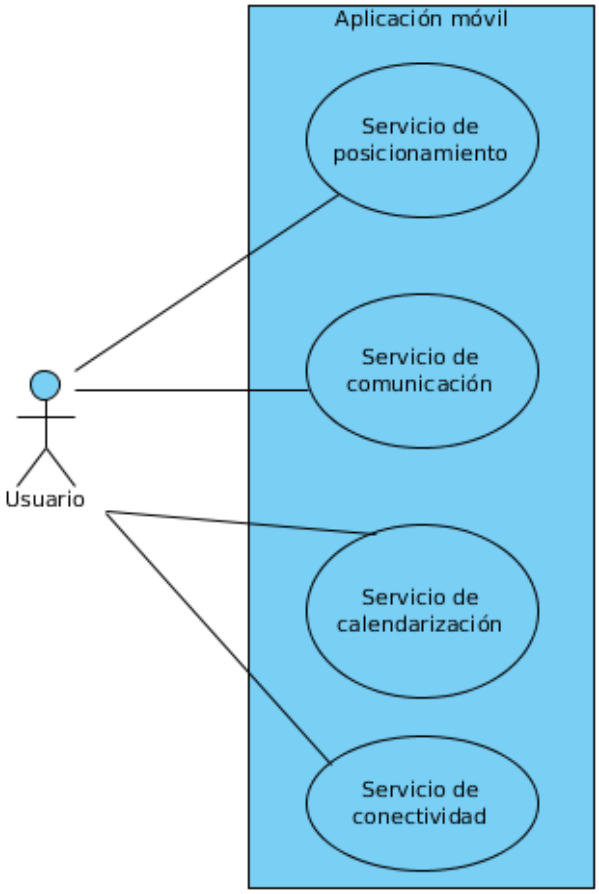


Figura 3.6: Casos de uso de la aplicación móvil

3.3.1 Casos de uso de la aplicación móvil

La funcionalidad que debe ofrecer la aplicación móvil consiste básicamente en la posibilidad de determinar la posición actual del dispositivo, o de no ser posible, recabar la información necesaria y enviarla al servidor para que éste último registre su ubicación aproximada.

Lo anterior se puede lograr a través de la interacción de los siguientes servicios:

- **Servicio de posicionamiento:** Este servicio consistirá en obtener las coordenadas de ubicación del dispositivo móvil, siempre y cuando cuente con línea de vista entre el receptor GPS del dispositivo y el sistema satelital. En caso contrario, se puede utilizar un mecanismo alterno, como la dirección de red en caso de estar vinculado con un punto de acceso *WiFi*, para aproximar la ubicación a partir de la dirección IP(*Internet Protocol*) de red [7].

- **Servicio de comunicación:** El mecanismo de localización espera recibir información de los usuarios registrados para poder mantener un historial de sus ubicaciones a lo largo del tiempo. Por lo tanto, debe contarse con un medio de comunicación fiable, a través del cual se envíen los datos de latitud y longitud y el identificador de usuario. Se propone el uso del protocolo SIP como medio de identificación debido a que permite la gestión de sesiones al soportar la movilidad de dispositivo y de usuario.

- **Servicio de calendarización:** Se desea que tanto la obtención de la ubicación física como su envío, a través de redes con diferentes características, sean procesos automáticos, es decir, que no requieran de la iniciativa del usuario ni interrumpan el uso que haga del dispositivo móvil. Es así como se determina calendarizar estas actividades para su ejecución periódica de acuerdo a los parámetros que determine el usuario, el cual puede decidir el intervalo entre cada proceso, desde unos minutos hasta horas, según sea la frecuencia con la que cambie de ubicación. Aquí es importante considerar el tiempo que lleva al dispositivo obtener la información de su ubicación, por lo tanto no es recomendable que el periodo de tiempo entre una ejecución y otra sea muy corto (menor a 5 minutos).

- **Servicio de conectividad:** Este servicio debe garantizar que haya conectividad de red siempre que sea necesario para enviar la información de la ubicación, ya sea a través de la red celular (GPRS, SMS) o de la red

inalámbrica *WiFi*. Además de dar preferencia a las redes inalámbricas sobre las celulares, por cuestión de costo y velocidad de transmisión, se utilizarán los conceptos de la comunicación oportunista para aprovechar cualquier medio de conectividad disponible para el dispositivo móvil [27].

3.3.2 Casos de uso del servidor

El servidor de localización tendrá la función de recibir los datos de ubicación de todos los dispositivos móviles registrados, y concentrarlos en una base de datos local. Además, debe ofrecer servicios a usuarios por medio de una interfaz Web.

Como se puede apreciar en la Figura 3.7, existen cuatro casos de uso posibles para el servidor:

- **Servicio de localización:** Debido a que se establecerán sesiones de usuario a través de una red IP (ya sea una WLAN o a través de la red celular), es posible mantener comunicación con el dispositivo móvil si aún se encuentra conectado desde la última ubicación que reportó, suponiendo que su dirección IP no ha cambiado. En caso de que ya no sea posible localizarlo en la misma ubicación de red, se debe solicitar al dispositivo móvil que reporte su ubicación utilizando un medio de comunicación alternativo, *e.g.* mediante un SMS Por cuestiones de confidencialidad, el usuario registrado en el sistema, debe autorizar a los usuarios que serán capaces de solicitar su ubicación por medio de este servicio.

- **Servicio de persistencia:** Mediante este servicio, el servidor concentra en una base de datos la información de la ubicación enviada desde los dispositivos móviles, para que así sea posible intentar rastrearlo en alguna de las ubicaciones reportadas.

- **Servicio de control de usuarios y dispositivos:** El servidor debe permitir que un usuario envíe la información de la ubicación desde uno o más dispositivos móviles, asociando los distintos equipos entre sí para permitir, además de la movilidad física y de red, la de dispositivo.

- **Servicio de bitácora:** El servidor almacenará todas las ubicaciones reportadas desde los teléfonos móviles, de este modo se construye un historial por cada usuario, el cual puede ser consultado con fines estadísticos o simplemente para analizar los lugares desde los cuales se han enviado los datos de la posición geográfica.

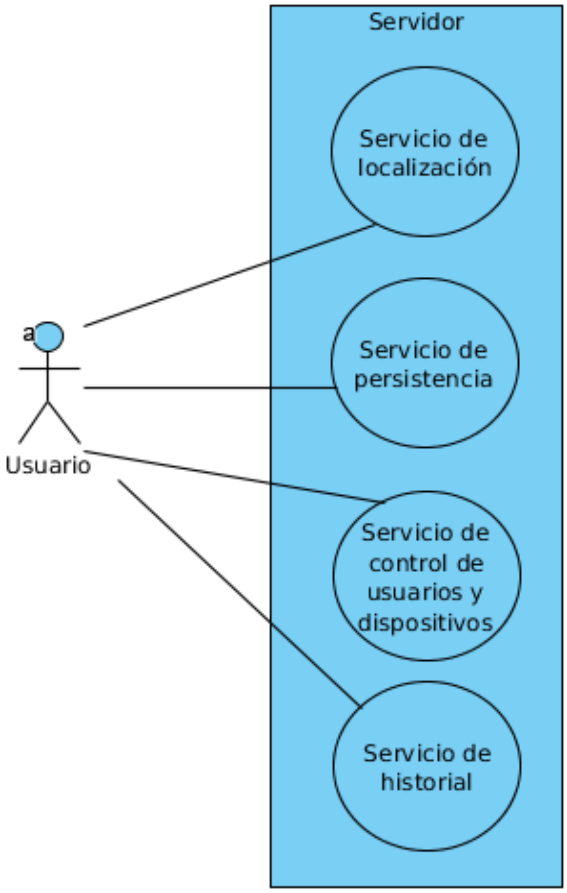


Figura 3.7: Casos de uso del servidor

3.4 Diagramas de secuencia

Dentro de UML, los diagramas de secuencia se utilizan para agregar información a un caso de uso. Estos diagramas muestran a los actores involucrados en la interacción, los objetos del sistema con los que se puede interactuar y las operaciones asociadas con estos objetos [29].

A partir de los casos de uso de la aplicación móvil, se muestra el diagrama de secuencia del servicio de comunicación en la Figura 3.8.

La comunicación entre la aplicación móvil y el servidor del sistema debe ser sencilla y transparente, con la finalidad de disminuir el tiempo de transmisión de datos y permitir que la ejecución cíclica del envío de las información de la ubicación se realice sin retrasos ni traslapes.

El primer paso en la comunicación será autenticar al usuario, de este modo se evita recibir información falsa o desde dispositivos no registrados. Una vez que se ha verificado la identidad del usuario, se espera que envíe los datos de la

ubicación para almacenarlos en el servidor. Asimismo, se notifica al usuario la respuesta a esta transacción.

Para el proceso de autenticación se plantea el uso del protocolo SIP, por lo tanto es importante conocer la estructura del protocolo para analizar si se necesita construir un tipo especial de paquete o se aprovecharán los ya existentes en la especificación de SIP [30]. La idea de utilizar este protocolo como auxiliar proviene del manejo transparente que hace de las sesiones de usuario, en especial en el entorno de la aplicaciones telefónicas, en las cuales también entra la telefonía móvil.

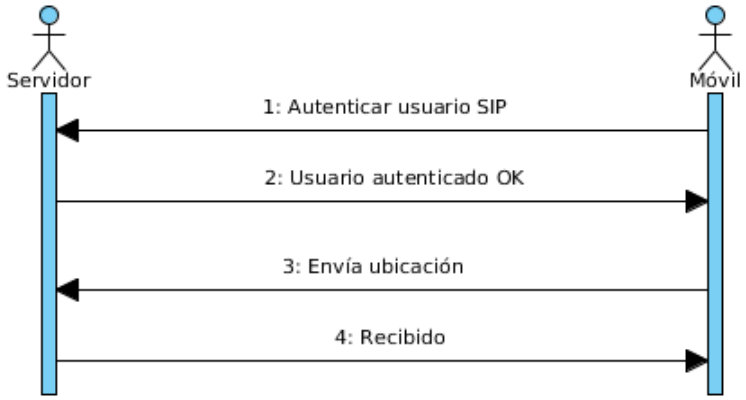


Figura 3.8: Secuencia de comunicación

Otro servicio a detallar dentro de los casos de uso de la aplicación móvil, es el de conectividad, en el cual se deberá desarrollar un algoritmo basado en la comunicación oportunista, para aprovechar cualquier punto de conexión a través del cual se puedan enviar los datos hacia el servidor.

Debido a que existirán distintos escenarios posibles para el caso en que el teléfono móvil busque un punto de conexión a Internet, *e.g.* el dispositivo se encuentra conectado a una red WiFi o logra conectarse al primer punto de acceso disponible, a continuación (Figura 3.9) se detalla el escenario más general, en el cual el teléfono no logra la conectividad con ningún punto de acceso *WiFi* y por lo tanto debe tratar de conectarse por medio de la red celular.

En este caso se deberá dar prioridad a las redes inalámbricas (IEEE 802.11) debido a que no representan un costo para el usuario, además de que ofrecen mayor fiabilidad en el establecimiento de la conexión y mayores velocidades de transmisión de datos.

El primer paso a realizar, antes de poder enviar los datos de la ubicación hacia el servidor, es verificar si existen datos de redes inalámbricas configuradas en el dispositivo móvil, esto debido a que comúnmente el usuario se conecta en lugares

de confianza como la casa o el trabajo, o incluso en lugares públicos como aeropuertos, centros comerciales y restaurantes. Mientras existan datos para conectarse a estas redes, como el SSID (*Service Set Identifier*) [4] de la red y la contraseña de acceso (en caso de contar con algún tipo de seguridad), se intentará la conexión con cada red existente. Si no es posible establecer la conexión, entonces es necesario buscar redes inalámbricas en la cercanía, lo cual es una actividad que consume tiempo y energía en el dispositivo móvil. Para evitar la desmedida utilización de estos recursos, la búsqueda de redes inalámbricas sólo se realiza una vez por cada intento de comunicación hacia el servidor.

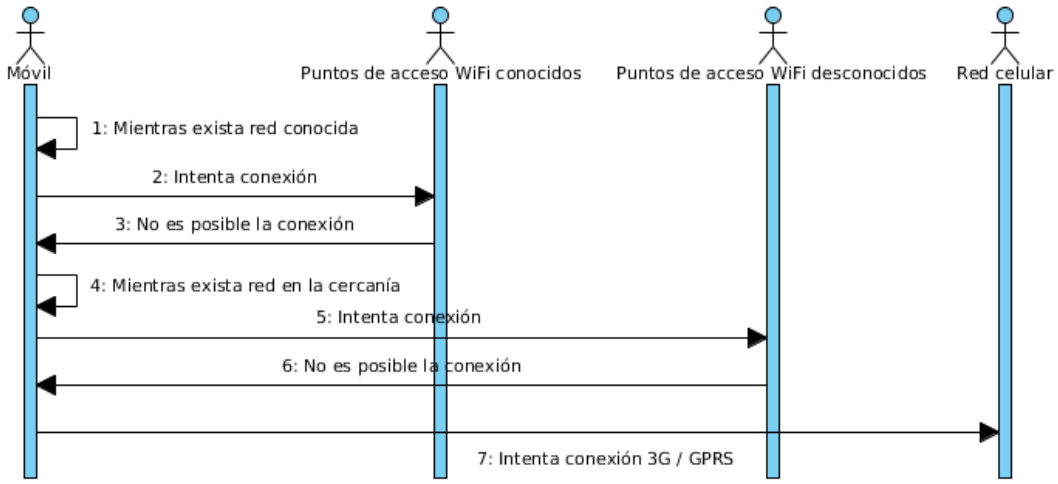


Figura 3.9: Secuencia de búsqueda de conectividad

En caso de que no se obtenga conectividad con las redes cercanas, ya sea porque no las hay o porque cuentan con protección con contraseña, entonces se debe proceder a enviar los datos a través de la red móvil, ya sea 3G o GPRS. Cabe destacar que actualmente muchos dispositivos poseen las configuraciones necesarias para acceder al servicio celular de datos de manera automática.

3.4.1 Diagramas de secuencia del servidor

Recordemos que para el caso del servidor, el usuario podrá interactuar con dos servicios: el de localización de usuario y el de historial. Para el primer caso se tiene en la Figura 3.10 la secuencia de acciones para localizar a un dispositivo móvil. El primer paso es autenticar al usuario, para conocer los dispositivos móviles que está autorizado a localizar; es importante señalar que cada usuario registrado tiene la capacidad de rastrear sus propios dispositivos, lo cual puede

aprovecharse en caso de extravío del dispositivo.

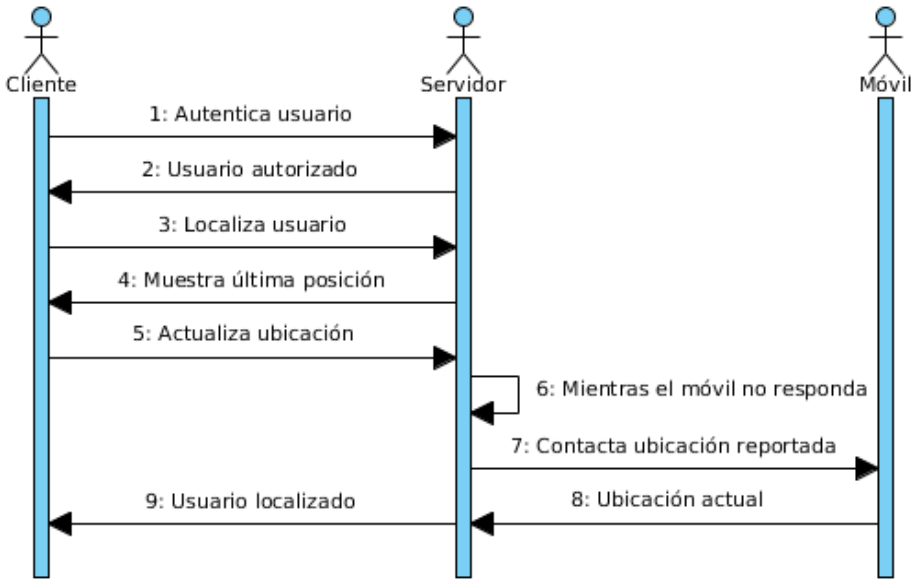


Figura 3.10: Secuencia de localización

Como primer recurso se sugiere mostrar el último reporte recibido desde el dispositivo móvil, y en caso de que esta información no sea útil al usuario, ya sea porque es obsoleta o no provee la información suficiente para conocer la ubicación del móvil, se solicitará al servidor que lo localice nuevamente.

Para realizar el rastreo, el servidor utilizará la última información recibida desde el dispositivo móvil. Debido a que la transmisión de la información se realiza a través de redes IP, sin importar si proviene de redes WiFi o redes móviles, es posible volver a contactar al usuario a través de la URI del protocolo SIP, utilizado para el establecimiento de sesiones entre el móvil y el servidor.

La URI provee un identificador único y válido de forma global para cada usuario. Los esquemas basados en URIs están asociados a varios protocolos y servicios como: FTP (*File Transfer Protocol*), HTTP, correo electrónico y Telnet. [1]

Dentro de los mensajes SIP, las URIs se utilizan para indicar el autor (`From`) y el destinatario final (`To`). Cada vez que el dispositivo móvil envíe información de su ubicación hacia el servidor, el dominio de su URI corresponde a la dirección IP del servidor de localización. El servidor SIP almacenará la dirección desde la cual recibe la petición y es así como posteriormente trata de localizar al dispositivo móvil en alguna dirección IP recibida.

La resolución de direcciones es una de las funciones más importantes del

protocolo SIP, al principio se conoce sólo una URI y al final se obtiene un nombre de usuario y una dirección IP. Esta resolución de un nombre general hacia un usuario y un host, es muy poderosa debido a que varios tipos de movilidad y portabilidad son implementados de forma automática. La resolución de direcciones puede realizarse tanto por agentes de usuario como por servidores [1].

3.5 Modelo de la base de datos del servidor

Una parte importante del modelado de sistemas es definir la forma lógica de los datos procesados por el sistema. La técnica más utilizada es la de Entidad-Relación, que muestra las entidades de datos (nodos) y las relaciones entre ellas (vínculos). Estos modelos se han utilizado ampliamente en el diseño de bases de datos [29]. La Figura 3.11 muestra el modelo de entidad-relación para diseñar la base de datos que reside en el servidor.

Para el servidor, un usuario será todo aquel que utilice uno o varios dispositivos móviles para reportar su ubicación o aquel que desee rastrear algún dispositivo en específico. Este usuario puede estar autorizado para localizar a varios móviles, así como es capaz de consultar la cantidad de reportes de ubicación que desee.

Para representar las entidades del mundo real que manipulan el sistema, es posible utilizar un enfoque orientado a objetos. En el paradigma orientado a objetos, una clase es una abstracción sobre el conjunto de atributos comunes que identifican los servicios u operaciones que provee cada objeto. Los objetos son instancias de dicha clase, de tal forma que se pueden crear diferentes objetos a partir de una clase [29].

Las clases se organizan en una jerarquía, que es un esquema de clasificación que muestra la manera en que se relacionan entre sí, a través de atributos y servicios comunes. Para desplegar esta clasificación, las clases se organizan en una jerarquía de herencia en la que las clases de objetos más generales se encuentran en la parte alta de la jerarquía. Los objetos más especializados heredan sus atributos y servicios.

La herencia se representa con una flecha que apunta de la clase que hereda atributos y operaciones a la superclase. La Figura 3.12 muestra la jerarquía de las ubicaciones que se pueden reportar desde los dispositivos móviles, respectivamente.

Por medio de los modelos de sistemas, se ha especificado el flujo de datos y las entidades involucradas en el funcionamiento del mecanismo de localización. Una vez especificado esto, es posible planear la estructura general del sistema a través del diseño de su arquitectura.

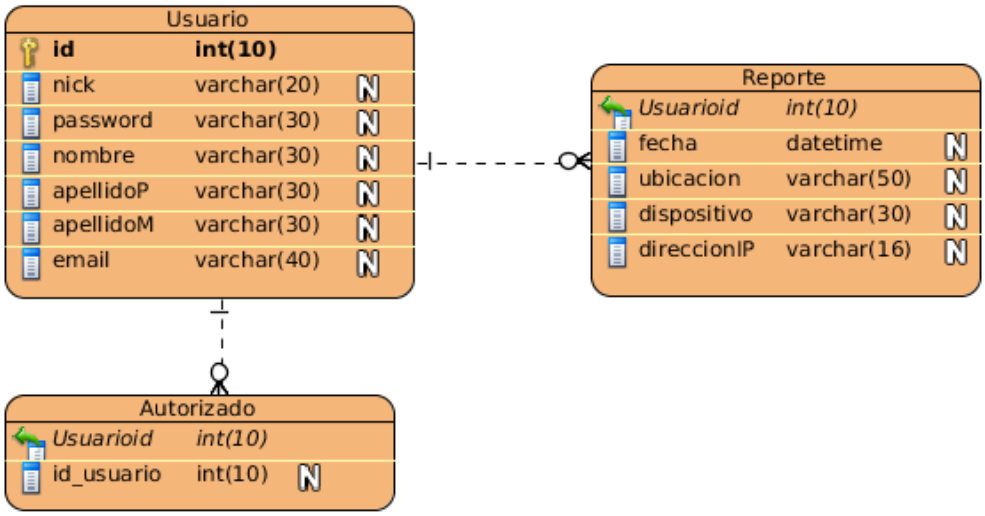


Figura 3.11: Modelo entidad-relación

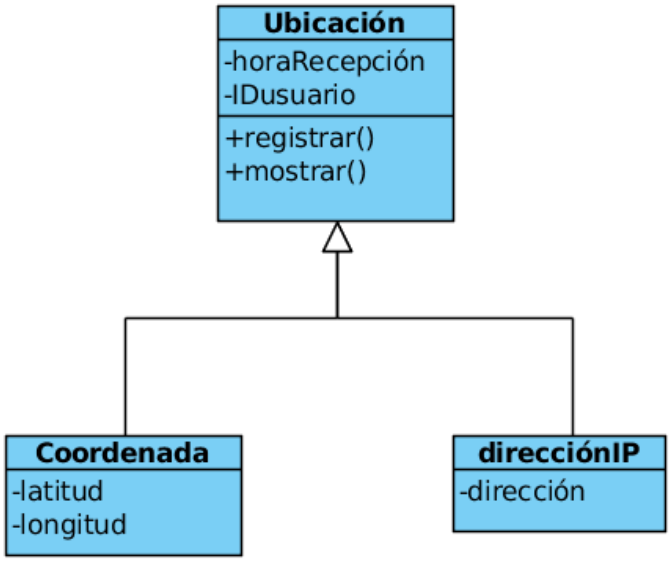


Figura 3.12: Jerarquía de clases de ubicaciones

3.6 Discusión

La metodología llevada a cabo para conformar nuestra propuesta de tesis se basa en el proceso de ingeniería de software, cuyas etapas principales son: especificación, desarrollo, validación y evolución. En este capítulo se cubrió la etapa de especificación del la propuesta, la cual consiste en el análisis de los requerimientos y el diseño de la arquitectura de la solución.

El mecanismo de localización se basa en una arquitectura cliente-servidor, donde los clientes son dispositivos móviles que poseen una aplicación "inteligente" que ejecuta tareas variadas. Las tareas del servidor son: atender peticiones, recibir y almacenar la información de la ubicación y permitir el rastreo de los dispositivos.

Tanto la parte cliente como servidor, se dividen en módulos que proveen una funcionalidad específica; el cliente hace uso de los servicios provistos por el servidor. Se utilizaron diagramas UML para modelar servicios, interacción de componentes y entidades del sistema.

Capítulo 4

Implementación

Después de haber detallado la arquitectura de nuestra propuesta, es posible proceder a la instalación y configuración de herramientas, así como a la codificación de los módulos que conforman al mecanismo de localización. Debido a que la transmisión de la información sigue el modelo cliente-servidor, es necesario implementar estas dos entidades por separado. La parte "cliente" del mecanismo de localización se compone de los dispositivos móviles; la parte del servidor debe ser un dispositivo capaz de atender las peticiones de cada cliente de forma continua a través de Internet. Prácticamente cualquier computadora personal puede cumplir con la función de servidor.

En esta sección se describirá la infraestructura con la que se cuenta para implementar la propuesta, así como las diferentes opciones tecnológicas existentes para desarrollar una aplicación móvil. Como se mencionó en el capítulo 2, se utilizará el PBX Asterisk como servidor que administre las sesiones de usuario a través del protocolo SIP.

4.1 Infraestructura de desarrollo

Los dispositivos con los que se cuenta para implementar la tesis son los siguientes:

- PC Intel(R) Core i7 de 4 núcleos a 2.8 GHz cada uno, RAM de 4 GB, S.O. Ubuntu Linux 9.10
- PC Intel(R) Core i7 de 4 núcleos a 2.8 GHz cada uno, RAM de 4 GB, S.O. Fedora Linux 12

- Ruteador inalámbrico WRT54G Linksys.
- Gateway VoIP – GSM Ansel 7005.
- Teléfono móvil Nokia N95, S.O. Symbian v9.2, con soporte para conectividad WLAN 802.11 b/g y GPS integrado.
- Teléfono móvil HTC Wildfire, S.O. Android 2.2, con soporte para conectividad WLAN 802.11 b/g y GPS integrado.

La computadora con el sistema operativo Ubuntu será utilizada como servidor del mecanismo de localización, por lo tanto, es necesario que se pueda acceder a sus servicios a través de Internet. Esto lo logramos a través de una dirección IP homologada (148.247.102.22) provista por el Cinvestav. La otra computadora se utilizará para el desarrollo de las aplicaciones necesarias.

El ruteador posee conectividad a Internet a través de la red del departamento de Computación del Cinvestav y a su vez provee la conexión a Internet de forma inalámbrica a los demás dispositivos.

Debido a que es de vital importancia contar con un dispositivo de posicionamiento global, el desarrollo móvil se apegará a los dos *smartphones* con los que se cuenta. Este aspecto influirá en la decisión sobre qué plataforma de desarrollo utilizar.

4.2 Implementación del servidor

Es necesario que el servidor del sistema permita que cada cliente establezca una sesión de comunicación utilizando el protocolo SIP. Uno de los PBX más populares para la transmisión de voz sobre IP es Asterisk; se decidió utilizar esta herramienta debido a que es software libre y provee la funcionalidad requerida para nuestro propósito, como es la flexibilidad en su configuración y el soporte del protocolo SIP.

4.2.1 Instalación y configuración de Asterisk

La principal función de Asterisk en nuestro sistema será registrar los dispositivos móviles que se comunican con el servidor. A partir de la guía de instalación y configuración de Asterisk [21], se puede tener esta herramienta funcionando siguiendo los siguientes pasos:

1. Instalar las dependencias necesarias para compilar e instalar Asterisk, entre ellas se encuentra el compilador *gcc*, la herramienta *make*, la biblioteca *ncurses* y los encabezados del kernel del sistema (Linux).
2. Descargar y descomprimir el código fuente de Asterisk, *libpri* (bibliotecas) y *zaptel* (controladores para tarjetas telefónicas). Se recomienda realizarlo en la carpeta */usr/src*.
3. Configurar los módulos de Asterisk que se desean instalar, por medio de la herramienta *Menuselect*.
4. Compilar e instalar *zaptel, libpri* y Asterisk (en ese orden).
5. Iniciar el servicio de *zaptel* y Asterisk, respectivamente.
6. Modificar los archivos de configuración *extensions.conf* (plan de marcado) y *sip.conf* (configuración SIP). Estos archivos están en la carpeta `/etc/asterisk`, después de realizar la instalación.

Una vez que el servicio de Asterisk se está ejecutando en el servidor, es posible administrar y detectar los eventos que ocurren, a través de la consola de comandos. Asterisk cuenta con distintos archivos de configuración, que es posible modificar y personalizar de acuerdo a nuestras necesidades, sin embargo, para los propósitos de este trabajo de tesis únicamente modificaremos los que se relacionan con el protocolo SIP y el plan de marcado, con la finalidad de comprobar la correcta comunicación entre los dispositivos móviles.

En el archivo *sip.conf* se registran los usuarios que serán los clientes o agentes de usuario (*user agents*) de Asterisk a través del protocolo de inicio de sesiones. Es necesario que los usuarios, que deseen utilizar el mecanismo de localización para sus dispositivos móviles, se registren en este archivo, de otro modo no será posible establecer la comunicación con el servidor. En la Figura 4.1 tenemos el ejemplo de los datos de un dispositivo dentro del archivo *sip.conf*. El nombre de usuario se coloca entre corchetes, la contraseña para el registro se especifica con la etiqueta `secret`.

```
[spa]
secret=welcome
type=friend
context=phones
host=dynamic
```

Figura 4.1: Datos para registrar un agente SIP

Existen tres tipos de usuarios en Asterisk: *user, peer y friend*. El último tiene permitido tanto hacer como recibir llamadas. La opción `host` sirve para indicar la

ubicación del cliente dentro de una red, puede ser una dirección IP específica o dinámica (`dynamic`). Cabe destacar que esta opción permite que Asterisk reciba un paquete REGISTER desde el dispositivo móvil junto con la dirección IP que está utilizando [21].

La opción `context` indica la sección del plan de marcado en donde se procesarán las llamadas de este dispositivo, el plan de marcado se almacena en el archivo *extensions.conf*. En la Figura 4.2 se muestra el contexto `phones`, en el que se crea una extensión para cada agente de usuario Asterisk, por ejemplo, al marcar la extensión 200 o 400, la llamada se establece por medio del protocolo SIP hacia el dispositivo (móvil o *softphone*) registrado por el usuario especificado.

```
[phones]
; Extensión 100: teléfono IP
exten => 100,1,Dial(${IP_PHONE},30)
exten => 100,2,SendText(hola)
exten => 100,n,Hangup()

; Extensión 300: teléfono analógico conectado a FXS
exten => 300,1,Dial(${OLD_PHONE},10)

; Extensión 200: john@148.247.102.22
exten => 200,1,Dial(SIP/john,30)
exten => 200,n,Hangup()

; Extensión 400: paul@148.247.102.22
exten => 400,1,Dial(SIP/paul,30)
exten => 400,n,Hangup()
```

Figura 4.2: Un contexto dentro del plan de marcado

Asimismo, el dispositivo móvil debe contar con su identificador de usuario (nombre) y su contraseña, para llevar a cabo la autenticación con Asterisk. En un ambiente de producción sería necesario contar con un dominio específico para conectarse con el servidor, sin embargo, en nuestro caso es suficiente con que el cliente conozca la dirección IP homologada que se mencionó anteriormente y el puerto en el cual Asterisk recibe peticiones (por defecto es el 5060). Estos datos se pueden incluir en el código fuente de la aplicación móvil y los datos de usuario y contraseña pueden ser provistos por el usuario del dispositivo móvil (Ver Figura 4.3).

Cada vez que el cliente se registra con el servidor, a través de SIP, aparece un mensaje en la línea de comandos de Asterisk, notificando la dirección IP desde la que se conectó el cliente y el dispositivo desde el cual se realizó la conexión (Ver Figura 4.4). Esto es de vital importancia para nuestra aplicación debido a que de

esta forma es posible almacenar las direcciones desde las cuales se reporta el cliente, así como los dispositivos que utiliza. Recordemos que el protocolo SIP permite movilidad de ubicación (es posible que el mismo usuario se conecte desde distintas redes) y movilidad de dispositivo (el usuario puede conectarse desde distintos dispositivos).

Figura 4.3: Datos SIP en el dispositivo móvil

```
labsd-desktop*CLI>
    -- Registered SIP 'john' at 189.217.3.16:5060
       > Saved useragent "Twinkle/1.4.2" for peer john
labsd-desktop*CLI>
```

Figura 4.4: Registro de un cliente SIP en Asterisk

De este modo queda configurado el servidor Asterisk para registrar a todos los agentes de usuario que estén dados de alta en el archivo *sip.conf*.

4.2.2 Implementación de la base de datos

En el capítulo 3 se diseñó la estructura general de la base de datos en la cual se almacenarán la información que fluye en el sistema. Este diseño se llevó a cabo con una herramienta de ingeniería de software asistida por computadora, o mejor conocida como CASE (*Computer Aided Software Engineering*), llamada *Visual Paradigm*. Esta herramienta facilita las etapas del proceso de software mediante el uso de diagramas UML (*Unified Modeling Language*) y la integración con herramientas de desarrollo, con la finalidad de automatizar tareas repetitivas.

De este modo es posible generar o actualizar la base de datos a partir del diagrama de Entidad-Relación (Figura 3.11), especificando parámetros como: ruta de almacenamiento, sistema de gestión de bases de datos (*Database Management System*) y su respectivo controlador, dirección IP del servidor en donde reside la base de datos y el nombre de esta última. En la Figura 4.5 se muestra la configuración necesaria para nuestro caso.

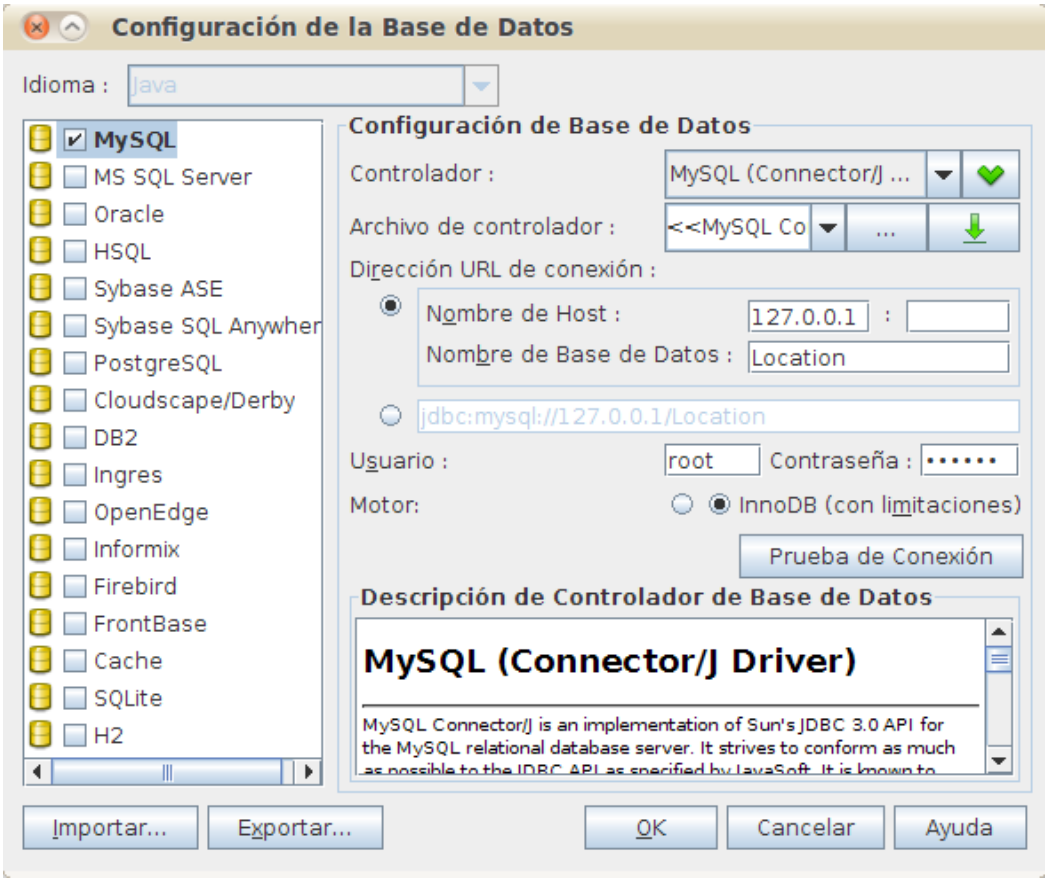


Figura 4.5: Configuración para generar la base de datos

Como se puede apreciar, existe una lista considerable de sistemas de gestión de bases de datos, cada uno con sus propiedades específicas y enfocados hacia determinados sectores de usuarios. Se optó por un gestor de código abierto MySQL, lo cual nos exime del pago de licencias o de la adquisición de productos de determinado proveedor (como es el caso de SQL Server) y ofrece una muy buena integración con sistemas Linux. Para integrarlo al servidor, es necesario instalar el programa `mysql-server` junto con todas sus dependencias, y con eso el sistema está listo para contener la base de datos diseñada previamente.

Después de haber generado la base de datos con la herramienta CASE, se comprueba su existencia mediante la consola de comandos, como se muestra en la Figura 4.6.

```
juank@juank-lap:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.1.41-3ubuntu12.10 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| Location           |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)

mysql> use Location;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_Location |
+--------------------+
| Autorizado         |
| Reporte            |
| Usuario            |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

Figura 4.6: Sistema de gestión de la base de datos

4.2.3 Interconexión de Asterisk con la base de datos

Debido a que se cuenta con el código fuente de Asterisk, es posible analizar cualquier detalle de su funcionamiento y de las operaciones que realiza durante su ejecución. El archivo *chan_sip.c* contiene la implementación completa de SIP para Asterisk y entre sus funciones se encuentra *parse_register_contact()*, la cual se encarga de analizar los datos de una petición de registro enviada desde un *peer* hacia el servidor Asterisk. Esta función recibe la estructura que describe a la solicitud y la que describe al *peer* o usuario, de este modo tenemos acceso a la información enviada desde los dispositivos móviles.

El siguiente código en lenguaje C fue insertado en la función *parse_register_contact()*, con la finalidad de identificar al usuario y almacenar su reporte de ubicación en la base de datos. Por identificación del usuario nos referimos únicamente a su localización dentro de la base de datos, ya que Asterisk se encarga de la autenticación mediante el nombre de usuario SIP y su respectiva contraseña. Obsérvese que sólo se almacena una nueva ubicación cuando se realiza la conexión desde una nueva IP, esto con el fin de evitar que se almacene la misma ubicación en repetidas ocasiones o cuando el usuario es huésped de la misma red, esto puede ocurrir por ejemplo cuando está conectado a la red del trabajo.

```
/* Si la petición SIP proviene de una IP no registrada y contiene un mensaje (no sólo
    encabezados) */
if (VERBOSITY_ATLEAST(2) && ast_sockaddr_cmp(&peer->addr, &oldsin)&&req->lines){

    char coordinate[1400],command[128],id[256]="1",query[256];
    FILE *fpipe;

    // Query para obtener el identificador del usuario
    sprintf(command,"mysql -u user -ppass -e 'SELECT id FROM Usuario WHERE nick=\"%s\"'
           Location",peer->name);

    // Ejecuta query y almacena su resultado
    if ( !(fpipe = (FILE*)popen(command,"r")) )
           perror("Problems with pipe");
    while ( fgets( id, sizeof id,fpipe));
    pclose(fpipe);

    time_t     now;
    struct tm  *ts;
    char date[32];

    // Obtiene la fecha y hora actual
    now = time(0);

    // Da formato a la fecha y hora, para ser almacenadas en la base de datos
    ts = localtime(&now);
```

```
        strftime(date, sizeof date, "%Y-%m-%d %H:%M:%S", ts);

        // Obtiene la coordenada enviada desde el dispositivo
        get_msg_text(coordinate, sizeof(coordinate), req, FALSE);

       // Query para insertar los datos de la ubicación en la base de datos
        sprintf(query,"-e INSERT INTO Reporte
        (Usuarioid,fecha,ubicacion,dispositivo,direccionIP) VALUES
        (%s,'%s','%s','%s','%s');",id,date,coordinate,get_header(req, "User-
        Agent"),ast_sockaddr_stringify(&peer->addr));

        char *argumentos[]={"/usr/bin/mysql","-u user","-ppass",query,"Location",
                (char *)0};
        int pid, rtn;

        // Se bifurca el programa ya que la llamada exec no es reentrante
        pid=fork();
        if (pid==0) { // Proceso hijo que ejecuta la actualización a la BD
                rtn=execvp("mysql",argumentos);
                if (rtn==-1) perror("Error en execvp");
        }
        // Espera a que el proceso hijo culmine su ejecución
        waitpid(pid,(int *)0,0);
}
```

Después de insertar estas instrucciones, es necesario volver a compilar el código fuente de Asterisk (el módulo *zaptel* y las librerías *libpri* no se han modificado). Esto se logra mediante la ejecución de las siguientes instrucciones (como super-usuario o *root*), en el orden especificado y en la carpeta que contiene el código (generalmente */usr/src/asterisk-1.8.0*):

1. `make distclean`
2. `./configure`
3. `make install`
4. `service restart asterisk`

De este modo se reinicia el servidor Asterisk y se le agrega la funcionalidad necesaria para nuestra aplicación. Para el caso del dispositivo Android, el cual no cuenta con soporte SIP, es necesario implementar un mecanismo alternativo para recibir los datos en el servidor, esto se detalla en la siguiente sección.

4.2.4 Implementación del servidor HTTP

Se eligió la plataforma Java debido a que ofrece las herramientas necesarias para implementar un servidor de acuerdo a las necesidades de esta aplicación. Java ofrece el componente web conocido como *Servlet*, el cual está diseñado para proveer contenido dinámico a las solicitudes de los clientes a través de la red [31].

Un *Servlet* se ejecuta dentro de un entorno de ejecución provisto por un contenedor Web, como lo es *Apache Tomcat*.

Al igual que con *Asterisk*, el primer paso es instalar las dependencias del contenedor *Apache Tomcat*, *e.g.* el compilador *gcc* y el kit de desarrollo para Java (JDK).

Para programar en el lenguaje Java se utilizó un entorno integrado de desarrollo (*NetBeans*), el cual permite la integración con *Tomcat* de forma transparente, de tal forma que al compilar y ejecutar el código, este queda implementado a través del servidor *Tomcat*.

Es necesario indicarle al servidor qué controlador utilizar para la base de datos. El controlador de MySQL para la plataforma Java se conoce como Connector/J y debe colocarse en la carpeta de librerías del servidor (*/usr/share/Tomcat6/lib* en nuestro caso). Posteriormente, es necesario realizar la configuración de los recursos y la conexión a la base de datos, por medio de archivos XML.

A continuación se muestra la estructura principal del *Servlet* que recibe las peticiones de los clientes en el servidor:

```
public class LocationServlet extends HttpServlet {

    // Procesa las peticiones HTTP, tanto del método GET como POST.
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/plain;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {

            // Obtiene el contexto del entorno de nomenclatura
            Context initCtx = new InitialContext();
            Context envCtx = (Context) initCtx.lookup("java:comp/env");

            // Busca la fuente de información (base de datos)
            DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");

           // Inicia una nueva conexión a la base de datos
            Connection con = ds.getConnection();

           // Query para obtener el identificador de usuario en la base de datos
            String query = "SELECT id FROM Usuario WHERE nick='"+
                    request.getParameter("user")+"' and password='"+
                    request.getParameter("pswd")+"'";

            PreparedStatement stmt = con.prepareStatement(query);
            ResultSet rs = stmt.executeQuery (query);

            int numCols = rs.getMetaData().getColumnCount ();

           // Si se encontró uno y sólo un identificador
            if(numCols==1&&rs.next()){
```

```
            // Obtiene la fecha y hora actuales y les da formato
             Calendar today=Calendar.getInstance();
             String date = String.format("%1$tY-%1$tm-%1$td %1$tH:%1$tM:%1$tS", today);

           // Query para insertar la ubicación en la base de datos
            query="INSERT INTO Reporte (Usuarioid,fecha,ubicacion,dispositivo,
            direccionIP) "+ "VALUES ("+rs.getString(1)+",'"+date+"','"+
                     request.getParameter("location")+"','"+
                     request.getParameter("device")+"','"+
                     request.getRemoteAddr()+"')";

             Statement command = con.createStatement();
             command.executeUpdate(query);

             out.println("Ubicación almacenada!");
          }
         // Cierra la conexión con la base de datos
         rs.close();
         stmt.close();
         con.close();
    ...
```

De este modo se reciben los datos desde los dispositivos móviles por medio del protocolo HTTP y se almacenan en la base de datos. Por lo tanto ahora es posible revisar el historial de reportes enviados desde el dispositivo móvil para realizar el seguimiento del dispositivo hasta su ubicación actual. El primer paso de este servicio consiste en autenticar al usuario que accede al servidor vía Web para realizar el seguimiento y una vez que se ha verificado su identidad, se muestra el historial de ubicaciones reportadas por cada usuario que le haya autorizado su rastreo.

A continuación se muestran los componentes principales del *Servlet* encargado de autenticar y mostrar el historial de ubicaciones de un usuario.

```
public class LoginServlet extends HttpServlet {

// Procesa las peticiones HTTP, tanto del método GET como POST.
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            // Se muestra el contenido de la página mediante código HTML
            out.println("<html>");
            out.println("<head>");
            ...

            String user = request.getParameter("user");
            String password = request.getParameter("password");

            // Formulario para capturar el nombre de usuario y contraseña
            if(user==null||password==null){
```

```
            out.println("<h3>Introduce tus datos</h3>");
            out.print("<form action=\"");
            out.print("LoginServlet\" ");
            out.println("method=POST>");
            ...

        }else{

            // Verifica que el usuario y contraseña sean correctos

            String query = "SELECT id FROM Usuario WHERE nick='"+user+
                    "' and password='"+password+"'";

            PreparedStatement stmt = con.prepareStatement(query);

            ResultSet rs = stmt.executeQuery (query);

            int numCols = rs.getMetaData().getColumnCount ();

            /* Si la autenticación es exitosa, se buscan los usuarios que
                       autorizaron a ser rastreados */
            if(numCols==1&&rs.next()){

                query = "SELECT id_usuario FROM Autorizado WHERE Usuarioid="+
                        rs.getString(1);

                stmt = con.prepareStatement(query);
                rs = stmt.executeQuery (query);

                if(rs.next()){

                    String user_id=rs.getString(1);

                    query = "SELECT nick FROM Usuario WHERE id="+user_id+"";
                    stmt = con.prepareStatement(query);
                    rs = stmt.executeQuery (query);
                    numCols = rs.getMetaData().getColumnCount ();

                    // Muestra el reporte de ubicaciones
                    if(numCols==1&&rs.next()){
                   out.println("<h3>Reporte de usuario: "+rs.getString(1)+"</h3>");
                         out.println("<table border=\"1\">");
                         out.println("<tr>");
                         out.println("<th>Fecha</th>");
                         out.println("<th>Coordenada</th>");
                         out.println("<th>Dispositivo</th>");
                         out.println("<th>Dirección IP</th>");
                         out.println("</tr>");

                         query = "SELECT fecha,ubicacion,dispositivo,"
                                 + "direccionIP FROM Reporte WHERE "
                                 + "Usuarioid='"+
                                 user_id+"'";
                         stmt = con.prepareStatement(query);
                         rs = stmt.executeQuery (query);
```

```
    while(rs.next()){
        out.println("<tr>");
        out.println("<td>"+rs.getString(1) +"</td>");
        out.println("<td>"+rs.getString(2) +"</td>");
        out.println("<td>"+rs.getString(3) +"</td>");
        out.println("<td>"+rs.getString(4) +"</td>");
        out.println("</tr>");
    }
    out.println("</table>");
...
```

En síntesis, los pasos que lleva a cabo este componente son los siguientes:

1. Verificar si en la petición (*request*) se envían los datos de usuario y contraseña. De ser así, verifica la información con la base de datos de nuestro sistema, en caso contrario, muestra un formulario HTML para que el usuario capture sus credenciales.
2. Cuando la autenticación del usuario es exitosa, se recupera el identificador de los usuarios que lo han autorizado para que los rastree, desde la base de datos.
3. Al contar con los identificadores de usuario, recupera sus reportes de ubicación enviados hasta el día de hoy. En este paso es posible filtrar los resultados por fecha, para disminuir la cantidad de datos a mostrar. Los reportes se despliegan en el formato de tabla HTML.

4.3 Plataformas de desarrollo móvil

Debido a la amplia variedad de fabricantes de dispositivos móviles y por consiguiente de sistemas operativos en estos equipos, existen diversas herramientas y lenguajes de programación que pueden ser utilizados para desarrollar aplicaciones móviles.

Entre las plataformas existentes, se encuentran: Java Micro Edition, iOS (Apple), Symbian, Brew, Windows Mobile, Linux Maemo, Android y Python para S60 [32]. Para estar en la posibilidad de probar nuestra propuesta en un entorno real, nos apegaremos a las plataformas disponibles para los dispositivos con los que contamos actualmente. En el caso del teléfono Nokia, se puede desarrollar la aplicación en Java Micro Edition, Symbian o Python para S60; para el teléfono HTC únicamente se pueden desarrollar aplicaciones del sistema Android.

Como podemos ver, será necesario desarrollar al menos dos versiones de la aplicación móvil para el mecanismo de localización, sin embargo, en el caso del dispositivo Nokia, lo ideal sería elegir una sola plataforma de desarrollo para enfocarnos en ella.

4.3.1 Java Micro Edition

Esta edición para móviles es la especificación de un subconjunto de la plataforma Java y busca proveer una colección estandarizada de APIs para el desarrollo de software para dispositivos portátiles y que poseen recursos limitados [32], entre los que se encuentran teléfonos móviles, PDAs, dispositivos empotrados y multimedia.

Los principales componentes de la plataforma Java ME son: *Connected Device Configurations* (CDC), *Connected Limited Device Configurations* (CLDC) y *Mobile Information Device Profiles* (MIDP) [33]. La primera configuración es para PDAs de gama alta y la segunda para teléfonos móviles y otros dispositivos portátiles. Las configuraciones son incrementadas mediante los perfiles, los cuales definen APIs adicionales para las aplicaciones a desarrollar.

El perfil más común es MIDP, existe otro perfil conocido como *Personal Profile*, que se enfoca hacia dispositivos empotrados. Su arquitectura se organiza por medio de capas como se puede ver en la Figura 4.7, y va desde el hardware del dispositivo a las aplicaciones móviles. En resumen, CLDC contiene un subconjunto de las bibliotecas de clases Java y define una máquina virtual para dispositivos con recursos limitados. MIDP incluye los APIs más relevantes para el entorno móvil. Las aplicaciones móviles de Java son conocidas como MIDlets (MIDP *applets*) y son soportadas por una amplia variedad de dispositivos de los principales fabricantes como Sony Ericsson, Nokia y Motorola.

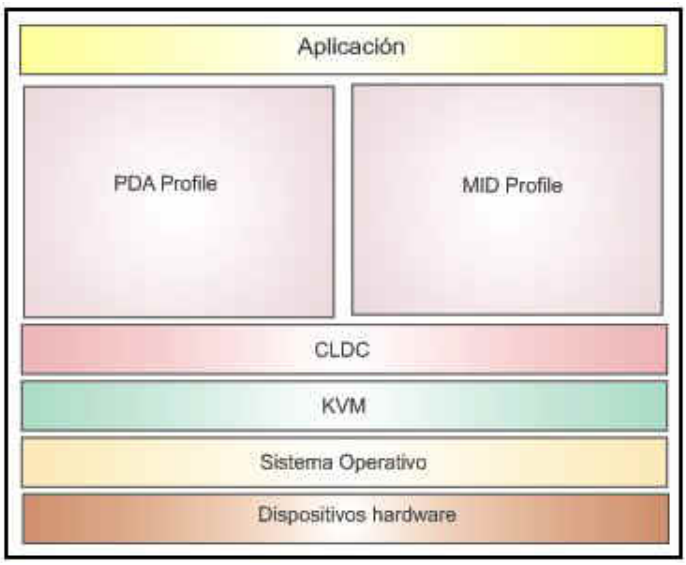


Figura 4.7: Arquitectura del entorno de ejecución Java ME

MIDP ofrece dos formas de desarrollar aplicaciones móviles: mediante APIs de alto nivel y bajo nivel [32]. El primero está diseñado para que las aplicaciones sean portables, es decir, posean las mismas características sin importar el dispositivo en que se ejecuten. El API de bajo nivel da a la aplicación el control directo de los gráficos a mostrar al usuario al permitirle cambiar su presentación y modo de interacción, sin embargo, en este caso el desarrollador debe tomar en cuenta las distintas características de los dispositivos, como es el tamaño de la pantalla.

Un MIDlet cumple con un ciclo de vida (Ver Figura 4.8) que consta de tres estados: pausado, activo y destruido. Al crear el MIDlet, el entorno de ejecución llama al constructor de la clase; cuando la aplicación se coloca en primer plano en la pantalla de dispositivo, se llama al método *startApp()* y cuando se deja de mostrar al usuario (pero continúa ejecutándose), se llama al método *pauseApp()*. Por último, cuando la aplicación termina su ejecución se llama al método *destroyApp()*.

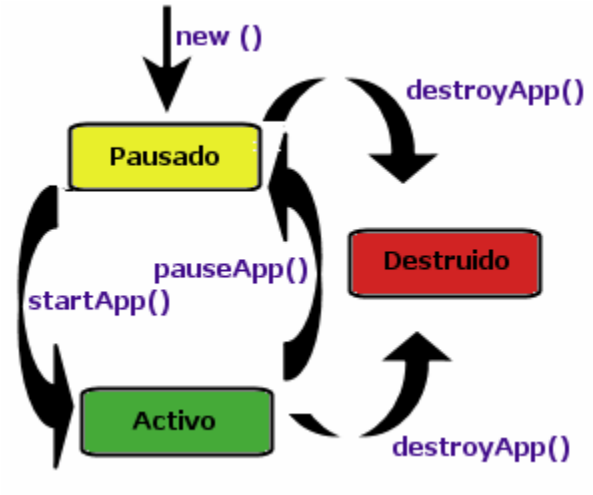


Figura 4.8: Ciclo de vida de un MIDlet

Para cumplir con los requerimientos descritos en el capítulo 3, Java ME cuenta con las siguientes interfaces de programación:

- *Location API*: Permite obtener información de la ubicación desde el dispositivo móvil.
- *Record Management System*: Maneja la persistencia de datos de la aplicación móvil en el dispositivo.
- *Push Registry*: Da a un MIDlet la posibilidad de iniciarse

automáticamente, sin necesidad de interacción por parte del usuario.

En general, Java ME ofrece la mayor portabilidad debido a que es soportada por la mayor cantidad de dispositivos móviles en el mundo, como se observó en la Figura 2.3.

Sin embargo, una de las desventajas de Java es que no permite el acceso a la funcionalidad subyacente del sistema y sólo expone las características del mismo a través de sus APIs [32]. Además, para tener acceso automático (sin solicitar permiso al usuario) a recursos como el receptor GPS o la conectividad inalámbrica, es necesario "firmar" o certificar la aplicación, proceso que conlleva tiempo y costo.

4.3.2 Android

Android es un sistema operativo y plataforma de software para dispositivos móviles basado en Linux. Utiliza Java como lenguaje de programación y entre las características más importantes de esta plataforma están:

- Máquina virtual (*Dalvik*) optimizada para dispositivos móviles.
- *SQLite* para almacenamiento estructurado de datos.
- Soporte de conectividad 3G y WiFi (dependiendo del hardware).
- Soporte de sensores como GPS, brújula y acelerómetro.

En la Figura 4.9 [34] se muestra los componentes más importantes del sistema operativo Android. Una de las diferencias con Java ME es que la máquina virtual no se ejecuta directamente sobre el sistema operativo del teléfono móvil (cualquiera que éste sea), sino sobre el kernel de Linux, para que éste se encargue de la funcionalidad a bajo nivel, como es la administración de procesos y memoria. Por lo tanto, las aplicaciones Android son soportadas únicamente por teléfonos que cuenten con dicho sistema operativo.

Entre las ventajas por las cuales Android ha tenido una amplia aceptación, tanto en el entorno de desarrolladores móviles como en el del mercado de *smartphones*, se encuentra el libre acceso a los recursos de hardware del dispositivo y a funcionalidades como la ejecución de servicios en segundo plano y la compartición de datos entre aplicaciones. No existe una distinción entre los APIs utilizados para desarrollar aplicaciones comunes y aquellos que se usan para crear aplicaciones de núcleo.

Las aplicaciones Android se ejecutan en un entorno aislado, para evitar problemas de seguridad. De este modo, basta con especificar en el código los

permisos necesarios para utilizar características del sistema, como son el GPS o la interfaz de red inalámbrica, sin necesidad de certificar la aplicación [32].

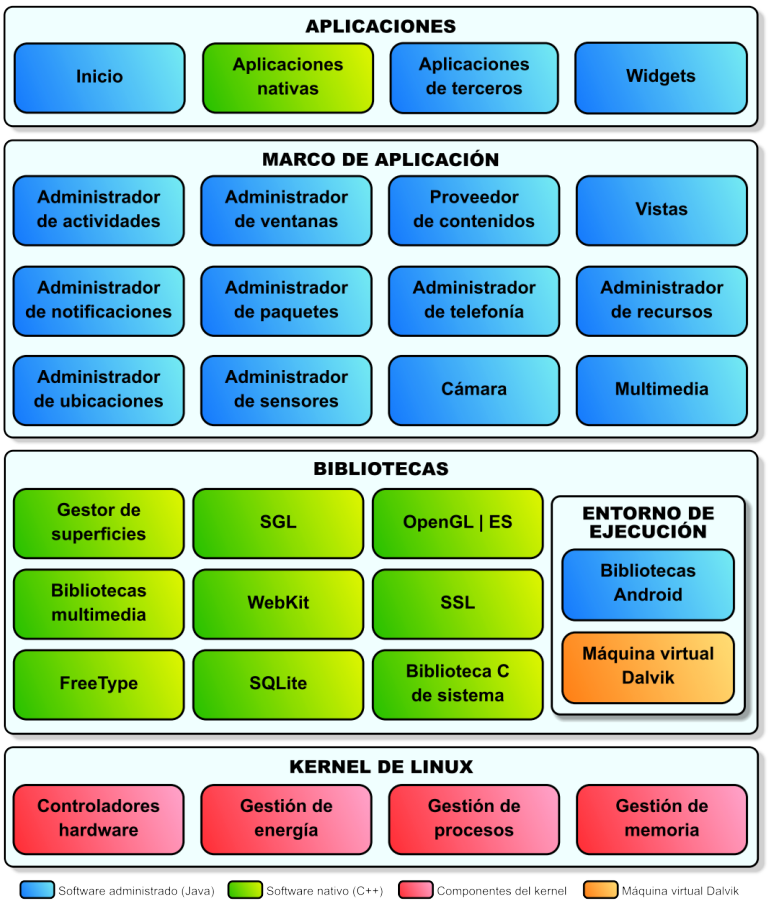


Figura 4.9 Arquitectura de Android

Entre las principales desventajas de desarrollar aplicaciones para Android está el número limitado de dispositivos que lo soportan, como fue posible observar en la Figura 2.3.

El común denominador entre las dos versiones de la aplicación móvil (Android y Java ME) es el uso del lenguaje de programación Java. Una vez elegidas las herramientas de desarrollo para la parte cliente, se procederá a la implementación de cada uno de los subsistemas y componentes del mecanismo de localización.

4.4 Implementación del cliente Java ME

Para que la aplicación móvil cumpla con los requerimientos del mecanismo de localización, es necesario que realice varias actividades de forma simultánea, es decir, a través del *multithreading*. Los APIs de desarrollo nos permiten especificar estas actividades para que se ejecuten en un proceso independiente (*thread*). En el caso de Java ME se cuenta con la clase `TimerTask`, que describe una tarea que puede ser programada o calendarizada para su ejecución.

4.4.1 Obtención de la ubicación

La obtención de la ubicación física es una tarea que debe ejecutarse de forma independiente y cíclica, por lo tanto se especifica en una clase que hereda de `TimerTask`:

```
public class LocationThread extends TimerTask{

    ...
    public void run(){

        Location l=null;
         LocationProvider lp=null;

        try{

            lp= LocationProvider.getInstance(cr);

            // Solicita la posición actual durante el tiempo especificado

            l= lp.getLocation(timeout);
        }catch(Exception e){
               l=lp.getLastKnownLocation();
         }

        Coordinates c;
        String coordenadas=null;

        // Obtiene las coordenadas, en el formato latitud,longitud

        if(l!=null&&(c = l.getQualifiedCoordinates()) != null ) {

            coordenadas=String.valueOf(c.getLatitude()) + "," +
               String.valueOf(c.getLongitude());

        }else{
```

```
            // Si por algún motivo no pudo obtener las coordenadas, envía las
            // últimas que se obtuvieron
            coordenadas=readLastLocation();
        }
        // Inicia el proceso de envío de datos al servidor SIP
        new SenderThread(midlet,coordenadas).start();

    }
}
```

El método *run()* contiene las instrucciones a ejecutarse en su propio *thread*. El API de Java nos permite obtener la posición física a partir de cualquier proveedor existente en el dispositivo (GPS o red), sin embargo, no permite especificar el uso de alguno de ellos es especial, por lo tanto cuando se obtiene la instancia del proveedor de ubicación, no se puede determinar si la ubicación se obtuvo vía satelital o a través de red. Al final de este método, el dispositivo envía su posición geográfica actual o la última de la que tenga registro.

4.4.2 Comunicación por redes IP

Todas las instrucciones que involucren comunicación por red deben ejecutarse en otro proceso (`SenderThread`).

```
public class SenderThread extends Thread{

    ...
    public void run() {
        try {

            SipClientConnection sc = null;

            // Se abre la conexión con el servidor SIP
            sc = (SipClientConnection)Connector.open("sip:"+user+"@148.247.102.22:5060");
            // Se inicia una petición de registro
            sc.initRequest("REGISTER", null);

            ...
            // Se establecen las credenciales para autenticar al usuario
            sc.setCredentials(user,readPassword(), "asterisk");

            // El asunto de esta petición será una actualización de ubicación
            sc.setHeader("Subject", "UPDATE");

            // Como dirección de contacto se envía la IP del dispositivo móvil
            sc.setHeader("Contact", "<sip:"+user+"@localhost>");

            // Si se pudieron obtener las coordenadas del receptor GPS, se
            // envían en el cuerpo del mensaje, con formato de texto plano
```

```
            if(coordenadas!=null){
                sc.setHeader("Content-Type", "text/plain");
                sc.setHeader("Content-Length", Integer.toString(coordenadas.length()));
                OutputStream os = sc.openContentOutputStream();
                os.write(coordenadas.getBytes());
                os.close();

            }else
                 sc.send();

        } catch (Exception e) {
            // Error: Se almacenar las coordenadas obtenidas para su envío futuro
            if(coordenadas!=null)
                saveLocation(coordenadas);
        }
    }
}
```

El momento en que se ejecutarán estas instrucciones no puede saberse con antelación, por lo tanto esta clase hereda de la clase `Thread`, la cual describe simplemente a un hilo de ejecución dentro de una aplicación.

Al momento en que la aplicación intenta enviar los datos hacia el servidor, se solicita al usuario que especifique de forma manual el método de conexión a utilizar, ya sea red Wi-Fi o celular. Después de que el usuario elige esta opción, se realiza con éxito el envío de datos. Como se puede observar, la elección oportunista del punto de acceso a Internet, no puede implementarse como se tenía previsto debido a que, con Java ME no es posible implementar de forma automática la elección de redes WiFi o celulares.

4.4.3 Calendarización de procesos

La aplicación móvil ejecuta dos actividades básicas: obtiene su ubicación física y establece la comunicación con el servidor para enviarle los datos. Esto se repite cada n minutos, de acuerdo a las preferencias del usuario. La ejecución cíclica de esta actividad no presenta dificultad alguna en caso de que la aplicación móvil esté siempre en primer plano, sin embargo, en un entorno real sabemos que esto no es posible, ya que el usuario utiliza el dispositivo móvil para muchos otros fines y llegará un punto en que sea necesario salir de nuestra aplicación.

Para resolver esto, se cuenta con el API *PushRegistry,* que permite que la aplicación móvil inicie su ejecución de forma automática, sin necesidad de que el usuario lo haga. Es posible iniciar una aplicación a través de una conexión de red o de un contador de tiempo; el modo que más se apega al escenario de uso de nuestra aplicación es el basado en el tiempo.

Al invocar el método `PushRegistry.registerAlarm()` se programa la futura ejecución de una aplicación. Los argumentos que necesita este método son el nombre completo del MIDlet y la hora en que se ejecutará (en milisegundos).

4.4.4 Alcance

Con las clases y métodos descritos hasta ahora, se cubren tres módulos de la aplicación móvil: ubicación, calendarización y comunicación SIP, quedando pendiente el módulo de conectividad. Desafortunadamente el conjunto de APIs de Java ME no permiten el acceso al hardware de red y por consiguiente no proveen las clases para manipular las interfaces de conexión disponibles en el dispositivo, sean redes WiFi o 3G. En caso de existir un API para manipular la conectividad de red del teléfono, la aplicación puede completarse de acuerdo a las necesidades de la propuesta.

Para resolver esta deficiencia, la selección de red a utilizar para el envío de información al servidor, se realiza de forma manual. Cuando el sistema operativo del teléfono detecta que la aplicación inicia una conexión hacia el servidor (`Connector.open()`), muestra al usuario la lista de puntos de acceso disponibles, ya sean redes Wi-Fi o celular. Depende del usuario el punto a utilizar y que la configuración del mismo sea la correcta.

Para cubrir este pendiente en nuestro trabajo, se implementó otra versión de la aplicación móvil para el sistema operativo Android, el cual sí provee los APIs de manejo de las interfaces de red disponibles en el dispositivo.

4.5 Implementación del cliente Android

A pesar de que la programación para la plataforma Android también se realiza en el lenguaje de programación Java, el desarrollo difiere de Java ME y por lo tanto es necesaria una reestructuración de la aplicación móvil implementada en el punto anterior.

4.5.1 Componentes de la aplicación

Existen cuatro tipos de componentes dentro de una aplicación Android [34]:

1. **Actividades**.- Representa una sola ventana con interfaz de usuario, por lo tanto una aplicación puede estar compuesta de varias actividades.
2. **Servicios**.- Es un componente que se ejecuta en segundo plano, para realizar operaciones de mayor duración. No posee una interfaz de usuario y puede ser iniciado por una actividad.
3. **Proveedores de contenido**.- Gestionan un conjunto de datos compartidos entre aplicaciones, como pueden ser los datos de contacto del usuario.
4. **Receptores de emisión**.- Es un componente que responde a anuncios que pueden provenir del sistema (*e.g.* batería agotada) o de las aplicaciones. Aunque tampoco presentan interfaz de usuario, pueden mostrar notificaciones en la barra de estado del dispositivo. En general solo son una puerta de enlace hacia otros componentes.

Cada uno de estos componentes son un punto de entrada distinto para que el sistema ejecute la aplicación, por lo tanto el enfoque es muy distinto al de los MIDlets de Java ME.

Para el caso de nuestra aplicación móvil, sólo se necesitarán tres componentes: **actividades**, **servicios** y **receptores**, ya que no será necesario manipular datos compartidos con otras aplicaciones. Los tres componentes a utilizar pueden ser activados por mensajes asíncronos denominados *intents*, los cuales enlazan un componente con otro en tiempo de ejecución. En el caso de las actividades y los servicios, un `Intent` define la acción a realizar; en el caso de los receptores de emisión, define el anuncio a emitir.

4.5.2 Servicio de ubicación

La obtención de la ubicación física debe realizarse en un proceso independiente, y de preferencia en segundo plano, por lo tanto se especificará su funcionalidad dentro de un **Servicio**. Existen dos formas de iniciar un servicio: con el método *startService()*, si no se espera recibir algún resultado y con el método *bindService()* si se desea tener una interfaz tipo cliente-servidor que permita interactuar con el servicio y obtener resultados.

En nuestro caso, deseamos que el servicio nos provea las coordenadas físicas del dispositivo móvil, por lo tanto debemos utilizar el método *bindService()* que a su vez recibe como parámetros: el servicio al que nos queremos enlazar (definido mediante un `Intent)`, una instancia de `ServiceConnection` a la cual reportar el estado de la conexión y banderas para distintas opciones. Esta llamada debe realizarse en nuestra **Actividad** principal:

```
bindService(new Intent(MyActivity.this, LocationService.class), serviceConnection,..)
```

De este modo, nuestro servicio estará definido en la clase `LocationService`, y como todo servicio, debe implementar algunos métodos que manejan aspectos clave del ciclo de vida del servicio, como se puede ver en el siguiente fragmento de código:

```
public class LocationService extends Service{

        ...
        // Método llamado cuando se crea el servicio
        public void onCreate (){

                /* Obtiene la última ubicación conocida en cada proveedor (GPS y red
                 inalámbrica) */
                lastGPSLocation = locationManager.getLastKnownLocation(GPS_PROVIDER);
                lastWiFiLocation = locationManager.getLastKnownLocation(NETWORK_PROVIDER);

                /* Compara las dos coordenadas obtenidas, para quedarnos con la más
                 * exacta y reciente */
                bestLocation = isBetterLocation(lastWiFiLocation,lastGPSLocation)?
                lastWiFiLocation:lastGPSLocation;

                /* Solicita notificaciones de ubicación por parte del GPS (cada 9 segundos)
                 * y de la red (cada 17 segundos) */
                locationManager.requestLocationUpdates(GPS_PROVIDER, 0, 0, locListener);
                locationManager.requestLocationUpdates(NETWORK_PROVIDER, 0,0, locListener);

        }

        // Método llamado al detener este servicio
        public void onDestroy() {
                locationManager.removeUpdates();
                stopSelf();
        }
}
```

El sistema llama al método *onCreate()* al crear el servicio, y al método *onDestroy()* cuando ya no es utilizado y será destruido. Al crear el servicio se obtiene la última posición geográfica conocida, ya sea por medio del GPS integrado o a través de la red inalámbrica (Wi-Fi o 3G) y se elige una de estas coordenadas dependiendo cuál sea la más reciente o la más exacta. Esto se hace debido a que por lo regular, la obtención de los datos de ubicación del dispositivo es una operación que consume tiempo y puede llegar a prolongarse más de lo debido.

Por último se solicita al gestor de la ubicación (`LocationManager`) que notifique eventos relacionados con la ubicación del dispositivo, ya sea del proveedor GPS o del proveedor de red. Al destruir el servicio es necesario indicar que ya no se desean recibir más notificaciones de ubicación.

4.5.3 Servicio de calendarización

El método que solicita las notificaciones es *requestLocationUpdates()* y recibe como parámetros al proveedor de ubicación, el intervalo mínimo (milisegundos) entre notificaciones, la distancia mínima (metros) entre notificaciones y al receptor de las notificaciones. De este modo se implementa de forma automática el módulo de calendarización, necesario para ejecutar de forma cíclica la obtención de las coordenadas geográficas del dispositivo móvil.

Las notificaciones de ubicación son recibidas por una clase que implementa a la interfaz `LocationListener` junto con sus métodos abstractos, que son llamados por el gestor de ubicación cuando, por ejemplo, el usuario cambia de posición geográfica.

El método *onLocationChanged()* se ejecuta cuando se encuentra una nueva coordenada geográfica por el proveedor de ubicación, y la recibe como parámetro:

```
public void onLocationChanged(Location location) {

	// Si la nueva posición es 'mejor' que la que se tenía previamente
	if(isBetterLocation(location,bestLocation)){

		bestLocation=location;
		String coordinates=String.valueOf(location.getLatitude())
			+ "," + String.valueOf(location.getLongitude());

		if(!isConnected())
		// Si no se cuenta con conexión a Internet, se debe almacenar la
		//información localmente
		else
		// Se envía la coordenada por Internet (SIP)
	}
}
```

4.5.4 Servicio de conectividad

En la versión del cliente en la plataforma Java ME nos encontramos con la desventaja de no contar con los APIs para controlar el hardware de red del

dispositivo móvil. Android sí ofrece las clases necesarias para determinar el estado de la conectividad del dispositivo móvil, y en caso de no contar con conexión a Internet, ser capaz de realizar una exploración para encontrar redes inalámbricas en la cercanía.

El siguiente método nos ayuda a determinar si el dispositivo cuenta con una conexión activa a Internet:

```
public boolean isConnected(){

       WifiManager wifiManager=(WifiManager)getSystemService(WIFI_SERVICE);

       if (connectivityManager.getNetworkInfo(TYPE_MOBILE).getState() == CONNECTED ||

              connectivityManager.getNetworkInfo(TYPE_WIFI).getState() == CONNECTED  ){

              // Conectado a Internet
              return true;

       }else{
              ...
              if(wifiManager.isWifiEnabled()){

                     configuredNetworks=wifiManager.getConfiguredNetworks();
                     // Se intenta conectar a las redes pre-configuradas (si las hay)

                     if(!configuredNetworks.isEmpty())
                       attemptConnection(configuredNetworks);

                     else
                     // Si no hay redes pre-configuradas, busca redes inalámbricas
                         searchWiFi();
              }else
                     wifiManager.setWifiEnabled(true); //Activa la interfaz WiFi
              ...
       }
       return false;
}
```

En caso de que el dispositivo no tenga conectividad a Internet en el momento en el que se necesita establecer comunicación con el servidor, el primer paso es tratar de conectarse a alguna red conocida, es decir, que alguna vez haya sido utilizada para acceder a Internet. Es en este punto donde entra el tercer componente de las aplicaciones Android: el receptor de emisión o `BroadcastReceiver`, cuya función será captar los eventos que ocurran en la interfaz de red del dispositivo.

Para intentar la conexión con una red conocida se implementó el método *attemptConnection()*, que recibe como parámetro la lista de redes conocidas. En cuanto el estado de la interfaz de red cambie, se generará un `Intent`, por lo tanto debemos definir un filtro (`IntentFilter`) para informar al sistema que nuestro

`BroadcastReceiver` es capaz de manipular este tipo de evento.

```
public synchronized void attemptConnection(List<WifiConfiguration> configuredNetworks){

       // Itera sobre el conjunto de redes conocidas
       ListIterator<WifiConfiguration> iterator=configuredNetworks.listIterator();

       // Creación del IntentFilter para los eventos de estado de red
       IntentFilter intentFilter = new IntentFilter();
       intentFilter.addAction(WifiManager.NETWORK_STATE_CHANGED_ACTION);

       // Registra un nuevo receptor para los intents que cumplan con el filtro
       registerReceiver(new ConnectionReceiver(iterator), intentFilter);

       // Permite que se asocie a una red conocida
       wifiManager.enableNetwork(iterator.next().networkId,true);
}
```

El receptor de la emisión sólo es válido durante la ejecución de su método *onReceive()*, al finalizar, el objeto deja de estar activo. `ConnectionReceiver` es un receptor que se activa cuando el estado de la interfaz de red ha cambiado:

```
public class ConnectionReceiver extends BroadcastReceiver {

       private ListIterator<WifiConfiguration> iterator;

       ...
       public void onReceive(Context c, Intent intent) {

              // Determina el estado de la interfaz de red
              switch(connectivityManager.getNetworkInfo(TYPE_WIFI).getState()){

                     case CONNECTED:
                            //Conectado a una red conocida
                     break;

                     case DISCONNECTED:

                     // Si sigue desconectado, se intenta con la siguiente red WiFi
                     if(iterator.hasNext()){

                            WifiConfiguration network=iterator.next();
                            wifiManager.enableNetwork(network.networkId,true);

                     }else

                     // si no hay otra red configurada, se buscan redes en la cercanía
                     parent.searchWiFi();
                     break;
              ...
       }
}
```

En caso de que los intentos de conexión no sean exitosos, es necesario buscar redes inalámbricas en la cercanía del dispositivo móvil. Para lograrlo también se hará uso de un receptor de emisión (`ScanReceiver`), para captar la notificación del sistema de que existen resultados de la exploración de redes WiFi:

```
private void startWifiScan(){

        IntentFilter i = new IntentFilter();
        i.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);

        registerReceiver(new ScanReceiver(this), i );

        WifiManager.startScan();
}

public class ScanReceiver extends BroadcastReceiver {
        ...
        // Método invocado al recibir los resultados de la exploración
        public void onReceive(Context context, Intent arg1) {

                // Obtiene los resultados de la exploración
                List<ScanResult> APs=wifiManager.getScanResults();
                // En caso de haber encontrado redes al alcance
                if(!APs.isEmpty()){

                        // Ordena las redes encontradas por intensidad de señal
                        Collections.sort(APs,new ScanComparator());
                        ...
                        attemptConnection(APs);
                }
        }
}
```

El método exhaustivo implementado hasta ahora se basa en el concepto de la comunicación oportunista, en el cual se debe aprovechar cualquier punto de conectividad para enviar los datos hacia el servidor del mecanismo de localización. Primero se busca la conectividad con todas las redes configuradas previamente en el dispositivo, según sea la prioridad que tienen asignada, y posteriormente se realiza la búsqueda de redes en la cercanía, y se intenta conectar a cada una de ellas con base en la intensidad de su señal.

Si la conexión vía Wi-Fi no es posible, se debe intentar la conexión vía **3G** o GPRS (Ver Figura 3.9), para lo cual, Android provee los medios para configurar un punto de acceso a la red celular de datos. Los datos del punto de acceso deben almacenarse en una base de datos existente en el dispositivo, en la cual se almacena la información de los proveedores de Internet celular.

El siguiente fragmento de código muestra el ejemplo de configuración de un punto de acceso, para el caso del proveedor Telcel.

```
ContentValues values=new ContentValues();
values.put("name","Internet");
values.put("mcc","334");
values.put("mnc","020");
values.put("apn","Internet.itelcel.com");
values.put("user","webgprs");
values.put("password","webgprs2002");
values.put("type","default");

// Inserción en la base de datos
getContentResolver().insert(Uri.parse("content://telephony/carriers"),values);
```

Al contar con un punto de acceso válido, es posible enviar datos a través de la red 3G o GPRS, según sea el caso, hacia el servidor de localización.

4.5.5 Alcance

De acuerdo a los servicios descritos hasta este punto, es posible observar que se implementó la aplicación cliente completa, con todos los módulos especificados en el capítulo 3. Desde nuestro punto de vista, esta plataforma ofrece las mayores ventajas en cuanto a la implementación del mecanismo de localización tal y como se tenía pensado desde un principio. En el siguiente capítulo se aplicarán pruebas para corroborar esta idea.

4.6 Discusión

Una vez que se cuenta con el diseño de la solución, se debe llevar a cabo la planificación del desarrollo, comenzando con la elección de las herramientas convenientes para nuestro caso particular.

La fase de desarrollo se tuvo que adecuar a la infraestructura de cómputo existente y se dio preferencia al uso de software libre. Para el caso del servidor, se configuró el PBX *Asterisk* y el contenedor Web conocido como *Apache Tomcat*, para atender las peticiones de los clientes mediante el protocolo SIP o HTTP. El almacenamiento de la información se realiza por medio de una base de datos MySQL.

Fue necesario modificar el código fuente de Asterisk para adaptar su funcionamiento a los requerimientos de nuestra propuesta, en específico, se manipuló el mecanismo de registro de *peers* SIP para que fuera posible almacenar la información de la ubicación en nuestra base de datos. En la implementación del

Servlet también se incluyó la conexión a MySQL.

Para el caso del cliente, se llevó a cabo un análisis de las plataformas de desarrollo móvil disponibles. Tomando en cuenta sus ventajas y desventajas, además de apegarnos a los dispositivos con los que se cuenta, se optó por implementar la parte cliente de nuestro mecanismo de localización en la plataforma Java ME y Android. Estas dos plataformas tienen en común el lenguaje de programación Java.

A lo largo del capítulo se detallaron cada uno de los módulos y componentes, incluyendo en la mayoría de los casos, los fragmentos de código más relevantes para mostrar la implementación de la propuesta. No fue posible implementar la comunicación oportunista en la plataforma Java ME, este punto se completó en la plataforma Android.

La aplicación móvil desarrollada para Android es la que presenta la funcionalidad más completa con respecto a los requerimientos detallados en el capítulo 3.

Capítulo 5

Pruebas y resultados del sistema

La fase de pruebas de un proyecto busca comprobar el correcto funcionamiento del sistema desarrollado, así como el cumplimiento de los objetivos planteados al diseñar y estructurar la aplicación. En este capítulo se llevan a cabo distintos tipos de pruebas con la finalidad de corroborar que el sistema realiza las actividades para las que fue construido, siendo la actividad principal el envío de reportes de ubicación del usuario desde un dispositivo móvil hacia el servidor.

En el capítulo 3 se especificó la arquitectura del sistema como cliente-servidor, por lo tanto es necesario llevar a cabo pruebas en estos dos componentes, de forma separada y durante su interacción. Asimismo, como cada elemento se desarrolló de forma modular, las pruebas se realizan en cada módulo por separado y como conjunto.

Después de realizar las pruebas, es necesario analizar los datos obtenidos para obtener conclusiones sobre el el funcionamiento del mecanismo de localización y su desempeño en un entorno real.

5.1 Pruebas de la aplicación móvil

Durante la especificación del sistema se determinó que la aplicación móvil se conformaría de los siguientes módulos:

- Ubicación
- Comunicación
- Autenticación
- Calendarización

Debido a las características de cada módulo, las pruebas fueron más exhaustivas en unos módulos que en otros. En las secciones subsecuentes se detallan las pruebas realizadas en cada uno de ellos, primero se describe el caso del teléfono Nokia N95 y después el del teléfono Android.

5.1.1 Pruebas del módulo de ubicación

Los elementos principales a comprobar en esta sección son los métodos de obtención de la ubicación desde el dispositivo móvil, los cuales pueden ser el GPS o la red inalámbrica (WiFi o celular). Entre los aspectos que caracterizan a estos métodos son el consumo de batería y la latencia para obtener las coordenadas geográficas del dispositivo. Es por esto que la primera prueba consistió en medir el tiempo necesario para que el dispositivo cuente con información de su posición. Los receptores GPS necesitan un tiempo de calibración inicial conocido como TTFF (*Time To First Fix*), que es el tiempo empleado para dar una primera posición después de poner en marcha al receptor.

Un receptor que está apagado o con un tiempo considerable en desuso, realiza un arranque "en frío" (*cold start*) durante el cual debe obtener todos los datos posibles para determinar su ubicación. Por el contrario, se denomina *warm start* a la situación en la cual el receptor GPS cuenta con la información necesaria en memoria y por lo tanto tarda menos en obtener la posición física del dispositivo.

Las pruebas se realizaron al aire libre, en la calle y en espacios abiertos como parques y plazas, con la finalidad de que las pruebas se asemejen lo más posible al entorno en que comúnmente se utilizaría nuestro método de ubicación. Para el caso del teléfono **Nokia N95**, se cuenta con una aplicación pre-instalada que permite habilitar o deshabilitar los siguientes métodos de ubicación disponibles:

- GPS Bluetooth
- GPS asistido
- GPS integrado
- Basado en la red

El primer método no es posible utilizarlo ya que no se cuenta con un GPS externo que permita conexión a través de Bluetooth. El método basado en la red no fue capaz determinar la ubicación del teléfono celular por sí solo, por lo tanto se tiene la hipótesis de que el dispositivo no es capaz de obtener la información suficiente, ya sea de la red celular o de la red WiFi.

Se realizaron series de 30 lecturas del receptor GPS, utilizando los métodos mencionados anteriormente. Entre cada lectura se apagó por un momento el

receptor GPS con la finalidad de eliminar datos de memoria y y forzar a que el receptor obtuviera una nueva coordenada geográfica. Los tiempos se midieron mediante un cronómetro, y se obtuvieron los resultados mostrados en la Tabla 5.1.

	TTFF	**Mínimo**	**Máximo**	**Promedio**
GPS simple	289	11.2	38.3	18.41
GPS asistido	410	8	36.6	19.8
GPS combinado	103	18.7	35.1	25.32
Aplicación	N/A	4.843	38.773	16.65

Tabla 5.1: Tiempo de lectura del GPS en el teléfono N95

En la primera prueba se utilizó únicamente el GPS integrado, en la segunda el GPS asistido, el cual teóricamente tiene la finalidad de disminuir los tiempos de obtención y cálculo de la posición geográfica del dispositivo. La tercera prueba consistió en activar tanto el método de GPS integrado como asistido. Obsérvese que a pesar de que el tiempo para la calibración fue significativamente menor que en las pruebas anteriores, el tiempo para las demás lecturas fue mayor (reflejado en el promedio).

Las tres primeras pruebas se realizaron utilizando la aplicación pre-instalada en el teléfono, y tuvieron la finalidad de estimar el tiempo que comúnmente implica determinar la ubicación del teléfono mediante los satélites GPS, independientemente de la aplicación desarrollada.

La cuarta prueba consistió en medir el desempeño del módulo de ubicación de la aplicación móvil implementada, para lo cual se llevaron a cabo dos series de 30 pruebas con la siguiente configuración:

- Frecuencia de envío: cada minuto
- Tiempo de espera (timeout): 45 segundos

Esta configuración es en la que se hacen envíos más frecuentes, respetando el tiempo que necesita el GPS para obtener la ubicación y que fue determinado con la aplicación pre-instalada en el teléfono. Debido a que la aplicación móvil requiere determinar la ubicación del dispositivo de forma cíclica, ya no se apaga el receptor GPS porque en un entorno real el receptor estará siempre activo. El comportamiento del receptor GPS sigue siendo similar al percibido en las pruebas anteriores.

En el caso del teléfono con sistema operativo **Android**, no se cuenta con una aplicación pre-instalada para medir la latencia de obtención de la ubicación, por

lo tanto fue necesario desarrollar una aplicación que permita realizar las métricas necesarias.

En el capítulo 4 se mencionó que, para desarrollar aplicaciones conscientes de la ubicación, los APIs de Android ofrecen la posibilidad de determinar la ubicación mediante GPS o mediante la red inalámbrica (celular o WiFi). La primer prueba consistió en tratar de obtener la ubicación del usuario, manteniendo el GPS, la red Wi-Fi y la conectividad de datos 3G desactivados, para ver si era posible determinar la ubicación únicamente con la señal GSM. El resultado después de varios intentos, fue que el sistema proporciona la ubicación de la antena de telefonía más cercana y no es capaz de calcular una nueva coordenada geográfica, por lo tanto una ubicación confiable no puede ser determinada utilizando únicamente la red GSM.

Para la siguiente prueba habilitamos la conectividad de datos 3G y observamos que para obtener la primera coordenada geográfica desde el dispositivo, tuvieron que pasar 210 segundos (3.5 minutos). Las actualizaciones subsecuentes implicaron tiempos similares o mayores, por lo tanto no lo consideramos un método viable para la aplicación móvil desarrollada.

A diferencia del dispositivo Nokia, en estas pruebas sí fue posible determinar la ubicación del dispositivo mediante la red WiFi. Para las pruebas del GPS en el teléfono Android, también se tomaron 30 lecturas y se observó que, aunque se encuentre apagado el receptor GPS, no necesita un TTFF en el orden de los minutos, como fue el caso del dispositivo Nokia. La primera lectura se obtuvo en cuestión de segundos y los tiempos subsecuentes se reflejan en la Tabla 5.2.

	Mínimo	Máximo	Promedio
WiFi	5	52	16.48
GPS simple	4	67	17.56
GPS y 3G	2	30	8.6
Aplicación	0.109	20.6	8.2

Tabla 5.2: Tiempo para obtener la ubicación en el teléfono Android

En la segunda serie de pruebas al GPS, se activaron los datos 3G con el objetivo de determinar si se optimizaba el tiempo de cálculo de la coordenada geográfica. Como se puede observar, la hipótesis fue comprobada debido a que el tiempo promedio para determinar una coordenada fue mucho menor. Esto nos lleva a concluir que el dispositivo no es capaz de obtener su ubicación utilizando únicamente los datos 3G, como lo vimos en las pruebas anteriores, sin embargo el GPS sí es capaz de utilizar estos datos para optimizar el tiempo de cálculo de sus coordenadas.

Teniendo los tiempos de obtención de coordenadas desde red y GPS, se procedió a realizar las pruebas con la aplicación móvil que implementa nuestra propuesta. La aplicación tiene la posibilidad de obtener información de la ubicación de todos los proveedores disponibles, por lo tanto se especificaron los siguientes intervalos:

- Lectura del GPS cada 9 segundos
- Lectura de la red cada 17 segundos.

Aquí se puede observar una mejora con respecto al teléfono Nokia, debido a que el intervalo fue menor y por lo tanto se enviarán reportes con mayor frecuencia al servidor. De este modo se llega a la conclusión de que al utilizar varios proveedores de ubicación (GPS y red), el tiempo de cálculo de la coordenada geográfica disminuye, y por lo tanto la información enviada al servidor permitirá aproximar la ubicación del usuario de forma más frecuente.

5.1.2 Pruebas al módulo de comunicación/autenticación

La siguiente prueba a realizar, una vez que se están obteniendo coordenadas geográficas en el dispositivo móvil, es la verificación de que toda la información enviada desde el cliente, llegue con éxito al servidor.

Como se mencionó en el capítulo 4, la autenticación entre el cliente y el servidor se realiza a través del protocolo SIP, por lo tanto es necesario capturar los paquetes enviados a través de la red y analizar su contenido, para verificar que la información sea consistente entre un extremo y el otro.

Para capturar y analizar los paquetes que recibe el servidor, se utiliza la herramienta *Wireshark*, la cual muestra todos los paquetes que transitan en determinada interfaz de red.

Antes de que el servidor acepte la información de la ubicación enviada desde algún dispositivo móvil, es necesario que éste se identifique por medio del nombre del usuario que lo está utilizando. Para llevar a cabo esta autenticación, se utiliza el método `REQUEST` del protocolo SIP, el cual es utilizado durante el registro de agentes de usuario ante un servidor SIP o registrador (Ver Figura 5.1). El primer paso consiste en que el cliente envíe la solicitud `REQUEST` al servidor, a lo cual recibe una respuesta con código 401 (No autorizado); esto significa que es necesario que se autentique con su nombre de usuario y contraseña. Una vez que cuenta con estos datos, los agrega a un campo del encabezado y reenvía la solicitud `REGISTER` al servidor. Si los datos son correctos, el servidor responde con un mensaje del tipo `OK`. Es importante mencionar que el nombre de usuario SIP

es sensible al uso de mayúsculas.

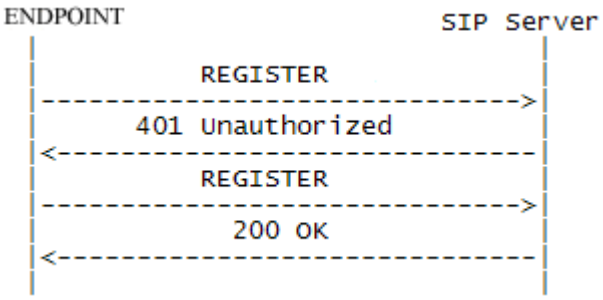


Figura 5.1: Registro en SIP

Por lo tanto la primer prueba consistió en verificar que los paquetes SIP cumplían con esta secuencia de intercambio, lo cual se puede corroborar en la Figura 5.2, en donde se muestra el caso en el que el usuario ingresa correctamente sus datos de usuario y contraseña.

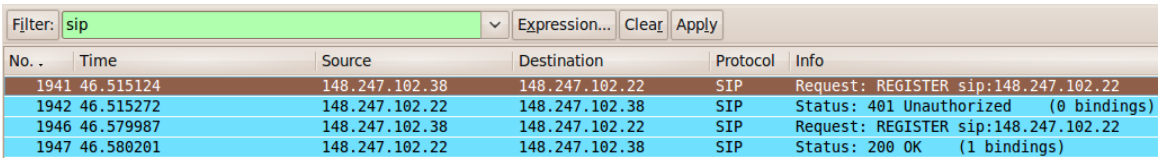


Figura 5.2: Captura de paquetes de registro SIP

Para el caso contrario, en el cual los datos del usuario son incorrectos, ocurre la secuencia de mensajes mostrada en la Figura 5.3. El servidor responde con un mensaje `403 Forbidden`, que significa que el servidor entendió la petición pero se niega a completarla (porque el usuario no es quien dice ser), además se le indica al agente de usuario que no vuelva a enviar la petición.

Por último, es importante destacar que durante la realización de todas las pruebas, nunca hubo ausencia o pérdida de paquetes al enviarlos desde el teléfono móvil hacia el servidor. Aún así, la pérdida de paquetes es algo que depende más de la red que de la aplicación desarrollada.

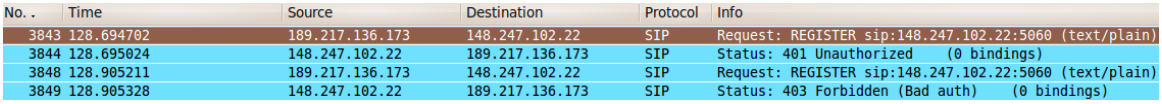


Figura 5.3: Autenticación fallida en SIP

También se verificó que cada uno de los paquetes SIP contenga la coordenada de ubicación del dispositivo móvil, como se muestra en la Figura 5.4:

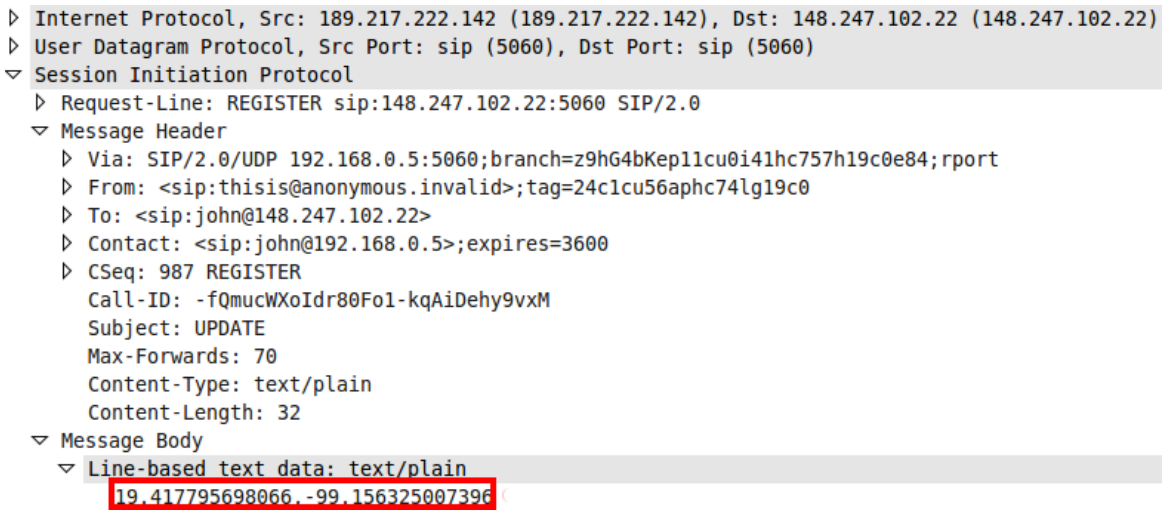


Figura 5.4: Detalle de un paquete SIP recibido en el servidor

El teléfono **Android** cuenta con la versión 2.2 del sistema operativo del mismo nombre, desafortunadamente el soporte para SIP se ofrece a partir de la versión 2.3 y por lo tanto no fue posible implementar el módulo de autenticación de la forma descrita previamente. Se optó por realizar el envío de reportes a través de HTTP, implementando un *Servlet* que recibe los datos en el servidor.

De este modo corroboramos que SIP no sólo es útil para el manejo de sesiones de usuario sino también como medio confiable de autenticación, ya que al implementar la comunicación únicamente utilizando HTTP, el envío de datos se realiza en texto plano, lo cual pone en riesgo la información de los usuarios. Para evitar esto sería necesario desarrollar otro módulo independiente que se encargue de la seguridad de los datos enviados, lo cual está fuera del alcance de este trabajo.

A partir de un filtrado con la herramienta *Wireshark*, es posible verificar que no se pierdan los paquetes enviados desde el teléfono móvil, lo cual no ocurrió durante las pruebas realizadas. Esto es importante debido a que inicialmente, se tuvo la hipótesis de que el tráfico de red y la pérdida de paquetes podría afectar la entrega de los paquetes que envía el dipositivo móvil al servidor. La implementación de nuestra propuesta se caracteriza por enviar una cantidad muy pequeña de paquetes.

La propiedad que permite identificar los paquetes de nuestra aplicación es la bandera `PUSH` activada (valor = 1), lo cual denota el envío de datos desde el

cliente al servidor. En la Figura 5.5 se muestra la captura de este tipo de paquetes, en la parte superior se muestran algunos de los paquetes enviados desde el cliente hacia el servidor, posteriormente las banderas activadas en uno de ellos y por último el contenido del paquete (puede observarse en la parte inferior la petición del tipo `POST`). El método `POST` en HTTP se utiliza para enviar información desde un cliente hacia un servidor Web. Entre los usos más comunes, este método permite enviar los datos de un formulario, que el usuario llena en una página HTML, hacia el servidor para que almacene esa información en una base de datos o los envíe por correo electrónico.

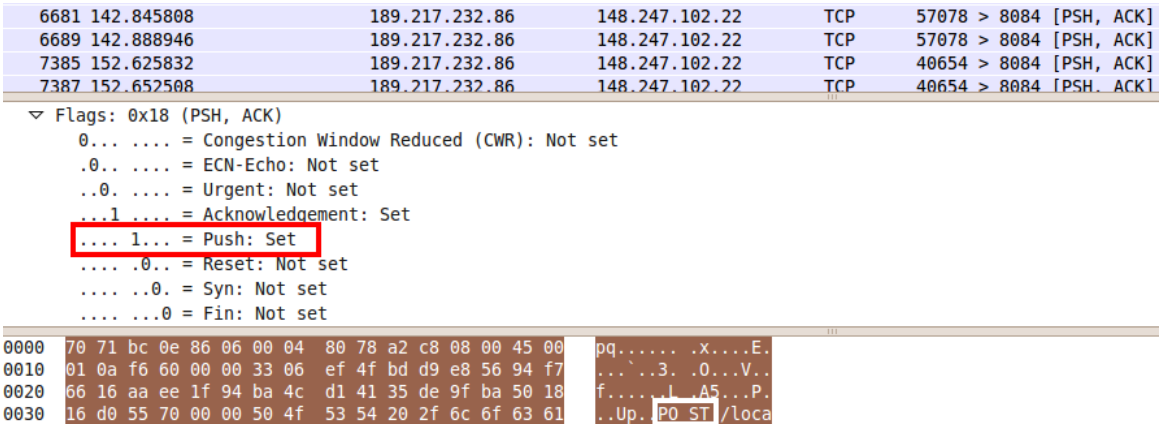


Figura 5.5: Reportes de ubicación enviados por HTTP

5.1.3 Pruebas al módulo de calendarización

Otra parte de la aplicación móvil que debe ser verificada es la que corresponde a las preferencias del usuario con respecto a la frecuencia de envío de la ubicación hacia el servidor. Seleccionando el intervalo de un minuto, se observaron variaciones en el orden de los milisegundos, las cuales suponemos se originan del tiempo que implica la ejecución de cada instrucción de la aplicación.

La Figura 5.6 muestra las variaciones entre cada evento de obtención y envío de la ubicación. Las variaciones negativas significa que el evento inició antes de lo requerido (menos del tiempo de espera). El mayor retraso fue de 56 milisegundos y el menor de 8. El máximo anticipo en la ejecución de la tarea fue de -19 milisegundos y el mínimo de -2 milisegundos.

Las diferencias comparando una ejecución contra la siguiente son insignificantes, sin embargo si acumulamos los retrasos y adelantos y los comparamos con el tiempo en que se inició la primera obtención de la

coordenada, el desfase es más notable, aún así no tiene relevancia en el escenario de uso de nuestra aplicación, ya que no se trata de una aplicación sensible al tiempo de ejecución. La gráfica de variación acumulada se muestra en la Figura 5.7.

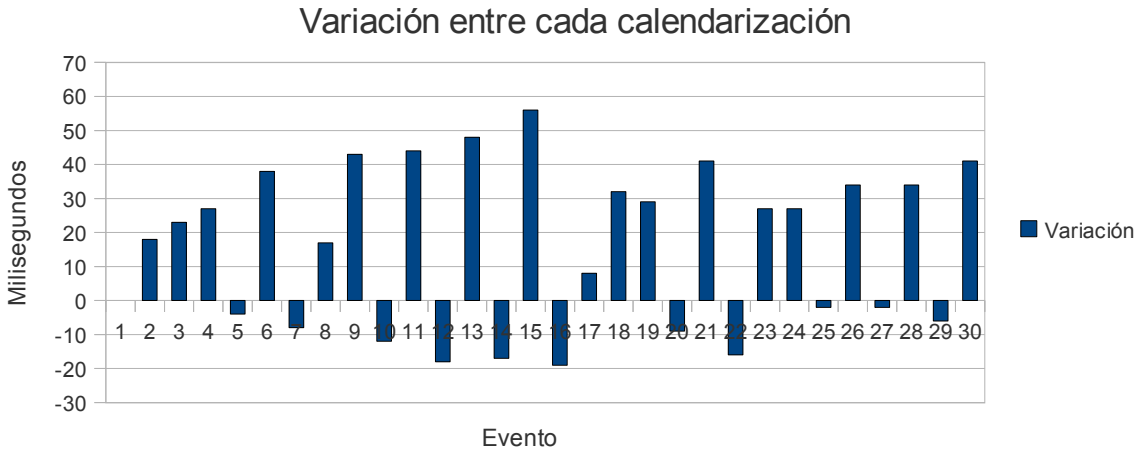


Figura 5.6: Variación entre cada evento de obtención de la ubicación

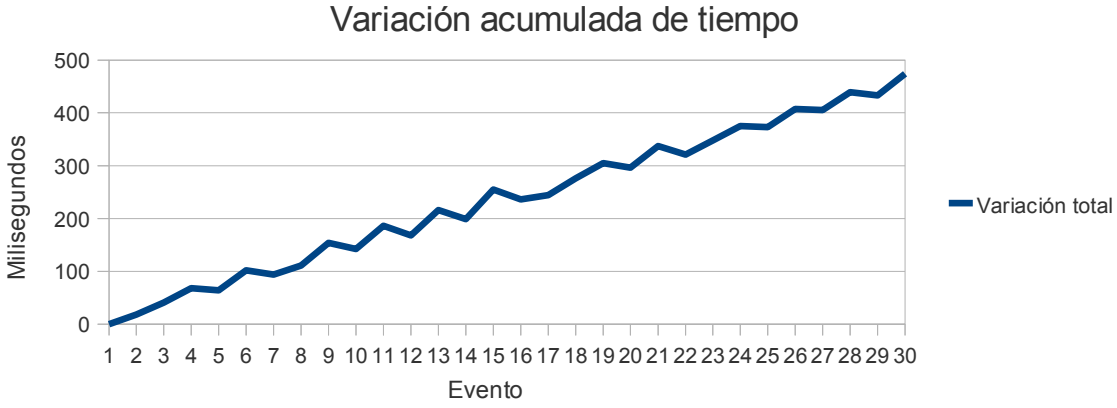


Figura 5.7: Variación acumulada entre cada obtención de la ubicación

Para el caso del teléfono **Android** no se implementó una calendarización fija como tal, sino que se utilizó un método provisto por el API de localización, que solicita la notificación periódica del proveedor de localización hacia la aplicación desarrollada. La documentación nos dice que el proveedor puede tomar una pausa de acuerdo al intervalo que se especifique (ver apartado 4.3.1), sin embargo no nos dice que esta pausa sea obligatoria ni que siempre ocurra. Esto provoca que la

ubicación sea determinada antes o después del intervalo especificado, según sea la disponibilidad de la misma y que los reportes se envíen al servidor con una frecuencia variable.

Al finalizar las pruebas detalladas, se llegó a las siguientes conclusiones:

- El mecanismo de localización permite el envío frecuente de la información de la ubicación hacia el servidor, por lo tanto es factible su uso en un entorno real.
- El tipo de dispositivo no afecta el envío de reportes. La latencia en la obtención de las coordenadas es la que puede llegar a afectarlo.
- A mayor frecuencia de envío y número de proveedores, mayor será el consumo de energía en el dispositivo móvil.
- La ejecución de nuestra aplicación móvil no interfiere en el uso común del teléfono, sea para realizar llamadas o usar otra aplicación, debido a que puede continuar ejecutándose en segundo plano.
- En el caso de nuestra propuesta, SIP no sólo es un protocolo de manejo de sesiones, sino también de autenticación de usuarios, ofreciendo ventajas sobre otros protocolos como HTTP.

5.2 Componentes de la aplicación móvil

Ya que se ha comprobado la funcionalidad de cada módulo de nuestra propuesta, es necesario mostrar cómo se conforma la aplicación implementada con el objetivo de demostrar cómo será utilizada por un usuario.

La interfaz gráfica debe ser lo más simple posible, ya que de acuerdo a los requerimientos, la ejecución de la aplicación requiere la mínima interacción del usuario y el reporte de ubicaciones debe realizarse de forma automática. Al desarrollar una aplicación móvil, los ejecutables se empaquetan en un archivo del tipo `.jar` (para el caso de Java ME) o `.apk` (para Android). Este archivo se envía para ser instalado en el dispositivo móvil por medio de un cable USB conectado a la computadora o a través de un medio inalámbrico, como Bluetooth o Internet. Cuando el usuario instala la aplicación Java, el sistema muestra datos relevantes como el origen o autor de la aplicación, los permisos que utiliza, descripción, tamaño, etc. Una vez instalada la aplicación, se agrega automáticamente al menú de aplicaciones existentes en el teléfono y al seleccionar el icono de la misma, se inicia su ejecución. Para el caso de Android, al enviar la aplicación al teléfono móvil, el sistema muestra los permisos que requieren autorización explícita del usuario, si se aceptan, la aplicación se instala, de lo contrario no.

Al arrancar la ejecución de nuestra aplicación se muestran las pantallas de la

Figura 5.8. Esta interfaz puede aprovecharse para mostrar mensajes al usuario o incluso mensajes de bitácora o *log*, para saber lo que está ocurriendo en la aplicación. Para el caso de Java se le indica al usuario que seleccione alguna opción del menú (mostrado en la Figura 5.9) y también se le muestra la opción `Exit` para salir de la aplicación.

Figura 5.8: Pantalla de inicio de la aplicación móvil

Figura 5.9: Menú de opciones

Las opciones contenidas en el menú permiten:

1. Configurar los parámetros e información del usuario (`Configuración`).
2. Limpiar los mensajes mostrados en la pantalla principal (`Limpia pantalla`).

La interfaz gráfica que es posible desarrollar en Android, permite una mejor integración con dispositivos que cuentan con pantalla táctil, de este modo la interacción del usuario con los menús de la aplicación de la aplicación, ocurre de forma más ágil e intuitiva.

Figura 5.10: Menú de opciones Android

Los parámetros que es posible configurar en el dispositivo son tres: posicionamiento, SIP y frecuencia de notificaciones (Ver Figura 5.11). Para el primer caso, el usuario puede especificar la precisión del proveedor de ubicación, en caso de disponer de varios, con la finalidad de filtrar las ubicaciones menos precisas. También en este apartado (Figura 5.12) se especifica el máximo tiempo para esperar que el proveedor determine la ubicación, de este modo puede indicarse que si después de 2 minutos no se ha obtenido la coordenada, deje de esperar al proveedor.

Figura 5.11: Menú de opciones

Figura 5.12: Parámetros de posicionamiento

La sección de SIP (Figura 5.13) es la más importante debido a que es aquí donde se especifican los datos de autenticación para poder enviar los datos al servidor. Si estos datos no son especificados, el servidor rechazará cualquier petición generada desde el dispositivo móvil.

Para el caso de Android, en el cual no fue posible probar la autenticación por medio del protocolo SIP, se utilizan los mismos datos de usuario y contraseña

(Figura 5.14), la única diferencia es que se utilizará en protocolo HTTP en lugar de SIP.

Figura 5.13: Parámetros SIP

Figura 5.14: Parámetros HTTP

Por último, en los parámetros de notificación se despliega una lista de opciones

con intervalos de tiempo (Figura 5.15) para que el usuario determine la frecuencia a la que se enviarán reportes de ubicación al servidor. Para el caso de Java, en cuanto el usuario elige un intervalo, se calendariza la ejecución del proceso mediante el cual se obtendrá la coordenada geográfica del dispositivo; además, se registra una notificación *Push* para el caso en que el usuario cierre la aplicación. La manera de comprobar que la notificación *Push* se realiza con éxito, se muestra en la Figura 5.16, donde se observa la solicitud de aprobación para que nuestra aplicación inicie de forma automática en un futuro.

Android realiza la ejecución periódica de esta tarea de forma automática, sin necesidad de usar notificaciones *Push* para obtener la ubicación del usuario cada determinado tiempo.

Figura 5.15: Intervalos de envío de coordenadas

Si el usuario opta por cerrar la aplicación, cada vez que se cumpla el intervalo de tiempo especificado, la aplicación se iniciará de forma automática para enviar los datos de la ubicación. En caso contrario, si el usuario deja la aplicación ejecutando en primer plano, podrá observar la información que se envía al servidor (Figura 5.17). En el servidor, se recibe el paquete tal y como se mostró en la Figura 5.4.

La aplicación desarrollada para la plataforma Android tiene el mismo funcionamiento, lo único distinto son los detalles gráficos mostrados por el dispositivo móvil.

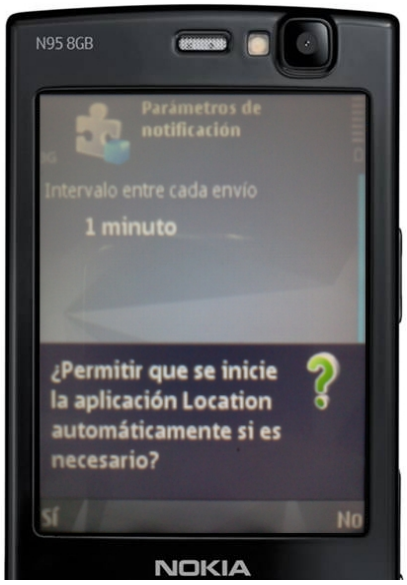


Figura 5.16: Solicitud de permisos *Push*

Figura 5.17: Envío de coordenada y respuesta del servidor

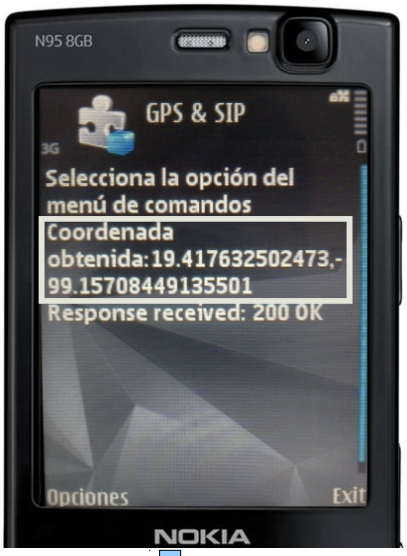
5.3 Pruebas del servidor

En el capítulo 4 se demostró que la información enviada desde el dispositivo móvil era recibida de forma exitosa en el servidor. Ahora se debe comprobar que esa información es almacenada con el objetivo de acceder a ella para su futuro

análisis.

Cuando el cliente obtiene su ubicación y la conectividad para enviarla hacia el servidor, el paquete de información viaja a través de la red hasta el servidor, llevando consigo los datos del usuario y de su ubicación actual. Una vez que se ha autenticado al usuario, la información se almacena en una base de datos diseñada a partir del modelo de entidad-relación descrito en el capítulo 3.

En la Figura 5.18 se muestra lo que ocurre en el servidor cada vez que se envía una coordenada desde el móvil (Observe que el valor de la coordenada geográfica es el mismo).




```
mysql> select * from Reporte where Usuarioid=1;
+-----------+---------------------+-----------------------------------+-------------+
| Usuarioid | fecha               | ubicacion                         | dispositivo |
+-----------+---------------------+-----------------------------------+-------------+
|         1 | 2011-05-26 17:30:55 | 19.417632502473,-99.15708449135501 |             |
+-----------+---------------------+-----------------------------------+-------------+
1 rows in set (0.00 sec)

mysql>
```

Figura 5.18: Almacenamiento de la coordenada de ubicación

El manejador de la base de datos nos permite obtener la información que buscamos con la ejecución del *query*: `SELECT * FROM Reporte WHERE Usuarioid=1`, esto significa que muestre todos los campos de los registros en la tabla '`Reporte`' cuyo campo '`Usuarioid`' sea igual a 1. El identificador de usuario

se obtiene de la tabla Usuario, como se muestra en la Figura 5.19, y es un número que aumenta cada vez que se ingresa un nuevo usuario al sistema.

```
mysql> SELECT * FROM Usuario WHERE nick='john';
+----+------+----------+-------------+-----------+-----------+-----------------------------------+
| id | nick | password | nombre      | apellidoP | apellidoM | email                             |
+----+------+----------+-------------+-----------+-----------+-----------------------------------+
|  1 | john | juank    | Juan Carlos | Perez     | Perez     | jperez@computacion.cs.cinvestav.mx |
+----+------+----------+-------------+-----------+-----------+-----------------------------------+
1 row in set (0.00 sec)
```

Figura 5.19: Registro de usuario en la base de datos.

Para que *Asterisk* autentique a los usuarios, es necesario que los datos también sean incluidos en el archivo *sip.conf*, el cual lista a los *peers* que pueden comunicarse con el servidor (Ver Figura 5.20). El nombre y contraseña de usuario deben ser iguales en ambas ubicaciones.

```
labsd@labsd-desktop:~$ more /etc/asterisk/sip.conf
[general]

[authentication]

[spa]
secret=welcome
type=friend
context=phones
host=dynamic

[john]
secret=juank
type=friend
context=phones
host=dynamic
nat=yes

[gsm]
secret=voip
type=friend
context=phones
host=dynamic
nat=yes
...
```

Figura 5.20: Archivo de usuarios de Asterisk.

En el caso del dispositivo Android, ocurren las mismas operaciones en el servidor, sólo que en lugar de ser Asterisk el que actualice la base de datos, el que se encarga de hacer esto es el *Servlet* que atiende las peticiones HTTP.

De este modo se almacenan uno a uno de los reportes de ubicación en la base de datos, con lo cual es posible construir un historial por cada usuario registrado

ya que cada coordenada se almacena con la fecha en la cual fue enviada. En la Figura 5.21 se muestra la ejecución del *query* `SELECT * FROM Reporte WHERE Usuarioid=1 AND fecha like '2011-05-26%'`, el cual obtiene todos los reportes cuya fecha comience con el día 26 de mayo de 2011. Observe que un mismo usuario puede enviar reportes desde distintos dispositivos (columna 'dispositivo').

```
mysql> SELECT * FROM Reporte WHERE Usuarioid=1 and fecha like '2011-05-26%';
+-----------+---------------------+-------------------------------------+-------------+
| Usuarioid | fecha               | ubicacion                           | dispositivo |
+-----------+---------------------+-------------------------------------+-------------+
|         1 | 2011-05-26 13:51:30 | 19.417632502473,-99.15708449135501  |             |
|         1 | 2011-05-26 17:14:06 | 19.41777269,-99.15676328            | LG-P500h    |
|         1 | 2011-05-26 17:14:56 | 19.41777269,-99.15676328            | LG-P500h    |
|         1 | 2011-05-26 17:15:36 | 19.41776771,-99.1567233             | LG-P500h    |
|         1 | 2011-05-26 17:30:55 | 19.417632502473,-99.15708449135501  |             |
|         1 | 2011-05-26 18:15:04 | 19.417727969999994,-99.15678657999999 | LG-P500h    |
|         1 | 2011-05-26 18:15:49 | 19.417748019999998,-99.15683615     | LG-P500h    |
+-----------+---------------------+-------------------------------------+-------------+
7 rows in set (0.00 sec)
```

Figura 5.21: Reportes filtrados por fecha

Para que los usuarios puedan visualizar estos datos, se implementó un *Servlet* para atender las peticiones vía Web. En el capítulo 4 se detalló el código fuente de este componente, a continuación se muestra la interfaz gráfica con la cual se comprueba su funcionamiento:

Figura 5.22: Ventana de autenticación Web

La Figura 5.23 muestra lo que alguien autorizado por el usuario `'john'` recibe como respuesta ante la consulta en el servidor Web:

Figura 5.23: Reporte de ubicaciones

5.4 Análisis de los resultados

Al demostrar y corroborar el funcionamiento de la propuesta, se concluye que es un mecanismo eficaz para determinar la ubicación del cliente en un tiempo mínimo. La transmisión de esta información hacia un servidor central también ocurre de forma inmediata, protegiendo la información del usuario mediante métodos de autenticación.

La información que produce el sistema puede ser fácilmente explotada para ofrecer servicios basados en la ubicación, debido a que su finalidad es reportar frecuentemente los cambios de ubicación del usuario. Un posible escenario de aplicación consistiría en mostrar las ubicaciones reportadas en un mapa y ofrecer el cálculo de rutas óptimas. Asimismo es posible calcular y analizar el tiempo invertido en algún trayecto ya que se cuenta con la hora de cada reporte de ubicación.

Servicios como *Google Places*© ya hacen uso de la aproximación de la ubicación para ofrecer información relevante al usuario. Asimismo se han desarrollado aplicaciones móviles que utilizan las notificaciones *Push* y la ejecución de procesos en segundo plano para rastrear dispositivos.

Sin embargo,ninguna de las soluciones existentes integra el concepto de comunicación oportunista como medio para obtener conectividad a Internet de forma automática. Tampoco buscan que el funcionamiento de la aplicación sea "invisible" para el usuario, como lo dictan los principios del cómputo ubicuo. Se cree que la inclusión de ambos conceptos es necesaria para mejorar y ampliar los servicios basados en la ubicación.

5.5 Discusión

Las pruebas realizadas al mecanismo de localización se aplicaron tanto en la parte cliente como en la parte servidor. En el caso de los dispositivos móviles, se evaluaron los mecanismos disponibles para obtener la posición geográfica y se tomaron métricas de latencia para determinar los límites de tiempo que garantizan el correcto funcionamiento de nuestro sistema. Asimismo se obtuvo el tiempo promedio en el cual nuestra implementación determina la ubicación del usuario.

La plataforma Android permite aproximar la ubicación a partir de la red celular e inalámbrica (WiFi), lo cual es una gran ventaja con respecto a Java ME ya que los reportes de ubicación serán más confiables y frecuentes. Se observó que la latencia del GPS integrado, depende sobre todo del hardware con el que cuente el equipo y no de la plataforma de desarrollo. También se verificó el funcionamiento del GPS asistido.

Se analizó minuciosamente la comunicación de extremo a extremo, entre cliente y servidor, detallando el contenido de los paquetes SIP y HTTP, con la finalidad de demostrar que el flujo de datos ocurría según los requerimientos de nuestra propuesta.

Fue posible implementar el envío cíclico de la información con variaciones de tiempo no significativas. Las pruebas nos permitieron llegar a ciertas conclusiones sobre el desempeño de la aplicación móvil (sección 5.3.1).

Posteriormente se detalló gráficamente el flujo de operación de la aplicación móvil, y se demostró la forma en que los reportes de ubicación son generados y enviados desde el cliente para su almacenamiento en el servidor. También se demostró la ejecución en segundo plano de la aplicación, lo cual evita la interrupción del uso normal que se le da al dispositivo.

En el caso del servidor, se verificó que la información recibida fuera almacenada de forma correcta en la base de datos, ya sea que provenga de *Asterisk* o del contenedor Web. Por último, se mostró la interfaz Web a través de la cual es posible rastrear un dispositivo y ver su historial de ubicaciones reportadas.

Capítulo 6

Conclusiones y trabajo futuro

A partir del trabajo presentado en este documento, es posible obtener las siguientes conclusiones y proponer el posible trabajo futuro a desarrollar, tomando como base nuestra propuesta.

6.1 Conclusiones

Se implementó un sistema con arquitectura cliente-servidor, en donde el servidor se encarga de recibir y almacenar los reportes de ubicación de clientes (dispositivos móviles), los cuales poseen distintas arquitecturas y sistemas operativos.

Cada cliente posee una aplicación móvil "inteligente", la cual busca los medios disponibles para determinar su ubicación geográfica así como la forma de comunicarse con el servidor del sistema. Para encontrar el medio ideal, nos basamos en los conceptos de la comunicación oportunista.

El desarrollo de la aplicación móvil implicó realizar una investigación y estudio exhaustivos de las tecnologías existentes, para comparar sus ventajas y desventajas. A partir de este análisis y de los requerimientos del proyecto, se decidió implementar la aplicación en la plataforma con mayor número de usuarios (Java ME) y en la que provee mejores interfaces de programación (Android).

La plataforma Java ME permitió comprobar el funcionamiento de las partes más importantes del sistema, siendo su única limitante la incapacidad de manipular las interfaces de conectividad inalámbrica disponibles en el dispositivo móvil. En caso de contar con un API para manejar estas interfaces, nuestra aplicación móvil sería funcional al 100%.

La plataforma Android permite un desarrollo de aplicaciones simple y muy completo, incluso al día de hoy es posible implementar por completo nuestra propuesta en esa plataforma, sin embargo no se cuenta con un dispositivo móvil

que tenga la última versión del sistema operativo (2.3) y que además soporte el desarrollo de aplicaciones SIP.

Al implementar la aplicación en dos plataformas distintas se minimizaron las desventajas de cada una, ya que de forma complementaria permitieron comprobar el funcionamiento del mecanismo de localización. Este mecanismo propuesto y desarrollado cumple con los objetivos planteados en un inicio, ya que permite llevar un historial de ubicaciones desde donde se ha reportado el usuario, así como su localización en cualquier momento.

Consideramos que es importante anticiparse al crecimiento exponencial de la información de los usuarios, visualizando el escenario en el cual una gran cantidad de usuarios envía reportes muy frecuentemente, por ejemplo, si el usuario envía reportes cada minuto y deja que la aplicación se ejecute de forma ininterrumpida en el dispositivo, se generarían casi 1,500 reportes diarios, lo que es equivalente a más de 43,000 reportes al mes, por cada usuario o dispositivo.

De acuerdo a las pruebas realizadas, se puede concluir que el envío continuo de reportes de ubicación implica mayor consumo de los recursos del dispositivo móvil, por lo tanto se recomienda establecer un intervalo más amplio para enviar la información.

Se diseñó un algoritmo "oportunista" mediante el cual el teléfono busca puntos de acceso a Internet en su cercanía y deja al final la opción de envío a través de la red celular, debido a que implica costo y es menos fiable en cuanto a la velocidad de transmisión. La ejecución de este algoritmo también consume tiempo y energía en el dispositivo, ya que se deben buscar las redes inalámbricas cercanas y analizar sus propiedades, con el objetivo de tener acceso a Internet en el menor tiempo posible.

El servidor *Asterisk* permite que la gestión de los usuarios y sus reportes de ubicación se realice de forma transparente, debido a que cada uno de los reportes es una petición de registro ante el servidor, con lo cual es equivalente a que el usuario inicie sesión continuamente, ya sea desde la misma red IP o desde distintas redes.

La propuesta de utilizar SIP como protocolo para lograr la comunicación entre cliente y servidor, se vio reforzada y justificada al momento en que se intentó implementar el mismo funcionamiento mediante otro protocolo (HTTP). En este caso es necesario contar con una capa de seguridad, lo cual se traduce en una mayor complejidad del sistema, al ser necesario incluir mecanismos de autenticación de usuarios y el uso de canales seguros de transmisión de la información.

Consideramos que el mecanismo de localización es completamente escalable, debido a que tanto los dispositivos móviles como las redes de telecomunicaciones poseen cada vez más capacidades y es posible que tanto la obtención de la

ubicación como su envío hacia el servidor, se realicen en menor tiempo.

6.2 Trabajo futuro

La continuación inmediata que se le puede dar a esta propuesta, es el desarrollo completo de la aplicación móvil en la última versión del sistema operativo Android, la cual incluye el soporte SIP.

Al contar con los distintos algoritmos que conforman la aplicación móvil, es posible portarla a las demás plataformas móviles que no fueron consideradas en este trabajo, esto con la finalidad de probar su correcto funcionamiento e incluir al resto de dispositivos móviles.

Una línea de trabajo con amplio crecimiento es la inclusión de otros medios de transmisión, como Bluetooth, para el envío oportunista de datos. En lugar de que el dispositivo móvil busque puntos de acceso a Internet, es posible que busque dispositivos que cuenten con acceso a la red y a los cuales sea posible confiarles los datos de la ubicación. De este modo se busca abatir la falta de conectividad en el propio dispositivo, mediante la "ayuda" de otros dispositivos similares.

En nuestra propuesta nos centramos en dispositivos móviles, sin embargo es posible extender el mecanismo para que el usuario pueda reportar su ubicación desde una computadora de escritorio. Actualmente existen herramientas de desarrollo que determinan la ubicación de un usuario conectado a Internet mediante el uso de complementos instalados en su navegador Web. De esta forma los usuarios que no cuenten con un dispositivo móvil capaz de ejecutar nuestra aplicación, pueden hacer uso del sistema.

Para el caso del servidor, es posible incluir mayor funcionalidad para la gestión de datos de ubicación, *e.g.* mostrar en un mapa las coordenadas reportadas por un dispositivo, filtrar las ubicaciones por intervalo de tiempo o cantidad de reportes y mostrar el estado de un dispositivo, esto último se puede lograr mediante el manejo de presencia que ofrece SIP.

La principal ventaja de este trabajo de tesis es que su aplicación en un entorno real es directa, por lo tanto es posible distribuir e incluso comercializar la aplicación en los medios existentes como la tienda de aplicaciones de Android y las páginas de Internet que distribuyen aplicaciones y juegos para celulares que soportan Java. Es posible que el usuario móvil le encuentre utilidad en distintos escenarios como el rastreo de personas en caso de emergencia, la localización de dispositivos extraviados y el análisis de rutas de desplazamiento.

Bibliografía

[1] A. Brimicombe, C. Li, *Location-based services and geo-information engineering*, 1ª edición, UK: Wiley, 2009, ISBN: 0470857374.

[2] A. Chaintreau, A. Mtibaa, L. Massoulie, y C. Diot, "The diameter of opportunistic mobile networks", en *International Conference On Emerging Networking Experiments and Technologies*, Nueva York, pp. 1-2, 2007.

[3] A. El-Rabbany, *Introduction to GPS: the Global Positioning System*, 1ª edición, USA: Artech House, 2002, pp. 1-3, ISBN: 1580531831 .

[4] A. Küpper, *Location-based services: fundamentals and operation*, 1ª edición, UK: John Wiley and Sons, 2005, pp. 234, ISBN: 0470092319 .

[5] A.K. Talukder, H. Ahmed, R.R. Yavagal, *Mobile Computing: Technology, Applications and Service Creation*, 2ª edición, India: Tata McGraw-Hill, 2010, pp. 2-16, ISBN: 0070144575 .

[6] B.W. Perry, *Java servlet and JSP cookbook*, 1ª edición, EU: O'Reilly Media, 2004, pp. 2, ISBN: 0596005725.

[7] C. Boldrini, M. Conti, A. Passarella, "Modelling data dissemination in opportunistic networks", en *International Conference on Mobile Computing and Networking*, California, pp. 89, 2008.

[8] C. Oggerino, *High availability network fundamentals*, 1ª edición, EU: Cisco Press, 2001, pp. 5, ISBN: 1587130173 .

[9] C.S.R. Prabhu, *Mobile Computing*, 1ª edición, India: Universities Press, 2004, pp. 3-6, 114, ISBN: 8173714045 .

[10] D. Gomillion y B. Dempster, *Building Telephony Systems with Asterisk*, 1ª

edición, UK: Packt Publishing, 2005, pp. 5, ISBN: 1904811159 .

[11] D. Preuveneers, A. Yasar, y Y. Berbers, "Architectural styles for opportunistic mobile communication", en *International Conference on Mobile Technology, Applications and Systems*, Taiwán, pp. 1, 2008.

[12] H. Sinnreich y A.B. Johnston, *Internet communications using SIP*, 2ª edición, USA: Wiley Publishing, 2006, pp. 1-124, ISBN: 0471776572 .

[13] I. Sommerville, *Ingeniería de Software*, 6ª edición, México: Pearson Educación, 2001, pp. 8-222, ISBN: 020139815X .

[14] J. Figueiras, Simone Frattasi, *Mobile Positioning and Tracking: From Conventional to Cooperative Techniques*, 1ª edición, UK: John Wiley and Sons, 2010, pp. 186, ISBN: 0470694513 .

[15] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, SIP: Session Initiation Protocol, RFC 3261, 2002.

[16] J. Van Meggelen, L. Madsen, J. Smith, *Asterisk: The Future of Telephony*, 2ª edición, EU: O'Reilly, 2007, pp. 37188 , ISBN: 0596510489 .

[17] M. Dawson, J. Winterbottom, M. Thomson, *IP Location*, 1ª edición, EU: McGraw-Hill, 2007, pp. 1-3, ISBN: 0072263776.

[18] M. Ilyas y I. Mahgoub, *Mobile computing handbook*, 1ª edición, USA: CRC Press, 2005, ISBN: 0849319714 .

[19] M. Mallick, *Mobile and Wireless Design Essentials*, 1ª edición, USA: Wiley, 2003, pp. , ISBN: 0471214191 .

[20] M. Weiser, "The Computer for the Twenty-First Century", *Scientific American*, vol. 265, no. 3, pp. 94-100, Septiembre 1991.

[21] P. Basu, C. Chau, "Opportunistic forwarding in wireless networks with duty cycling", en *International Conference on Mobile Computing and Networking*, 2008.

[22] P. Hui, A. Lindgren, "Phase transitions of opportunistic communication", en *International Conference on Mobile Computing and Networking*, California, pp.

73-78, 2008.
[23] R. Sparks, "SIP: Basics and beyond", *ACM Queue*, vol. 5, no. 2, pp. 22-33, Marzo 2007.

[24] Reza B'Far, *Mobile Computing Principles*, 1ª edición, UK: Cambridge University Press, 2005, pp. 7-25, ISBN: 0521817331.

[25] S. Posland, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, 1ª edición, UK: Wiley, 2009, pp. 2, ISBN: 0470035609 .

[26] S. Tarkoma, *Mobile Middleware: Architecture, patterns and practice*, 1ª edición, UK: Wiley, 2009, pp. 37 64 , ISBN: 0470740736 .

[27] S. Wu y Y. Tseng, *Wireless Ad-Hoc Networking*, 1ª edición, USA: Auerbach Publications, 2007, pp. 483, ISBN: 0849392543 .

[28] T. Phan, R. Montanari, P. Zerfos, *Mobile Computing, Applications, and Services*, 1ª edición, USA: Springer, 2010, pp. 113 - 117, ISBN: 3642126065.

[29] T. Wallingford, *VoIP Hacks*, 1ª edición, USA: O'Reilly, 2005, ISBN: 0596101333 .

[30] V. Jeyasri Arokiamary, *Mobile Computing*, 1ª edición, India: Technical Publications Pune, 2008, pp. 2, ISBN: 8184315082 .

[31] Android, "Android Developer Guide", Marzo 2011, Disponible en: http://developer.android.com/guide/index.html, Fecha de acceso: 5 de abril de 2011.

[32] Oracle, "Mobile Java", Mayo 2010, Disponible en: http://www.java.com/es/download/faq/, Fecha de acceso: 5 de abril de 2011.

[33] Blog, "J2ME vs. Symbian C++. Which one to choose from?", Junio 2009, Disponible en: http://blog.cogniance.com/65/, Fecha de acceso: 5 de abril de 2011.

[34] VisionMobile, "Developer Economics 2010", Londes, UK, 2010.